Aggressive Aerial Grasping using a Soft Drone with Onboard Perception

by

Samuel Ubellacker

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Samuel Ubellacker
Department of Mechanical Engineering
August 27, 2023

Certified by: Luca Carlone
Associate Professor of Aeronautics and Astronautics, Thesis Supervisor

Certified by: John Leonard
Professor of Mechanical and Ocean Engineering, Thesis Reader

Accepted by: Nicolas G. Hadjiconstantinou
Chairman, Committee for Graduate Students

Aggressive Aerial Grasping using a Soft Drone with Onboard Perception

by

Samuel Ubellacker

Submitted to the Department of Mechanical Engineering
on August 27, 2023 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

ABSTRACT

Contrary to the stunning feats observed in birds of prey, aerial manipulation and grasping with flying robots still lack versatility and agility. Conventional approaches using rigid manipulators require precise positioning and are subject to large reaction forces at grasp, which limit performance at high speeds. The few reported examples of aggressive aerial grasping rely on motion capture systems, or fail to generalize across environments and grasp targets. We describe the first example of a soft aerial manipulator equipped with a fully onboard perception pipeline, capable of robustly localizing and grasping visually and morphologically varied objects. The proposed system features a novel passively closing tendon-actuated soft gripper that enables fast closure at grasp, while compensating for position errors, complying to the target-object morphology, and dampening reaction forces. The system includes an onboard perception pipeline that combines a neural-network-based semantic keypoint detector with a state-of-the-art robust 3D object pose estimator, whose estimate is further refined using a fixed-lag smoother. The resulting pose estimate is passed to a minimum-snap trajectory planner, tracked by an adaptive controller that fully compensates for the added mass of the grasped object. Finally, a finite-element-based controller determines optimal gripper configurations for grasping. Rigorous experiments confirm that our approach enables dynamic, aggressive, and versatile grasping. We demonstrate fully onboard vision-based grasps of a variety of objects, in both indoor and outdoor environments, and up to speeds of 2.0 m/s— the fastest vision-based grasp reported in the literature. Finally, we take a major step in expanding the utility of our platform beyond stationary targets, by demonstrating motion-capture-based grasps of targets moving up to 0.3 m/s, with relative speeds up to 1.5 m/s.

Video Attachment: https://www.youtube.com/watch?v=HF4M7TooqfE

Thesis Supervisor: Luca Carlone
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank the many colleagues, friends, and family who helped make this project possible.

First, I would like to thank Aaron Ray, who did substantial design work of the quadrotor, implemented the fixed-lag smoother and moving polynomial planning, helped with the experimental evaluation, writing, system integration, and with both hardware and software debugging. Aaron's contributions to the project are invaluable and its safe to say the project would not have made it to this state without his help. On top of that, it was an amazing experience to have met and worked with Aaron. He introduced me to the world of fine dining, squash, and was always positive even in the project's roughest times. I shared many laughs and experiences with Aaron, both in and out of the lab.

I would also like to thank the other direct collaborators of this work. James Bern provided insights into control of soft manipulators, implemented the FEM-based optimization, and helped with the writing. Jared Strader provided insights into target pose filtering, helped with the experimental evaluation, and aided design decisions.

I am grateful for the contributions of Mubarik Mohamoud for the initial quadrotor design, Benjamin Evans for work assembling and manufacturing the gripper electrical and mechanical components, William Menken and Neel Mondal for electrical design of the gripper circuit board, and Joshua Fishman for useful discussion on soft aerial manipulation. I would also like to thank Varun Murali, Parker Lusk, and Nathan Hughes for feedback on initial drone design and system troubleshooting.

Finally, I would like to thank my supervisor, Luca Carlone, who provided invaluable supervision and guidance, and was a pleasure to have worked with. Luca has always been a supportive and understanding supervisor, while also being very engaged in the project and geniunely invested in the direction of both my research and general career. I couldn't imagine my MIT experience with another supervisor.

I would like to thank the many friends I met along the way both in and out of grad school. Anubhav, Jess, Will, Duncan taught me how to live life a litte more, Kevin, Pat, Adam were there for morning manatee practices, Emilio, Jordan, Justin for being great roomates, and Nick, Michael, and Henry for joining me on long bike rides.

Last of all, I would like to thank my family and parents. It goes without saying that I wouldn't be here if it weren't for the support and love of my parents. The older I get, the more appreciation I gain for the effort and time you invested in raising me.

Contents

Title page 1

Abstract 3

Acknowledgments 5

List of Figures 9

List of Tables 11

1 Introduction 13

1.1 Motivation 13

1.2 Contribution 15

2 System Overview and Experimental Evaluation 18

2.1 System Overview 18

2.1.1 Soft Gripper Design and Modeling 20

2.1.2 Perception System Design 23

2.2 Results: Grasping Static Targets 25

2.2.1 Grasp Success Rate 25

2.2.2 Error Analysis 27

2.2.3 Grasp Speed Analysis 29

2.2.4 Outdoor Grasps 30

2.3 Results: Grasping Moving Targets 33

3 Hardware Design and Software Methods **35**

3.1 Hardware Design 35

3.1.1 Drone Design 35

3.1.2 Gripper Design 36

3.2 Target Object Perception and Pose Estimation 36

3.2.1 Semi-Automated Keypoint Annotation 37

3.2.2 Keypoint Detection 38

3.2.3 Robust Point Cloud Registration 39

3.2.4 Fixed-Lag Smoother 40

3.3 Trajectory Optimization and Control 41

3.3.1 Minimum-Snap Polynomial Trajectory Optimization 41

3.3.2 Adaptive Quadrotor Control 42

3.3.3 FEM-Based Gripper Control 43

3.3.4 Measuring Sources of Error 45

4 Conclusion **47**

A Supplementary Figures **50**

B Design Guidelines and Error Analysis **61**

C Evaluating Reference Polynomial in Object Frame **64**

References **67**

List of Figures

1.1 **Overview of the Soft Drone.** 16

2.1 **Grasping pipeline overview.** 19

2.2 **FEM modeling and objective functions.** 21

2.3 **Keypoint-based object pose estimation.** 24

2.4 **Overview and timelapses of experiments with static targets.** 26

2.5 **Static grasp experiments results.** 28

2.6 **Timelapses of experiments with moving targets.** 31

2.7 **Moving grasp experiments results.** 32

A.1 **Tracking error breakdown for experiments with static target (continued below).** 50

A.1 **Tracking error breakdown for experiments with static target (continued).** . . . 51

A.2 **Tracking error breakdown for experiments with moving target.** 52

A.3 **Gripper dimensions and error bounds.** 53

A.4 **Passively closed finger mold.** 57

A.5 **Automated data annotation tool.** 58

A.6 **Keypoint detector training environments.** 59

List of Tables

A.1 **Longitudinal errors at the time of grasp.** 54
A.2 **Lateral errors at the time of grasp.** 55
A.3 **Vertical errors at the time of grasp.** 56
A.4 **Drone specifications.** 57
A.5 **Keypoint detector training parameters.** 59
A.6 **Fixed-lag smoother parameters.** 60

Chapter 1

Introduction

1.1 Motivation

Current quadrotor platforms can fly at high speeds, maneuver with great agility, and carry a wide range of payloads, but their ability to *interact* with the world remains limited. Most quadrotor platforms sense their environment for applications such as navigation, inspection, or videography [1]–[4]. However, many tasks require interacting with the environment, and quadrotors built to accomplish such tasks must handle complicated time-varying contact dynamics and have the ability to perceive task-relevant features of the environment. Recent work has increasingly focused in this direction, including a number of perching mechanisms [5]–[7], and full systems for applications such as contact-based inspection [8], and tree eDNA collection [9]. These tasks require contact with the environment, but not necessarily changing the environment. Other work has explored using quadrotors for building structures [10], [11], but requires external localization to function. Our work enables quadrotors to take a more active role in shaping the world around them by picking up and moving objects without the need for external motion capture infrastructure, expanding the kinds of tasks that can be accomplished by such platforms.

Building an aerial grasping system requires solving problems at the intersection of mechanical design, perception, motion planning, control, and manipulation. There is extensive prior work

toward solving each of these problems individually. Many works focus on gripping or grasping mechanisms for attaching to conventional drones [5]–[7], [12], [13] or helicopters [14]. Recent work has examined a closer coupling between the drone and gripper design [5], [15]. The control of aerial manipulators has been studied in [16]–[19], and recently for related tasks such as catching balls in flight [20] and grabbing other drones out of the air [21]. Vision-based object pose estimation and tracking are widely studied problems in robotics, and many existing methods are relevant for estimating the target's position for aerial manipulation; notable examples applied to aerial grasping include [22]. While each of these works reports promising advances in individual subsystems, *aggressive and versatile* aerial manipulation still remains out of reach, and prior work falls short of presenting a system that can perform aggressive grasping without strong assumptions about the target object and the environment.

Combining perception, control, and manipulation to achieve aggressive and versatile aerial manipulation presents several challenges. First, conventional approaches using rigid manipulators require precise positioning and are subject to large reaction forces at grasp, which limit performance at high speeds. Second, in contrast to systems that passively sense the environment, a quadrotor manipulating objects in the environment must adapt to changing dynamics, in particular since the added mass of the grasped object might be non-negligible compared to the mass of the drone itself. Third, if such systems are to be generally useful, they must also function without perfect state information from external motion capture systems; therefore they must rely on onboard perception and their perception system has to be resilient to the manipulator partially occluding the onboard camera. These problems must be solved in real-time, to enable efficient operation, and with the limited onboard computation, to circumvent the need for an external infrastructure.

The few reported examples of aggressive aerial grasping rely on motion capture systems, or fail to generalize across environments or grasp targets. For instance, Thomas et al. [16] demonstrate aggressive grasping in a motion capture system, but they assume a suspended target to avoid the risk of undesired contact forces with the surface the object is lying on. Fishman et al. [23] demonstrate dynamic grasping with a soft aerial gripper, but the system requires external local-

ization for the quadrotor and target. Existing vision-based aerial manipulation systems reduce errors by using a motion-capture system for the drone's self-localization [24], by flying at slower speed [25], [26], or by making strong assumptions about the target shape that cannot be extended to arbitrary objects [27], [28]. None of these works combines vision-based drone localization and target pose estimation, the ability to pick up targets with meaningful forward velocity, and fully onboard computation.

1.2 Contribution

We posit that aerial grasping can be improved by focusing on *soft* grasping mechanisms rather than trying to achieve extremely precise positioning with a rigid manipulator. We draw inspiration from biological systems (*e.g.,* birds), which utilize a combination of rigid and soft tissues to achieve unparalleled agility and robustness [29]. Soft robotic grippers can passively conform to the grasped object, reduce the need for explicit grasp analysis, and weaken the coupling of flight and manipulation dynamics; this is an example of *morphological computation*, *i.e.,* the exploitation of passive mechanical elements to supplement explicit control [30]. In this work, we combine a rigid quadrotor platform with a soft robotic gripper. Rigid and soft robots can operate together synergistically [31] as hybrid systems that get the best of worlds: our system combines the speed and agility of a rigid quadcopter with the natural compliance and robustness of a soft robotic gripper, and harnesses control methodologies that enable its rigid and soft components to work together.

Our first contribution is to develop a *soft, quadrotor-mounted gripper* with passively closing and tendon-actuated foam fingers. Our design enables fast closure at grasp, while compensating for positions errors, complying to the target-object morphology, and dampening reaction forces. Our second contribution is to develop a real-time and fully onboard perception pipeline for drone and target state estimation. Our perception pipeline combines a neural-network-based semantic keypoint detector with a state-of-the-art method for robust 3D object pose estimation and a fixed-lag smoother. Finally, we integrate a minimum-snap grasp trajectory planner with an adaptive

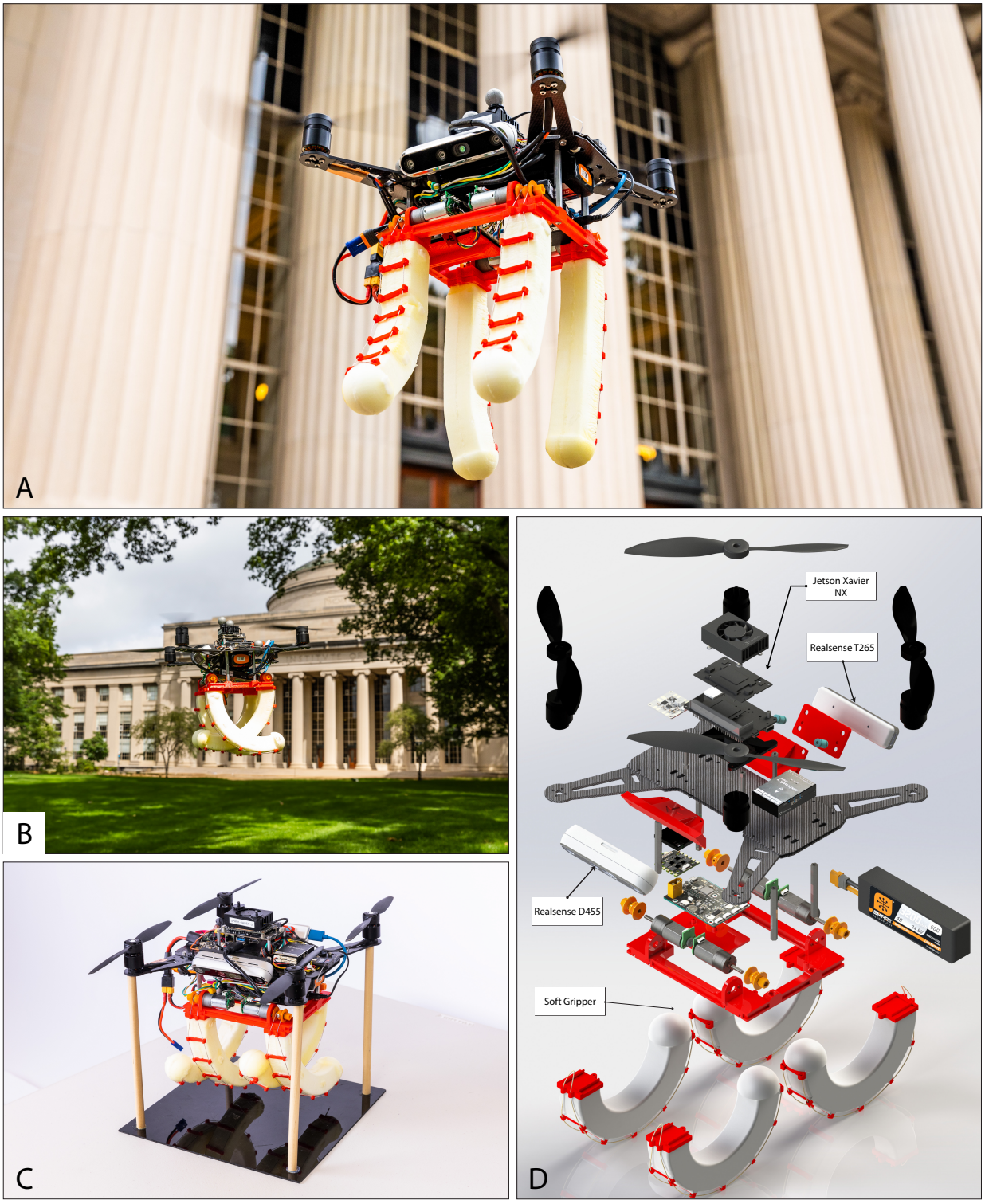


Figure 1.1: Overview of the Soft Drone. (**A-B**) The proposed Soft Drone platform flying outdoors. (**C**) Front view of the Soft Drone platform, with the soft gripper in the passively closed state. (**D**) Exploded CAD view of the quadrotor and gripper, including onboard sensing and computation.

controller for the drone and a finite-element-based approach for the soft gripper control. The adaptive controller is able to adjust to quickly changing dynamics, caused by the grasped object and related aerodynamic effects (e.g., ground effect and down-wash resulting from the grasped object); the finite-element-based approach computes optimal configurations for the soft gripper that are key for high-speed grasping.

We evaluate our soft drone with onboard perception when grasping a variety of objects in both indoor and outdoor environments. We present extensive experimental results across 180 flight tests. In contrast to prior work, our system performs all computation on board, can fly with or without a motion capture system, and can perform aggressive grasping of static targets at up to 2.0 m/s. Moreover, we show that when operating within a motion capture system the soft drone can also grasp objects from *moving* platforms: a quadruped robot carrying a med-kit moving forward at 0.3 m/s, and a rotating turntable with a relative grasp speed of 1.5 m/s.

Chapter 2

System Overview and Experimental Evaluation

We have designed and built a combined quadrotor and soft gripper platform (Fig. 1.1A,B) able to aggressively grasp different object types at high speeds. The gripper mechanism consists of four flexible, tendon-actuated foam fingers to provide robust grasp performance. The quadrotor uses two onboard cameras (Fig. 1.1C,D) to estimate its own state and a target object's pose, which enables operation without external localization infrastructure such as a motion-capture system. All perception, planning, and control algorithms run onboard the quadrotor. To demonstrate the effectiveness of our system, we performed 180 flights, where we measured grasping performance and provided an ablation of our main design choices. In this section, we provide an overview of our system (Section 2.1) and then discuss the results of our flights with both static (Section 2.2) and moving targets (Section 2.3). More technical details are postponed to Section 3, and visualizations of the system and experiments are shown in Movie 1.

2.1 System Overview

A successful object grasp requires the drone to detect and estimate the object's pose, plan a feasible grasping trajectory, and track that trajectory while facing complicated contact dynamics and other

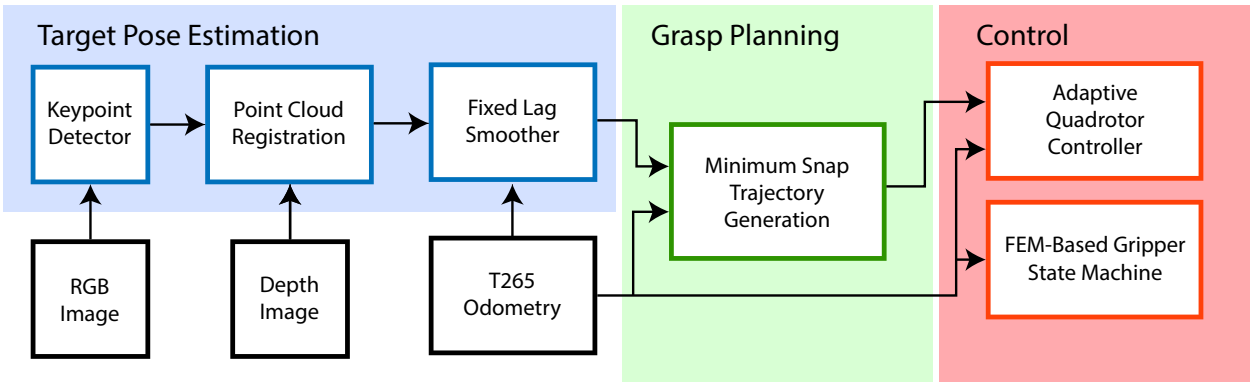


Figure 2.1: Grasping pipeline overview. An accurate estimate of the target and drone's pose is first identified and used to plan the polynomial grasp trajectory. This trajectory is tracked with an adaptive quadrotor controller, and the soft gripper states are dictated by an FEM-based optimization.

external disturbances.

Our system (Fig. 2.1) is designed to perform fully onboard perception, using a RealSense D455 color and depth camera for object pose estimation and a RealSense T265 stereo fisheye camera for visual-inertial odometry (VIO) for drone state estimation [32]. After estimating the target's and drone's pose in the global frame,[1] the quadrotor plans a minimum-snap polynomial trajectory [33] connecting its initial hover point, a grasp point directly above the target, and a terminal hover point. This reference trajectory is then tracked by an adaptive flight control law [34] which learns to compensate for the additional mass after grasp and for unmodeled aerodynamic effects. The gripper's finger positions are controlled along the trajectory using a finite-element-based approach [29], [35] governed by an objective functions that balances target object visibility and grasp robustness. The object pose estimation and trajectory planning algorithms run on a Jetson Xavier NX, the low-level flight control runs on a Pixhawk micro-controller, and the visual-inertial odometry pipeline runs on the RealSense T265 camera module.

Below, we provide an overview of our soft gripper design (Section 2.1.1) and the onboard perception system (Section 2.1.2), while we postpone the details to Section 3.

[1] While some aerial grasping approaches based on visual servoing avoid the need for an explicit world-frame target estimate [27], [28], we form a global estimate of the target's pose to remove the constraint that the target must always be visible in the camera's field of view, which is a restricting assumption in particular right before grasping.

2.1.1 Soft Gripper Design and Modeling

Eliminating all estimation and tracking errors from the perception and control systems is infeasible, so the grasp mechanism must be robust to imprecise positioning to enable reliable grasps. Collisions between the fingers and the ground present a particular challenge. Small amounts of error in the planned or executed trajectory lead to the manipulator contacting the ground. With a traditional rigid manipulator, these unexpected contact forces induce a moment on the quadrotor that results in it rapidly pitching forward, leading to a crash or failed grasp [23].

We propose to solve this issue with *compliant* fingers for picking up the object (Fig. 2.2A). Such fingers weaken the coupling between ground contact and drone dynamics, enabling steadier flight in the event of ground contact (Movie 1). Soft fingers also provide robustness for grasping the object: they conform to the object morphology and do not require precisely chosen grasp points.

Our finger design is made of foam molded in a closed position. Soft foams, especially castable expanding polyurethane, have been recently harnessed to produce highly-deformable soft robots with favorable strain-stress ratios [36]. A cable connected via eyelets along the outside edge of the finger controls its shape (Fig. 2.2B). As the motor at the base of the finger turns, the cable shortens and pulls the finger into an open position. This passively closed finger design has the advantage that the closing speed is limited only by the stiffness of the finger and no-load speed of the motor, in addition to lower power draw when the fingers are closed. The resulting finger design is mechanically quite simple compared to more traditional rigid finger designs that require actuation at each joint. In comparison to our previous soft finger design [23] that required two motors working in synchronization to open and close each finger, our current design has half the number of motors and is much easier to fabricate and control. Now each finger only has a single degree of freedom, yet it can assume nontrivial shapes (Fig. 2.2C).

The positioning of the fingers is important for a successful grasp. We model the body of the gripper as a finite element mesh, and the cables as unilateral springs running through via points in the mesh. Given a feasible choice of real-world control inputs, e.g., motor angles $\boldsymbol{u}$, this approach

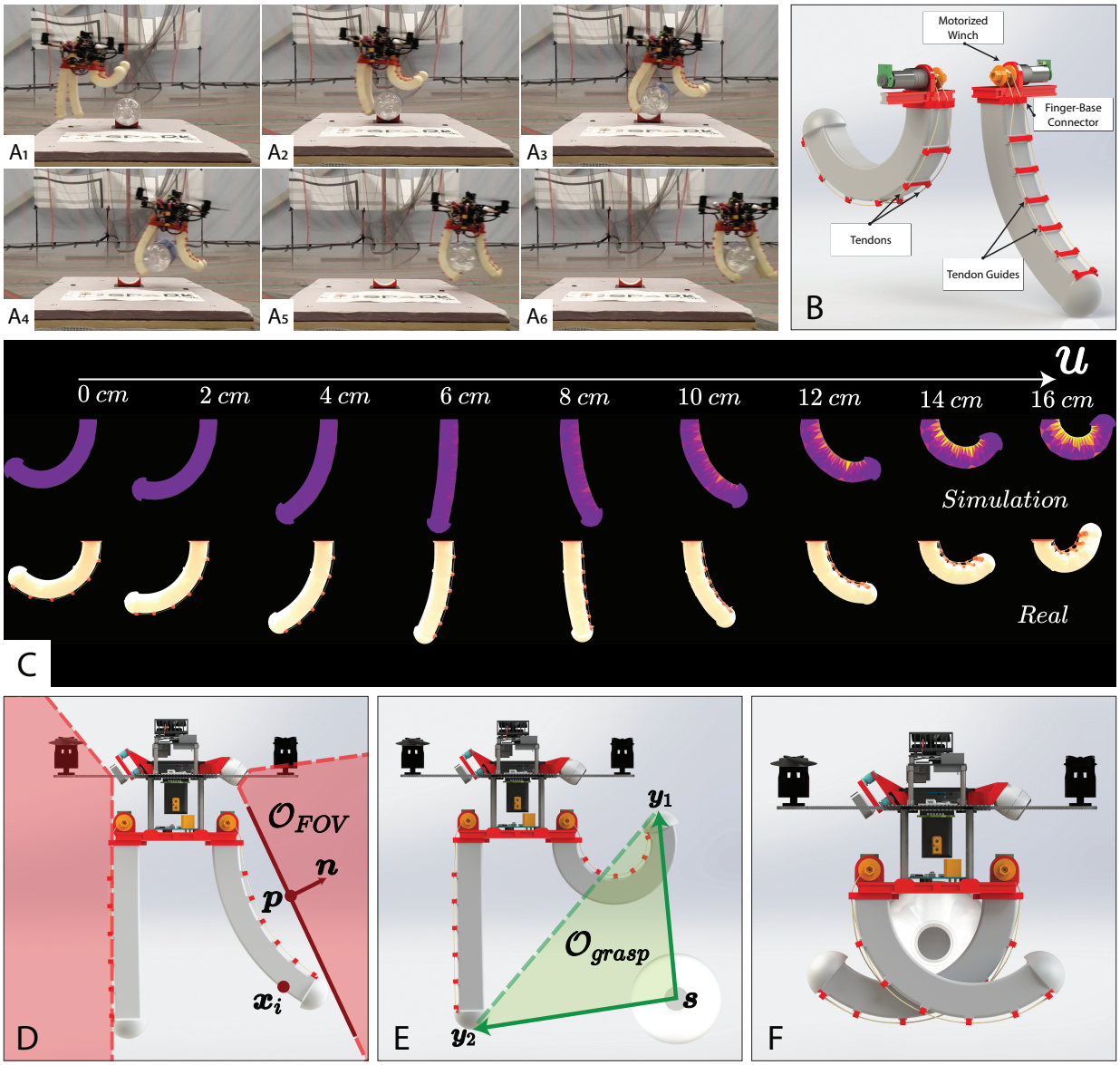


Figure 2.2: FEM modeling and objective functions. (**A**) Still frames from a grasp with over 2 m/s forward velocity. The compliance of the soft fingers mitigates the high reaction forces faced at this speed (**A$_3$**) and the gripper configuration reduces the sensitivity of the gripper closing timing. (**B**) Soft finger in its passively closed state (left) and partially open state (right). The finger opening is actuated by a motorized winch contracting a tendon fixed to the tip of the finger and routed through guides. (**C**) The finger shape predicted by the FEM model based on cable retraction length u (top) matches the actual finger shape for the same cable length (bottom). (**D**) Target observation state: the field-of-view objective, $\mathcal{O}_{\text{FOV}}$, drives the mesh nodes, including the node with position $\mathbf{x}_i$, out of the half space bounded by the plane defined by point $\boldsymbol{p}$ and normal $\boldsymbol{n}$ and corresponding to the cameras' frustum. (**E**) Pre-grasp state: the grasp margin objective, $\mathcal{O}_{\text{grasp}}$, maximizes the area enclosed by the grippers fingertips $\mathbf{y}_1$ and $\mathbf{y}_2$ and the target point $\mathbf{s}$. (**F**) Gripper closed configuration: the fingers return to their passively closed state.

to soft robotic modeling can accurately predict the real finger's deformed shape. This is done by minimizing the total energy of the system, to find a statically stable deformed mesh position, following the approach in [29], [35]:

$$\boldsymbol{x}(\boldsymbol{u}) = \arg\min_{\boldsymbol{x}} E(\boldsymbol{u}, \boldsymbol{x}). \tag{2.1}$$

In addition to helping us predict the gripper's motion, the FEM-based model can be harnessed in a nested optimization to do control. In this work, we find optimal control inputs by minimizing a suitable objective function:

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \Big(\mathcal{O}_{\text{grasp}}(\boldsymbol{x}(\boldsymbol{u})) + \mathcal{O}_{\text{FOV}}(\boldsymbol{x}(\boldsymbol{u})) \Big), \tag{2.2}$$

where the objective is designed to produce control inputs that deform the gripper into a shape $\boldsymbol{x}(\boldsymbol{u}^*)$ that maximizes the area enclosed by the fingertips and the target's centroid (as quantified by the term $\mathcal{O}_{\text{grasp}}(\boldsymbol{x}(\boldsymbol{u}))$ in Eq. 2.2, see Section 3 for a complete mathematical expression) while also not occluding the quadcopter's field of view (as quantified by the term $\mathcal{O}_{\text{FOV}}(\boldsymbol{x}(\boldsymbol{u}))$). The exact choice of objective depends on the different phases of the grasp, as discussed below. The optimization is performed offline to obtain optimal motor angles, which are then commanded during the grasp procedure.

When the drone is planning the grasp trajectory (Fig. 2.4A-1), the front fingers should not be in the front camera's field of view or they may obscure the object (Fig. 2.2D). The back fingers also need to stay out of the navigation camera's field of view, to avoid interfering with the drone localization system. We refer to this configuration as *target observation state* (Fig. 2.4B-1), and set $\mathcal{O}_{\text{FOV}}(\boldsymbol{x}(\boldsymbol{u})) = \mathcal{O}_{\text{FOV}_R}(\boldsymbol{x}(\boldsymbol{u})) + \mathcal{O}_{\text{FOV}_F}(\boldsymbol{x}(\boldsymbol{u}))$ as the sum of the FOV constraints for the rear (T265) and front (D455) cameras.

Right before grasping the target (what we call the *pre-grasp state*, Fig. 2.4B-2), we can remove the front camera FOV constraint because we have global knowledge of the target's location. Therefore, $\mathcal{O}_{\text{FOV}}(\boldsymbol{x}(\boldsymbol{u})) = \mathcal{O}_{\text{FOV}_R}(\boldsymbol{x}(\boldsymbol{u}))$, and the front fingers can reach the unconstrained grasp

configuration, which increases the grasp volume and greatly improves our position error margins (Fig. 2.2E). We note that this objective function also results in the rear fingers pointing downward; if the fingers are closed too late, the target will still be caught up by the back fingers and a successful grasp may still be possible (Fig. 2.2A).

To initiate the grasp closure, the tendons are simply commanded back to their rest positions and the fingers return to the passively closed state, thus enclosing the target (Fig. 2.2F, Fig. 2.4B-3).

2.1.2 Perception System Design

Our perception system enables global position and orientation estimation of the target object (Fig. 2.3). It is important to estimate the target's orientation in addition to position, as some objects are much more difficult to pick up along certain axes (*e.g.,* the two-liter bottle in Fig. 2.3B). The perception pipeline assumes that the shape (more precisely, the CAD model) of the objects to be picked up is known ahead of time, but the framework is generalizable to arbitrary objects.

A deep-neural-network-based keypoint detector can be rapidly trained with an automated data collection tool (Fig. 2.3A) and predicts a set of keypoints in each color image collected by the onboard Realsense 455 camera (Fig. 2.3C); each semantic keypoint corresponds to a specific 3D point on the object CAD model. The detected 2D points are mapped to 3D keypoints based on the image's depth channel. The object's pose can be then estimated by aligning the detected keypoints to the corresponding points in the known object CAD model (Fig. 2.3B). As there may be outliers in the semantic keypoint detection process, we employ a robust registration algorithm, namely TEASER++ [38], to find the target pose. Combining the camera-relative object pose with the quadrotor's own odometry estimate gives a global pose estimate for the target object (Fig. 2.3C). In our implementation, we use the visual-inertial odometry computed by the T265 RealSense camera as the quadrotor's state estimate.

Fig. 2.3D-F show the pixel-space, translation, and rotation errors for the pose estimates produced by out perception pipeline for a med-kit on a test dataset held out from training. As the distance to the target increases, the translation estimate degrades while the keypoint and rotation

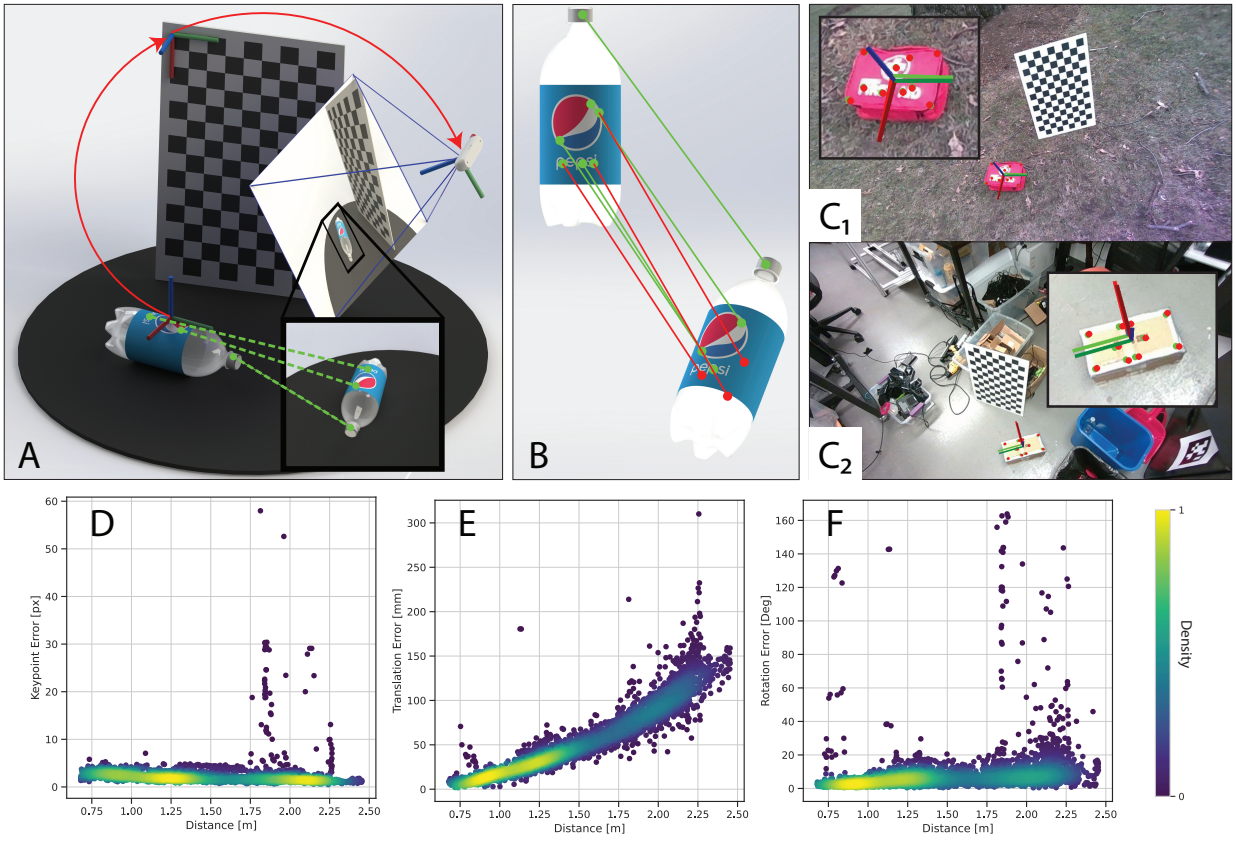


Figure 2.3: Keypoint-based object pose estimation. (**A**) Keypoints are labeled to generate training data for the semantic keypoint detector, for each object of interest. A checkerboard in the background of the collection area enables calculation of camera pose, which accelerates and partially automates the labeling process. A Resnet-18 neural network [37] is trained to predict object keypoints based on this dataset. (**B**) The 3D keypoints predicted by the perception system are matched against the corresponding keypoint locations on the object CAD model. A rigid transformation computed between the observed and CAD keypoints gives an estimate of the relative pose between the object and the camera. (**C**) The keypoint and pose detection system generalizes across environments and targets. The red points are predicted keypoints, the green points are ground-truth keypoints, the dark colored axes indicate the estimated pose, and the light colored axes denote the ground-truth pose. (**D-F**) Error characteristics for the proposed perception system, showing pixel-space, translation, and rotation error depending on distance between the target object and the camera for a med-kit object.

errors remain roughly constant, indicating that our range is mostly limited by the depth camera's accuracy. In our flight tests, we manage this error by estimating the target's pose when the drone is within 1.25m of the target. Additionally, these raw global pose estimates are filtered using a fixed-lag smoother to further improve accuracy; see Section 3 for details.

Our system is capable of estimating the full 3D rotation of the target, which is necessary to plan grasps for targets on inclines or for determining the feasibility of a grasp. However, for most grasping applications, the target can be assumed to rest on a flat horizontal plane. Thus, only the target's yaw is used to plan trajectories.

2.2 Results: Grasping Static Targets

We conducted grasps of three distinct stationary objects (a med-kit, a two-liter bottle, and a cardboard box, Fig. 2.4C-E) to demonstrate the versatility of the proposed soft gripper and vision system and quantify the effects of various sources of error. In our tests, upon takeoff, the drone flies to a randomized starting point from which it can see the target, as shown in Fig. 2.4A. Once the perception system has an estimate of the target's pose, it flies to a point such that the forward direction of the drone is aligned with a predefined grasp direction for the target. (Fig. 2.4A-1,B-1). From that starting point, the quadrotor flies a trajectory through the grasp point (Fig. 2.4A-2,B-2) that terminates at an endpoint away from the target (Fig. 2.4A-3,B-3). The drone stops updating its estimate of the target pose once it has started the grasp trajectory, relying on its initial estimate of target pose and updated estimate of its own state in the global frame in order to fly to the grasp point.

2.2.1 Grasp Success Rate

In this first set of tests, the reference trajectories were chosen to have a forward velocity of 0.5 m/s at the grasp point. A grasp is considered a success if the target remains grasped until the drone lands. For all tests, the target is secured to a horizontal surface with weak magnets to prevent the

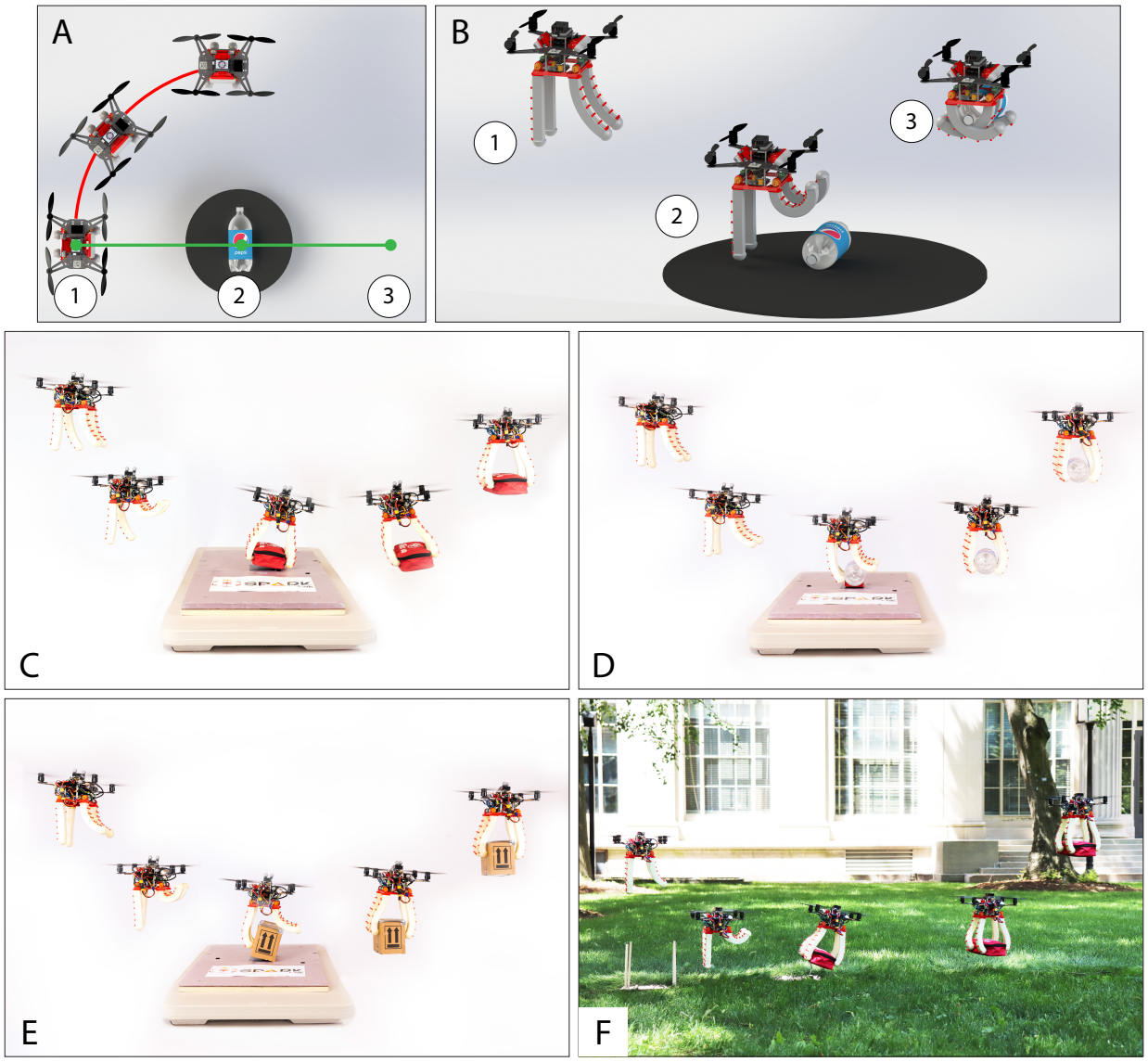


Figure 2.4: Overview and timelapses of experiments with static targets. (**A**) Experimental setup for the vision-based experiments. The red curve depicts the distribution of random start points, the green line the grasp trajectory; the numbers denote the grasp planning point ①, the grasp point ②, and the terminal point ③. (**B**) Configurations of the gripper states through the course of the grasp trajectory: ① target observation state, ② pre-grasp state, and ③ post-grasp state. (**C**-**E**) Static vision-based grasps at 0.5 m/s for med-kit, two-liter bottle, and cardboard box, respectively. (**F**) Outdoor vision-based grasped of the med-kit at 0.5 m/s.

downdraft of the quadrotor displacing the target pre-grasp. Under these conditions, the system achieves a success rate of 9/10, 6/10, and 10/10 for the med-kit, cardboard box, and two-liter bottle, respectively, and with fully onboard vision (Fig. 2.5A). We repeated these experiments with a motion capture system to determine the extent that our perception system affects grasp success. The motion capture baseline performs similarly to the vision-based approach (Fig. 2.5A), slightly under-performing for the med-kit, improving for the cardboard box, and performing equally for the bottle. Our results show that our vision system is able to consistently match the motion capture results at this speed, indicating that the gripper mechanism is robust against the additional error caused by the perception pipeline.

The biggest contributors to grasp performance difference among objects are their mass and surface morphology. The two-liter bottle's cylindrical geometry and low mass (60g) enable consistent enveloping grasps. The uneven, roughly concave edges of the med-kit provide good grasp surfaces, but the high mass (148g) makes sustaining the grasp more difficult. The cardboard box's tall, straight walls combined with its relatively high mass (115g) force the gripper to rely on large pinching force to secure the grasp, making it the most difficult target to grasp.

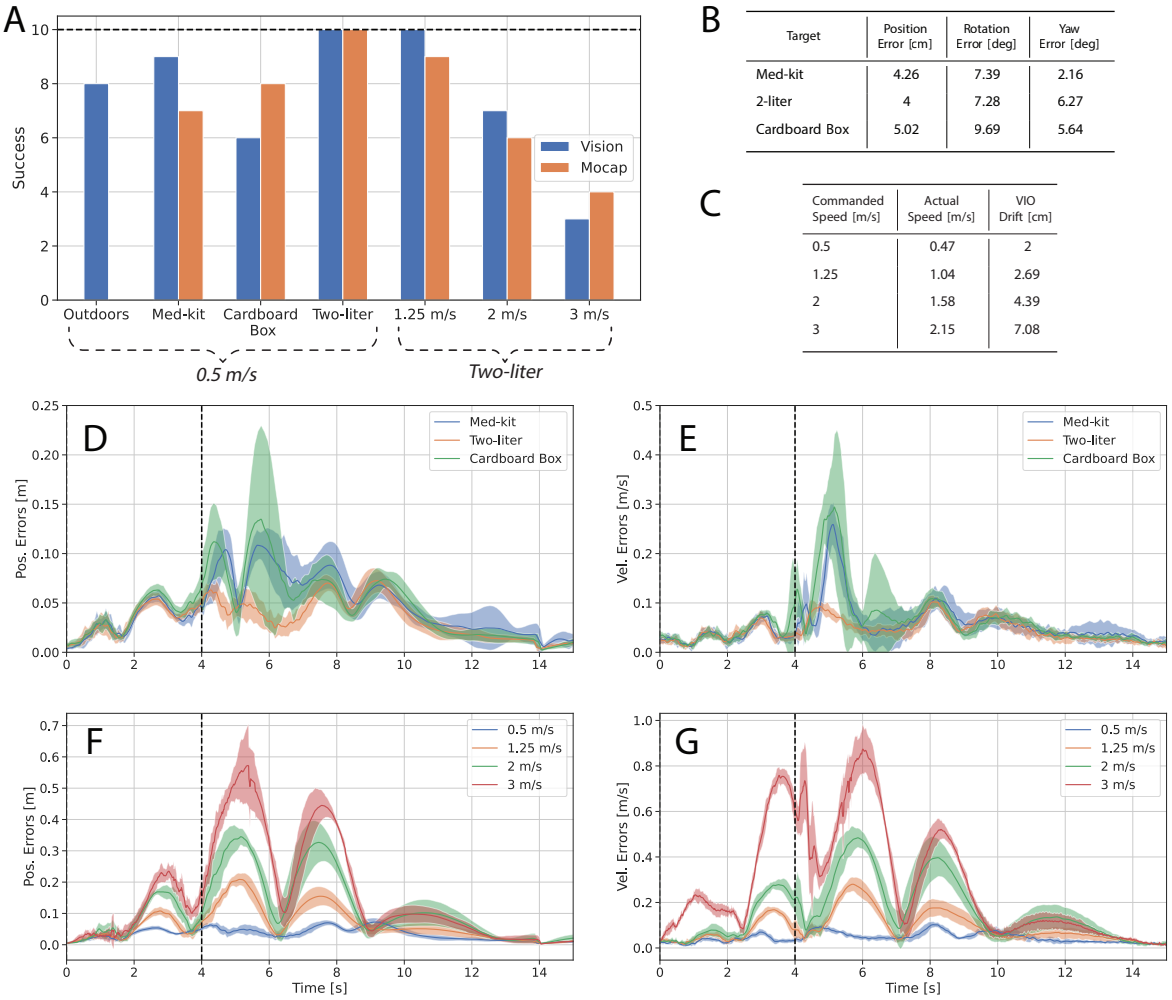
2.2.2 Error Analysis

We collected ground-truth pose data for the drone and target with a motion-capture system in order to understand the relative contribution of errors from the target estimation pipeline, the VIO estimate, and the trajectory tracking to the system's performance.

The filtered pose estimate for each target (Fig. 2.5B) at the start point of the trajectory has at most 5 cm translation error and less than 10 degrees of rotation error. Furthermore, only the target's yaw is needed for grasp planning and at most has roughly 6 degrees of error.

The average error between the quadrotor's actual position and VIO-based estimate (Fig. 2.5C) up until the grasp point is about 2 cm, for a forward speed of 0.5 m/s.

Finally, the trajectory tracking performance (Fig. 2.5D,E) before the grasp point is comparable for all three objects, with approximately 5 cm position error, and 0.05 m/s velocity error. One



B

Target	Position Error [cm]	Rotation Error [deg]	Yaw Error [deg]
Med-kit	4.26	7.39	2.16
2-liter	4	7.28	6.27
Cardboard Box	5.02	9.69	5.64

C

Commanded Speed [m/s]	Actual Speed [m/s]	VIO Drift [cm]
0.5	0.47	2
1.25	1.04	2.69
2	1.58	4.39
3	2.15	7.08

Figure 2.5: Static grasp experiments results. (**A**) Success rate for each experiment, comparing motion-capture-based performance (when available) with onboard-vision-based performance. 10 flights were conducted for each experiment, as indicated by the horizontal dashed line. (**B**) Filtered target pose estimation errors for each object at the time of trajectory planning, averaged across all runs. (**C**) Actual forward velocity as measured by motion capture and VIO drift for varying commanded speeds, averaged across runs. (**D-E**) Position and velocity tracking errors over the grasp trajectory for the three objects. (**F-G**) Position and velocity errors for four different speeds. The vertical dashed line denotes the time of grasp and the shaded regions represent one standard deviation from the mean. The errors here are computed as the mismatch between the vision-based estimate and desired setpoint.

cause of tracking error before the grasp time is the presence of the *ground effect*, where the drone experiences greater thrust efficiency when approaching a horizontal surface [39], thus raising the drone vertically above its desired setpoint (Fig. A.1). This is partially compensated by the adaptive controller, and the ability of the soft fingers to safely contact the ground reduces the risk from overcompensation. Apart from specific aerodynamic effects, our platform has a low thrust-to-weight ratio, which limits its control authority and invokes slight tracking errors even in normal flight conditions.

After the grasp, the higher mass of the med-kit and cardboard box lead to increased tracking errors primarily in the vertical direction (Fig. A.1), which reduce over time as the adaptive controller learns to compensate. It is important that both pre- and post-grasp errors are minimal for a successful grasp: high pre-grasp errors may cause the drone to be misaligned and miss the target, while high post-grasp errors could cause unintended collisions with the environment (*e.g.,* floor) and destabilize the drone.

Further breakdown of errors across separate axes is presented in Fig. A.1, Table A.1-Table A.3, and Appendix B, and information on how these error metrics are measured is provided in Section 3.3.4.

2.2.3 Grasp Speed Analysis

We further evaluate the performance of our system by increasing the desired forward grasp velocity to 1.25 m/s, 2 m/s, and 3 m/s for the two-liter bottle. We conducted ten flights for each speed with the full vision-based system, and repeated another ten flights for each speed in motion capture as a baseline. Our system is capable of grasping the bottle with full vision at over 2 m/s with a success rate of 3/10 (Fig. 2.5A). To our knowledge, this is the fastest fully vision-based aerial grasp reported in the literature.

The increases in speed are accompanied by larger tracking errors in both position and velocity (Fig. 2.5F, G). Because there exists a large discrepancy between the desired and actual speeds at grasp, we report the drone's average actual grasp speed as measured by the motion capture system

in (Fig. 2.5C). This error can be attributed to lower control authority at higher speeds; our platform operates near its maximum payload capacity and struggles to accelerate quickly. However, the error is mostly along the longitudinal direction (i.e., grasp or forward axis), with the lateral and vertical errors remaining low regardless of speed. Longitudinal tracking errors have little effect on our grasp success as the gripper control is informed by the current position of the drone, not its desired position. As long as the drone's state estimate remains accurate up until the time of grasp, the grasp will be triggered at the appropriate time. We validate this by analyzing the VIO drift for the varying speeds (Fig. 2.5C). VIO drift does increase with grasp speed, remaining manageable for desired speeds less than 2 m/s, but reaching a substantial 7 cm for a desired speed of 3 m/s. As reflected in Fig. 2.5A, this increase in VIO error proves to have a non-negligible effect on grasp performance. Table A.1-Table A.3 further break down VIO error per axis for each speed.

2.2.4 Outdoor Grasps

A central motivation for our onboard perception system is the ability to operate in places where a localization infrastructure is unavailable. To emphasize this capability, we demonstrate the quadrotor picking up the med-kit in an outdoor field (Fig. 2.4F) at 0.5 m/s. Grasping objects in outdoor environments presents several additional challenges, including unpredictable wind disturbances, varying illumination conditions, and visually diverse surroundings. Furthermore, the sustained ground plane introduces an additional challenge compared to the indoor flights where the target was placed on a raised pedestal. Post-grasp, the delay in adapting to the target's mass causes the drone to drag across the ground temporarily which increases the chance of failure. To combat this, we introduce a feedforward acceleration impulse that counteracts the immediate vertical disturbance post grasp and briefly raises the drone up from the ground plane. Afterwards, the drone continues tracking the nominal trajectory. We also note that the starting point of the drone was left fixed, rather than randomized as in the indoor experiments. We demonstrate a success rate of 8/10, and, to our knowledge, the first instance of aggressive manipulation in an outdoor environment.

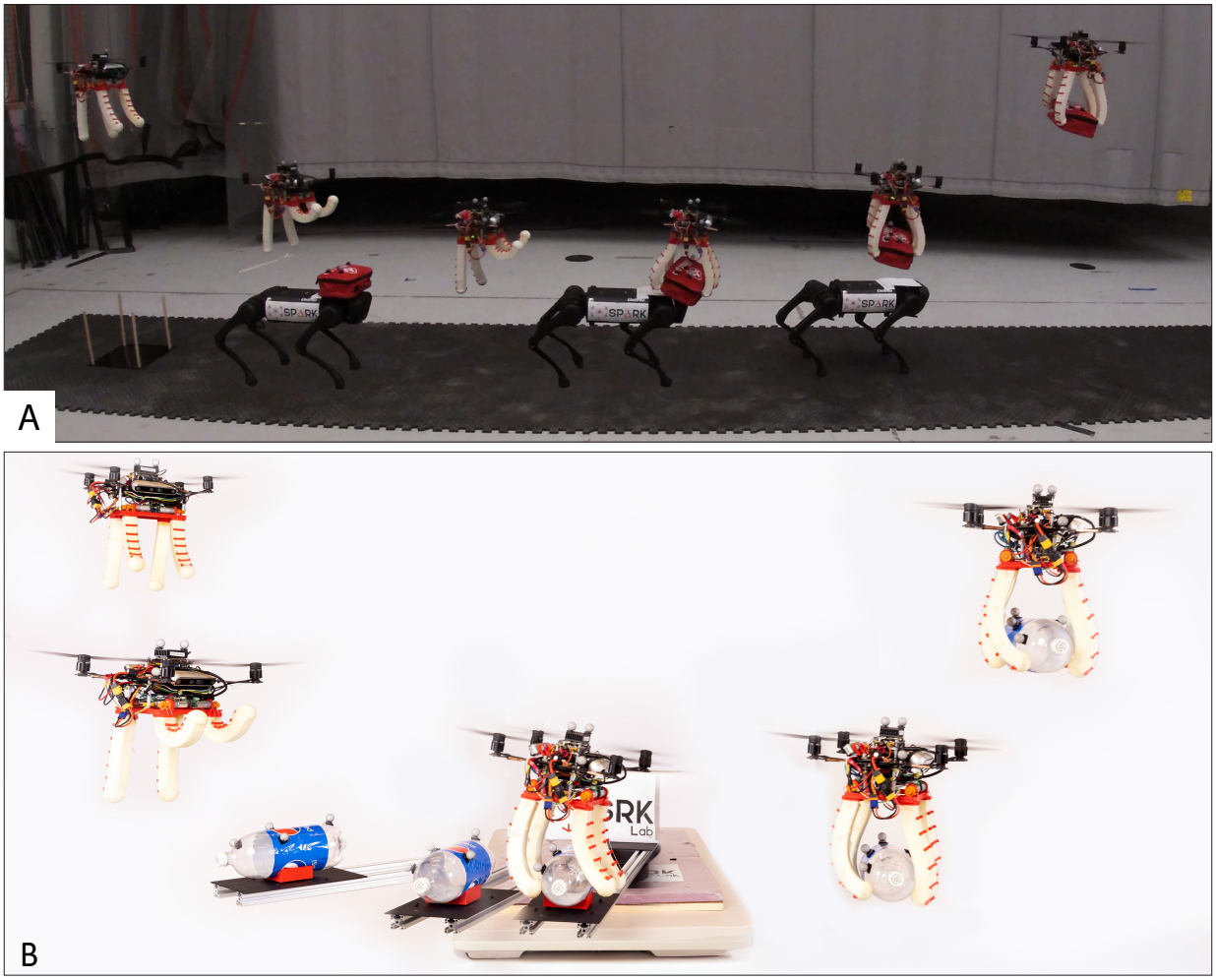


Figure 2.6: Timelapses of experiments with moving targets. (**A**) Moving grasps of med-kit attached to a quadruped robot that has forward velocity of 0.3 m/s and with a relative grasp speed of 0.1 m/s. (**B**) Moving grasps of two-liter bottle on a turntable moving with a tangential speed of 0.08 m/s and with a relative grasp speed of 0.5 m/s; see Movie 1 for more visualizations.

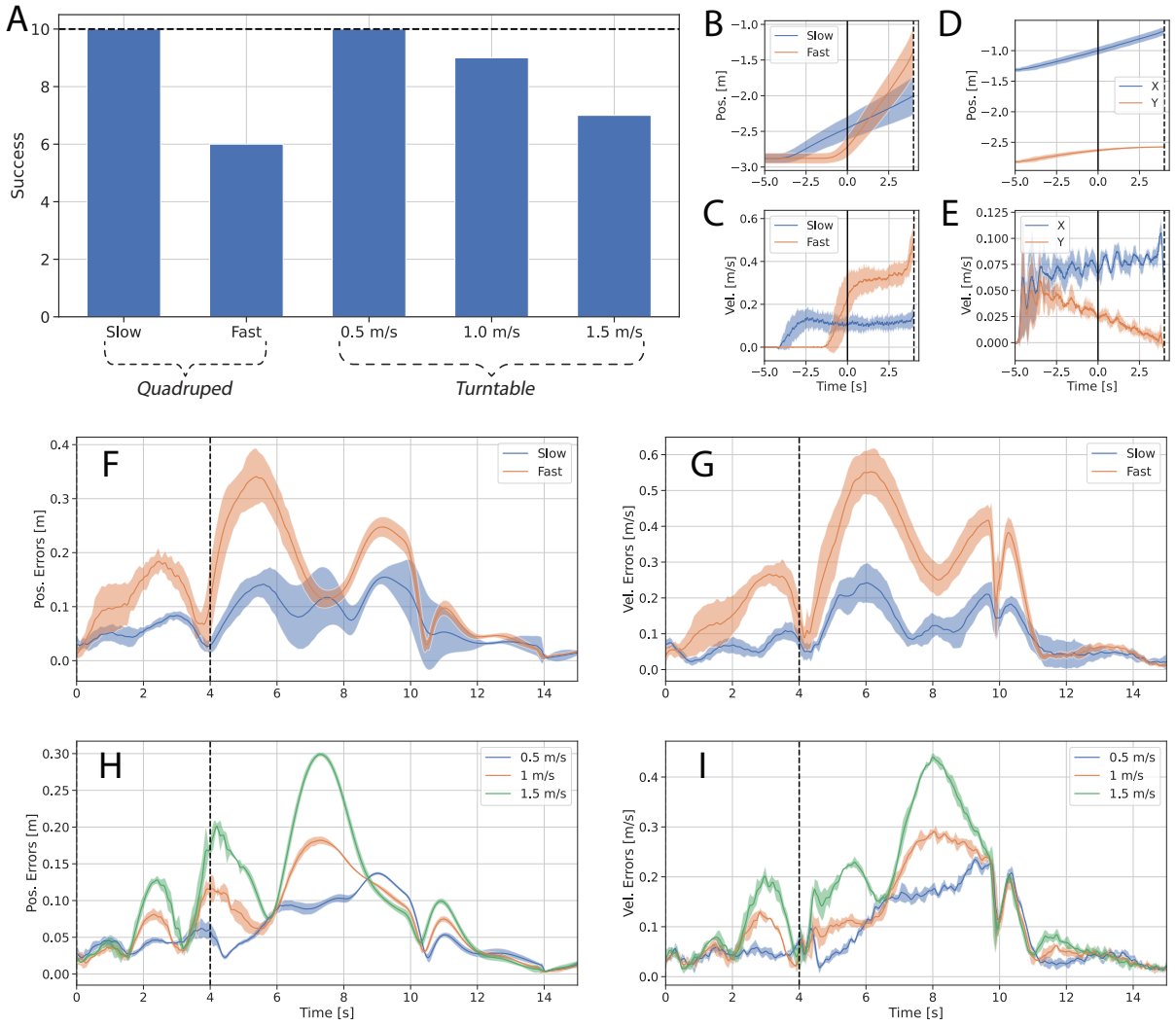


Figure 2.7: Moving grasp experiments results. (**A**) Success rates for each experiment, including different moving platforms (Quadruped and Turntable) and increasing speeds. (**B-C**) Position and velocity of the target for the quadruped experiments. (**D-E**) Position and velocity of the target for the turntable experiments. Here, X, Y represent the axes in the inertial floor plane. (**F-G**) Position and velocity tracking errors for the quadruped experiments. (**H-I**) Position and velocity tracking errors for the turntable experiments. For all plots, the solid black line denotes the start of the grasp trajectory, and the dashed black line is the time of grasp (after which the target pose is unavailable).

2.3 Results: Grasping Moving Targets

In this section we investigate grasping *moving* targets at high velocities, or with high relative velocity between the drone and the target. These maneuvers are common in biological systems, where birds of prey must quickly capture their prey before it can escape. In robotics applications, this capability can increase efficiency by removing the need to decelerate before grasping as well as minimize the time the drone must spend near the ground in dangerous environments.

Grasping moving targets presents additional challenges on top of the ones observed with static targets. When an object has rotational velocity, the desired grasp direction also rotates. This requires more lateral tracking accuracy than is needed for the static cases. Dynamic objects also have uncertainty in their motion model, which we observe when trying to grasp an object from a moving quadruped. While our system cannot simultaneously handle errors from perception and dynamics, we demonstrate the ability to aggressively grasp moving objects with the aid of motion capture for drone and target localization.

Our first set of experiments with moving targets consists of the med-kit mounted on top of a Unitree A1 quadruped robot (Fig. 2.6A). A human operator steers the quadruped in a roughly linear motion (Fig. 2.7B). Even though the trajectory is nominally linear and constant velocity, there is noticeable variation. We conducted 10 flights at a slow quadruped forward velocity (roughly 0.15 m/s) and a fast forward velocity (roughly 0.3 m/s) (Fig. 2.7C). The drone was commanded to grasp the med-kit with a relative forward velocity of 0.1 m/s. The drone achieved 10/10 success for the slow speed, and 6/10 for the fast (Fig. 2.7A). This difference can primarily be attributed to the increased tracking errors seen at grasp point for the higher speed (Fig. 2.7F,G), with noticeable increases in lateral and vertical errors (Fig. A.2, Table A.2, Table A.3).

For our second set of dynamic experiments, the two-liter bottle sits on a lever arm attached to a motorized turntable (Fig. 2.6B). With a fixed rotation speed of the turntable, we perform 10 tests for each desired relative grasp speeds of 0.5 m/s, 1.0 m/s, and 1.5 m/s. For all experiments, the two-liter bottle follows a circular arc (Fig. 2.7D) with a tangential speed of approximately 0.08 m/s

(Fig. 2.7E). Circular motion is challenging for our approach; because the drone tracks the target at a lever arm, the angular velocity of the bottle gets amplified into large translational velocities at the drone's setpoint. Despite this, the drone exhibited high success rates for the three speeds (Fig. 2.7A). Similar to the the static speed analysis, the tracking errors increase with relative speed (Fig. 2.7H, I). However, these errors remain mostly concentrated in the longitudinal axis (Fig. A.2, Table A.2, Table A.3), which has little effect on the grasp success. The rotational motion does induce higher lateral errors with speed (Table A.2), which causes a misalignment that contributes to failed grasps at high speeds.

Chapter 3

Hardware Design and Software Methods

3.1 Hardware Design

3.1.1 Drone Design

Our drone platform (Fig. 1.1C) is a quadrotor inspired by the *Intel Aero Ready to Fly* (RTF) drone. It has a custom waterjet carbon fiber frame, but uses the same Yuneec brushless DC motors and propellers as the RTF drone. A Pixhawk 4 Mini flight controller running custom PX4 firmware handles the low-level adaptive controller. The drone is powered by a four-cell Lithium Polymer battery, which provides about three minutes of flight time (enough for three grasp trajectories).

The drone has a front-facing Realsense D455 RGB-D camera pitched down at a 35 degree angle for target pose estimation, and a rear-facing Realsense T265 for VIO estimation. We observed the T265's odometry estimation failing under high-vibration conditions, so we added vibration damping silicone grommets between the camera and drone. The camera required physical masking on the bottom of the camera lens to match the field of view constraint used in the gripper's tendon length optimization, as the camera does not support software-defined region masking. Ensuring that the fingers were not in the camera frame was vital, as the visual odometry estimate when the fingers entered the frame proved to be quite poor.

The visual target estimation pipeline and trajectory planner are run on a Jetson Xavier NX. Both

Realsense cameras are connected to the Xavier via USB 3.0 and a powered USB hub. The Xavier communicates with the Pixhawk over UART. When using motion-capture pose estimates instead of visual navigation, the Xavier receives pose updates over Wifi from a base station connected to the motion capture system. See Table A.4 for detailed component specifications.

3.1.2 Gripper Design

The drone interfaces with the gripper through standoffs connected to a 3D printed base plate, which houses the gripper components (Fig. 1.1D). A custom-made, lightweight PCB embeds an ATMega32U4 microcontroller to handle logic control, 2 DRV8434 motor controllers which actuate four 31:1 gear ratio 12V DC motors, and a voltage regulator and connector to convert the Lipo battery's voltage into a stable 12V. Jumper cables are wired from GPIO pins on the Xavier to the PCB to connect the gripper state machine with the low level gripper controller.

The gripper features four passively closing, cable actuated fingers (Fig. 2.2B). Each finger is molded using Smooth-On *FlexFoam-iT! X* [40] and a custom 3D-printed mold, which follows a circular arc (Fig. A.4). The initial state of the finger is then its closed position, which is maintained by the elasticity of the mesh. A 3D-printed finger-base connector is super glued to the top of the finger and can be slotted into the base plate for quick replacement. The tendon guides are positioned between evenly placed foam extrusions and adhered with super glue. Braided fishing line is fixed to the last tendon guide and the motorized winch, passing through the interior tendon guides. As the winch tightens, the tendon contracts, compressing the outer edge of the finger and forcing it open. As the winch loosens, the elastic force of the mesh returns the finger to its default closed position.

3.2 Target Object Perception and Pose Estimation

We present a robust pose estimation system that combines a keypoint detector with a state-of-the-art algorithm for kepoint-based 3D pose estimation, namely Teaser++ [38]. Estimated poses are

further refined through a fixed-lag smoother (Fig. 2.1). We also developed a semi-automated data annotation tool to rapidly train the keypoint detector.

3.2.1 Semi-Automated Keypoint Annotation

Keypoint annotation can be a costly process, requiring multiple points to be labeled in each image of a large image dataset (*e.g.,* with tens of thousands of images). We draw inspiration from prior methods of speeding up keypoint labeling by using known camera poses at training time [41], [42], enabling a small number of manually labeled keypoints to be projected into many images. Fig. 2.3A presents an overview of the annotation pipeline. The target object and calibration board are positioned such that both are visible in the recorded camera frames, and they are fixed for the entirety of the recording process. We move the D455 camera at varying distances and angles from the object and record all RGB and depth images. The pose of the camera with respect to the calibration board for each image can be readily determined by computing the camera extrinsics using the calibration board. Known camera poses enable reasoning about the 3D position of each keypoint across the whole trajectory sequence. First, pixel locations for the keypoints in user-selected images are manually annotated using the Matlab tool GUI (Fig. A.5). The 3D position for each keypoint relative to the camera is determined by back-projecting the pixel coordinates using the depth of each keypoint, given by the D455. The keypoint positions can be further refined by minimizing the reprojection error in a small number of manually labeled training images, and then reprojected into all other images based on the camera model.

In our tests, we only annotate keypoints on the top face of the object and assume that no keypoints are occluded in our training set; this is a valid assumption considering the viewing angle of our drone. Proper annotation of keypoint visibility is important to prevent the neural net from associating features blocking the view of the keypoint with the true keypoint features. The labeling method does support keypoints placed on all sides of the object, but self-occlusions require visibility detection. This can be done by sampling the depth of each keypoint using the depth image and comparing that against the expected depth obtained by transforming the keypoint

relative to the camera. If there is a large discrepancy, the keypoint is assumed to be occluded. This method requires a highly accurate depth sensor and the D455 proved to be inadequate.

We add variety in the training set to improve generalization of the keypoint detector by collecting data in several environments: a motion capture room, a cluttered workshop, a lounge space, a foyer, and an outdoor field (Fig. A.6). For each location, the data collection process was repeated multiple times with different poses of the target object. A randomly selected subset of the runs were added to the validation set. Across the three targets, 39548 training images and 15172 validation images were collected and annotated, within only a few days of manual labor.

3.2.2 Keypoint Detection

To minimize training time and increase generalizability, we apply transfer learning on a state-of-the-art keypoint detection architecture, *trt-pose* [37]. The network is comprised of a ResNet-18 backbone, followed by a keypoint prediction head. The training data is augmented using standard techniques (perturbations of translation, rotation, scale, and contrast) to further improve diversity of the training set. We present the full set of parameters we used for trt-pose's training pipeline in Table A.5. Training takes around seven hours on an Nvidia GeForce GTX 1080 Ti GPU, with the loss converging after 75 epochs.

The original RGB images have size 1280×720 and are cropped by removing the top half and resized into 912×256 images to improve inference speed. Due to the geometry of the camera mount, cropping the top half of the images does not reduce performance as objects in that region are too far away to detect reliably anyway. The trained network is compiled to leverage the Jetson Xavier's hardware using TensorRT, and runs at approximately 14 Hz on the Xavier.

Fig. 2.3D shows the results of the keypoint detector on the med-kit test set, along with visual representations in Fig. 2.3C. Keypoint error is defined as the pixel distance between the estimated and ground-truth keypoints, averaged across all keypoints, and only for keypoints that were valid detections (an invalid detection occurs when the keypoint isn't detected and the network predicts $(0, 0)$). The error remains roughly constant with distance, but we note that pixel error is influenced

by the scale of the target. At far distances, the target's perceived area is smaller and a unit of pixel error translates to higher error in world coordinates than a unit of pixel error at close distances.

3.2.3 Robust Point Cloud Registration

From the depth image provided by the D455 camera and the detected keypoint pixel coordinates, we create an estimated point cloud of keypoints, denoted as $\mathcal{A} = \{\boldsymbol{a}_i\}_{i=1}^{N}$. Furthermore, the corresponding keypoints are also annotated on the known object CAD model through the data annotation tool, leading to a second point cloud, namely $\mathcal{B} = \{\boldsymbol{b}_i\}_{i=1}^{N}$. Solving for the transformation between these two point clouds allows us to estimate the pose of the target relative to the camera. We seek to find the rotation $\boldsymbol{R} \in SO(3)$ and translation $\boldsymbol{t} \in \mathbb{R}^3$ which minimize the following *truncated nonlinear least squares* problem [38]:

$$\min_{\boldsymbol{R}\in SO(3),\boldsymbol{t}\in\mathbb{R}^3} \sum_{i=1}^{N} \min\left(||\boldsymbol{b}_i - \boldsymbol{R}\boldsymbol{a}_i - \boldsymbol{t}||^2 \ , \ \bar{c}^2\right), \tag{3.1}$$

where $\bar{c}$ represents the upper bound on the residual error for a measurement to be considered an inlier, and is set to be 10 mm. In the truncated least squares problem in Eq. 3.1, measurements with residual error smaller than $\bar{c}$ are used to compute a least-squares estimate, while residuals greater than $\bar{c}$ have no effect on the estimate and are discarded as outliers. In our application, this is useful for filtering out outliers caused by spurious keypoint or depth detections. In practice, we utilize Teaser++ [38] to solve this problem, which takes on average 2 ms to solve a single registration problem on the Xavier NX. Fig. 2.3E-F demonstrate the results of our approach for the med-kit object. Translation error is defined as the Euclidean distance between the estimated and ground-truth translations, while rotation error is computed as the geodesic distance between the estimated and ground-truth rotations. Additionally, we only report data for detections that had at least one inlier point. We nominally plan trajectories at a distance of 1.25 m from the target, which equates to roughly 5 cm of translation error and 15 degrees of rotation error. Our experiments have validated that this error is well tolerated when combined with the fixed-lag smoother and the

robustness of the soft gripper.

3.2.4 Fixed-Lag Smoother

Estimates of the target's relative pose from individual images are fused together in a fixed-lag smoother to reduce noise in the estimate and provide estimates for linear and angular velocity. Knowledge of the target's velocity is necessary for grasping moving targets at the desired relative speed. We rely on the estimation process described below to infer velocity.

We model the target's motion as a nearly constant velocity model [43], with the target's pose $\boldsymbol{T}_k \in SE(3)$ and linear and angular velocity $\boldsymbol{\xi}_k \in \mathbb{R}^6$ at time k:

$$\begin{aligned} \boldsymbol{T}_{k+1} &= \boldsymbol{T}_k \boxplus (\boldsymbol{\xi}_k dt) \\ \boldsymbol{\xi}_{k+1} &= \boldsymbol{\xi}_k + \boldsymbol{\nu}_k dt, \end{aligned} \tag{3.2}$$

where $\boldsymbol{\nu}_k \in \mathbb{R}^6 \sim \mathcal{N}(\mathbf{0}, \Sigma)$ represents white-noise angular and linear acceleration, and $\boxplus$ maps a vector in $\mathbb{R}^6$ (on the right-hand-side of the operator) to $SE(3)$ (e.g., via an exponential map or, more generally, a *retraction* [44]) and then composes it with the pose on the left-hand-side of the operator. Given observations of the target's pose over the last τ timesteps, namely $\hat{\boldsymbol{T}}_{T-\tau:T}$, we estimate $\boldsymbol{T}_{T-\tau:T}$ and $\boldsymbol{\xi}_{T-\tau:T}$ as

$$\begin{aligned} \min_{\boldsymbol{T}_{T-\tau:t_T}, \boldsymbol{\xi}_{T-\tau:T}} \sum_{k=T-\tau}^{T} & ||\boldsymbol{T}_k \boxminus \hat{\boldsymbol{T}}_k||^2_{\Omega_1} \\ & + ||(\boldsymbol{T}_k \boxplus (\boldsymbol{\xi}_k dt)) \boxminus \boldsymbol{T}_{k+1}||^2_{\Omega_2} \\ & + ||\boldsymbol{\xi}_k - \boldsymbol{\xi}_{k+1}||^2_{\Omega_3} + ||\boldsymbol{\xi}_k||^2_{\Omega_4}, \end{aligned} \tag{3.3}$$

where $\boxminus$ denotes a tangent space representation of the relative pose between two poses. The first term encourages the estimate to match the target pose observations, the second term constrains the estimated velocity to be consistent with the poses, and the third term forces changes in velocity between timesteps to be small. The final term provides regularization on the velocity estimate.

We found a small amount of velocity regularization to be necessary for consistent initialization of the filter. $\Omega_1,\ \Omega_2,\ \Omega_3,\ \Omega_4$ are tunable weight matrices to trade off among the four terms (see Table A.6). The optimization is implemented in C++ using the GTSAM library [45]. The incremental iSAM2 algorithm [46] is used to optimize (3.3), as it enables reuse of computation between timesteps.

3.3 Trajectory Optimization and Control

To enable our grasps, we compute a smooth polynomial trajectory for the drone, passing at a set distance over the target (the *grasp point*). This nominal trajectory is tracked with an adaptive controller which compensates for the added mass after grasping and for other external disturbances. Throughout the grasp procedure, the drone's current position is fed into a gripper state machine, which sets the finger winch angle based on the results of an offline FEM-based optimization (Fig. 2.1). The trajectory optimization and control are based on prior implementations for motion-capture based grasping [23], with added functionality enabling trajectory planning during flight and updating of the reference trajectory to account for moving targets.

3.3.1 Minimum-Snap Polynomial Trajectory Optimization

We compute a fixed-time polynomial aligned with the target's grasp axis and constrained to start at the drone's initial position, pass through the grasp point, and end at an arbitrary final position. The grasp point is defined as the target's position, plus a tunable offset to account for the length of the fingers and any misalignment between the target frame center and the geometric center. We seek to find a polynomial that minimizes the 4th derivative of position, or snap, of the drone. As shown in [33], polynomials of this form enable smooth, continuous motion which allows for stable, consistent grasps. The key idea which enables grasping moving targets is that the polynomial is planned with respect to the *target's* frame, rather than the inertial frame. As the target moves the polynomial moves with it, inducing a disturbance which is compensated for by our low-level

tracking controller. The polynomial provides a series of position, velocity, and acceleration setpoints that are tracked by the drone's flight controller. The drone's yaw setpoint is set to always point toward the estimated object position. Appendix C provides further exposition on our specific trajectory optimization implementation.

3.3.2 Adaptive Quadrotor Control

The drone is subject to various forms of disturbances that could impede both grasp success and post-grasp flight performance. After the target is grasped, the effective mass of the quadrotor increases and the thrust efficiency decreases, as the grasped object partially obstructs the propellers. Estimates of system model parameters are also imperfect, and benefit from online adaptation. Additionally, we observe that our system is also sensitive to variable disturbances such as wind in outdoor environments or the ground effect. To mitigate these disturbances, we adopt the geometric adaptive control law presented in [34], that we briefly review below.

The external disturbances in our system primarily affect the translational dynamics of the drone, with negligible impact on the rotational dynamics. Hence, we model the quadrotor dynamics as:

$$\begin{aligned} m\ddot{\boldsymbol{p}} &= m\boldsymbol{g} + f\boldsymbol{b}_z + \boldsymbol{\theta}_f \\ \dot{\boldsymbol{R}} &= \boldsymbol{R}\hat{\boldsymbol{\Omega}} \\ \boldsymbol{J}\dot{\boldsymbol{\Omega}} &= -\boldsymbol{\Omega} \times \boldsymbol{J}\boldsymbol{\Omega} + \boldsymbol{\tau} \end{aligned} \tag{3.4}$$

where m is the drone's mass, $\boldsymbol{J}$ is the moment of inertia, $\boldsymbol{g}$ is the gravity vector, $\boldsymbol{p} \in \mathbb{R}^3$, $\dot{\boldsymbol{p}} \in \mathbb{R}^3$, $\boldsymbol{R} \in \mathrm{SO}(3)$, $\boldsymbol{\Omega} \in \mathbb{R}^3$ are the position, velocity, rotation, and angular velocity of the drone, respectively, f is the scalar thrust force applied along the local vertical direction $\boldsymbol{b}_z$, $\boldsymbol{\tau}$ is the torque applied by the quadrotor, and $\boldsymbol{\theta}_f$ is the unknown translational disturbance. The *hat* operator $\hat{\cdot}$ maps a vector in $\mathbb{R}^3$ to an element of the Lie algebra $\mathfrak{so}(3)$, i.e., a 3×3 skew symmetric matrix; see [34].

We then compute the quadrotor's thrust and moment as:

$$f = -\boldsymbol{b}_z^{\mathsf{T}}(k_p\boldsymbol{e}_p + k_v\boldsymbol{e}_v + m\boldsymbol{g} - m\ddot{\boldsymbol{p}}_d + \bar{\boldsymbol{\theta}}_f) \tag{3.5}$$

$$\begin{aligned}\boldsymbol{\tau} &= -k_r\boldsymbol{e}_r - k_\Omega\boldsymbol{e}_\Omega + \boldsymbol{J}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}_d\dot{\boldsymbol{\Omega}}_d \\ &\quad +(\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}_d\boldsymbol{\Omega}_d)^\wedge\boldsymbol{J}\boldsymbol{R}^{\mathsf{T}}\boldsymbol{R}_d\boldsymbol{\Omega}_d\end{aligned} \tag{3.6}$$

where $\boldsymbol{p}_d$, $\boldsymbol{R}_d$, $\boldsymbol{\Omega}_d$ are the desired position, attitude, and angular velocity, respectively, $\boldsymbol{e}_p$, $\boldsymbol{e}_v$, $\boldsymbol{e}_r$, $\boldsymbol{e}_\Omega$ are the position, velocity, rotation, and angular velocity errors, $\bar{\boldsymbol{\theta}}_f$ is the estimated disturbance on the translation dynamics, and k_p, k_v, k_r, k_Ω are user-specified control gains. We refer the reader to [34] for the definition of the errors and a more comprehensive discussion.

The estimated disturbance is adjusted online using the following law:

$$\frac{d\bar{\boldsymbol{\theta}}_f}{dt} = \Pi\left(\gamma_f(\boldsymbol{e}_v + k_{af}\boldsymbol{e}_p)\right) \tag{3.7}$$

where γ_f, k_{af} are user-specified gains and Π is a suitable projection function.

This formulation ensures that the tracking errors asymptotically go to zero even in the presence of unknown disturbances [34].

3.3.3 FEM-Based Gripper Control

To optimally actuate the gripper, the controller must deform its fingers such that they effectively enclose the target object, while simultaneously ensuring that they do not obstruct the view of the onboard cameras. To balance these two goals, we employ an optimization-based control method. At the core of our control method is a finite element-based (FEM) model.

The gripper's volume is discretized into a finite element mesh with nodal positions $\boldsymbol{x}$, and simulated cables are routed through this mesh. Given cable contractions $\boldsymbol{u}$, we can solve for a

corresponding statically stable pose

$$\boldsymbol{x}(\boldsymbol{u}) = \arg\min_{\boldsymbol{x}} E(\boldsymbol{u}, \boldsymbol{x}) \tag{3.8}$$

by minimizing the total potential energy of the system E using Newton's method. This general approach can be elegantly extended to account for dynamics, and is done in [29], [47].

Our control optimization is a *nested optimization*. Its objective $\mathcal{O}(\boldsymbol{u}, \boldsymbol{x}(\boldsymbol{u}))$ is written in terms of the physically valid pose $\boldsymbol{x}(\boldsymbol{u})$, found by solving Eq. 3.8. We minimize this objective using gradient descent to find optimal control inputs for a desired configuration

$$\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} \mathcal{O}(\boldsymbol{u}, \boldsymbol{x}(\boldsymbol{u})). \tag{3.9}$$

Computing the gradient of our control objective requires the Jacobian $\frac{d\boldsymbol{x}}{d\boldsymbol{u}}$ relating changes in control inputs $\boldsymbol{u}$ to changes in the corresponding physically-valid mesh shape $\boldsymbol{x}(\boldsymbol{u})$, which we solve for using direct sensitivity analysis, as described in [48].

Our specific control objective $\mathcal{O}$ is the weighted sum of two sub-objectives, which are illustrated in Fig. 2.2D-E:

$$\mathcal{O} = \mathcal{O}_{\text{grasp}} + \mathcal{O}_{\text{FOV}} \tag{3.10}$$

The first sub-objective, $\mathcal{O}_{\text{grasp}}$, drives the gripper's fingers towards a configuration where they can effectively grasp an object. The sub-objective

$$\mathcal{O}_{\text{grasp}} = ||(\boldsymbol{s} - \boldsymbol{y}_1(\boldsymbol{u})) \times (\boldsymbol{s} - \boldsymbol{y}_2(\boldsymbol{u}))||^2 \tag{3.11}$$

works by maximizing the area enclosed by the gripper's fingertips and a *target point* $\boldsymbol{s}$ (Fig. 2.2E, [23]). Here, $\boldsymbol{s}$ represents the centroid of the target shifted from the drone by a nominal offset which approximates the relative position between the drone and target at the moment the gripper begins to close. The fingertip positions $\boldsymbol{y}_1(\boldsymbol{u})$ and $\boldsymbol{y}_2(\boldsymbol{u})$ are defined in terms of the physically-valid mesh position $\boldsymbol{x}(\boldsymbol{u})$ using barycentric coordinates.

The second sub-objective, $\mathcal{O}_{\text{FOV}}$, ensures the the gripper's fingers do not obstruct the camera's field of view. This is accomplished by defining a plane with point $\boldsymbol{p}$ and normal $\boldsymbol{n}$, analogous to the bottom face of a camera's view frustum (Fig. 2.2D). To prevent the finger from entering into the camera's field of view, the sub-objective

$$\mathcal{O}_{\text{FOV}} = \sum_i f\left((\boldsymbol{p} - \boldsymbol{x}_i(\boldsymbol{u})) \cdot \boldsymbol{n}\right) \tag{3.12}$$

sums over the squared penetration depths of all nodes in the mesh. Here, the function

$$f(z) = \begin{cases} z^2 - \varepsilon z + \frac{\varepsilon^2}{3}, & \text{if } \varepsilon < z, \\ \frac{z^3}{3\varepsilon}, & \text{if } 0 < z \leq \varepsilon, \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

is a smooth one-sided quadratic, driving the mesh outside the camera's field of view while ensuring the optimizer is well-behaved. Depending on the grasp phase, $\mathcal{O}_{\text{FOV}}$ is added for both the front and the rear camera (to obtain the *target observation state*) or only for the rear camera (to obtain the *pre-grasp state*).

3.3.4 Measuring Sources of Error

The success of our vision system depends on three sources of error: target estimation error, VIO drift, and trajectory tracking error. We collect motion capture data alongside the vision flights to benchmark these errors, although the quadrotor only uses vision-based data for its own control.

To quantify the amount of target estimation error (Fig. 2.5B), we placed motion capture markers near the target, but far from the key visual features. The target's motion capture frame was manually adjusted until it visually appeared aligned with the expected target's vision frame. Estimation error is then reported as the difference between our vision based estimate and the motion capture ground truth. VIO drift (Fig. 2.5C) was measured by aligning the grasp trajectory as re-

ported by the vision system with the recorded motion capture data. This alignment was done using *evo* [49], which performs a least squares optimization to find the optimal rigid transformation between trajectories. Here, we are only concerned with aligning part of the trajectory up until the grasp point, as post-grasp drift generally has litte effect on grasp success. Therefore, we align trajectories starting from 9 seconds before the grasp time up until the grasp time. We found this range to be suitable to be sure that the resulting alignment errors were due to trajectory drift and not because of a lack of data points. Trajectory tracking errors (Fig. 2.5D-G) were analyzed by extracting the setpoint and pose estimate from the flight controller.

When generating the plots and tables for these metrics, we exclude runs that resulted in catastrophic failures (i.e., the drone crashed after grasp). In total, we discarded two runs: one vision-based run with the cardboard box and one vision-based run with the pepsi bottle at 3 m/s desired speed. These runs were counted as failures. Additionally, for each of the turntable experiments, 5 trials were conducted with the turntable spinning clockwise and 5 were conducted counterclockwise. Because there is large variation between these cases, the error plots only use the clockwise case. The success rates use both cases.

Chapter 4

Conclusion

We advance the state of the art towards deployable, aggressive, and versatile aerial manipulation. We accomplish this by combining the desirable properties of a soft gripper with an advanced quadrotor platform and a state-of-the-art perception system. The gripper's reliable performance —even with imperfect grasp placement— and its ability to decouple the quadrotor dynamics from contact constraints enable high speed grasps. The system's ability to operate outside of a controlled motion-capture room is vital in nearly every practical application, enabling aerial vehicles to perform manipulation tasks such as emergency supply distribution, package pickup and delivery, and warehouse automation.

The compliant fingers of our soft gripper are key to enabling the full system to grasp at high speeds. As the drone flies faster, its position estimate suffers and tracking errors increase. Furthermore, the detections of the target's pose have several centimeters of error and VIO drift adds additional error. In spite of these error sources, the system successfully picks up objects at over 2 m/s, with full onboard vision.

The gripper mechanism is mechanically simple, which we have found to be a great advantage in manufacturability and repairability. Each finger assembly requires only a handful of 3D-printed parts, fishing line, off-the-shelf foam, and a small motor. The fingers themselves have been surprisingly resilient when attached to the drone. Their most common failure modes are tears that

develop near the base, which we have been able to easily repair with common super glue. They also provide cushion to the drone during hard landings; during our development we have seen the drone fall over six feet onto solid surfaces and not require any repair. Even in these cases, the fingers did not break and were used successfully for subsequent grasps.

Controlling soft fingers can be challenging, as they are difficult to model. In comparison to rigid manipulators, which have a well-defined finite dimensional state space, soft fingers are a continuum and have an infinite-dimensional state space. This makes it difficult to relate control input to finger state, which is necessary for our gripper to open its fingers for object grasping while staying out of the cameras' field of view when necessary. We have demonstrated that FEM-based modeling and control enable us to accurately approximate the continuum mechanics of our soft fingers, enabling reliable grasp performance. Our approach is readily generalizable to different, task-specific gripper configurations, which could be designed to exploit the morphology of a specific target, following the geometric analysis in Appendix B.

We see several potential avenues for future work. One limitation of our pose estimation pipeline is its assumption of prior knowledge of the target object's geometry and visual features. Leveraging recent advances in category-level keypoint detection, e.g., [50], would enable the system's perception pipeline to generalize to other instances of the same semantic category (e.g., multiple types of packages or soda bottles). Our drone is also currently limited to picking up light objects. We expect that scaling up the drone platform relative to the gripper would lead to a wider range of graspable objects without having to change the rest of the system, as a larger drone would have more control authority and more reliable trajectory tracking both before and after grasp. Finally, full vision-based grasps of moving targets is an immediate next step for our system. Grasping a moving target using only vision-based inputs with our current system requires predicting the object's motion when it goes out of frame before the grasp point, inevitably adding error to the object pose estimate. Timing offsets between the drone's pose estimate and the camera image frame also manifest as error in the object pose estimate. These two additional sources of error currently prevent our drone from performing vision-based grasps of moving targets. More sophisticated

trajectory prediction techniques or constraints on the target's possible motion might be needed to enable these moving grasps.

Appendix A

Supplementary Figures

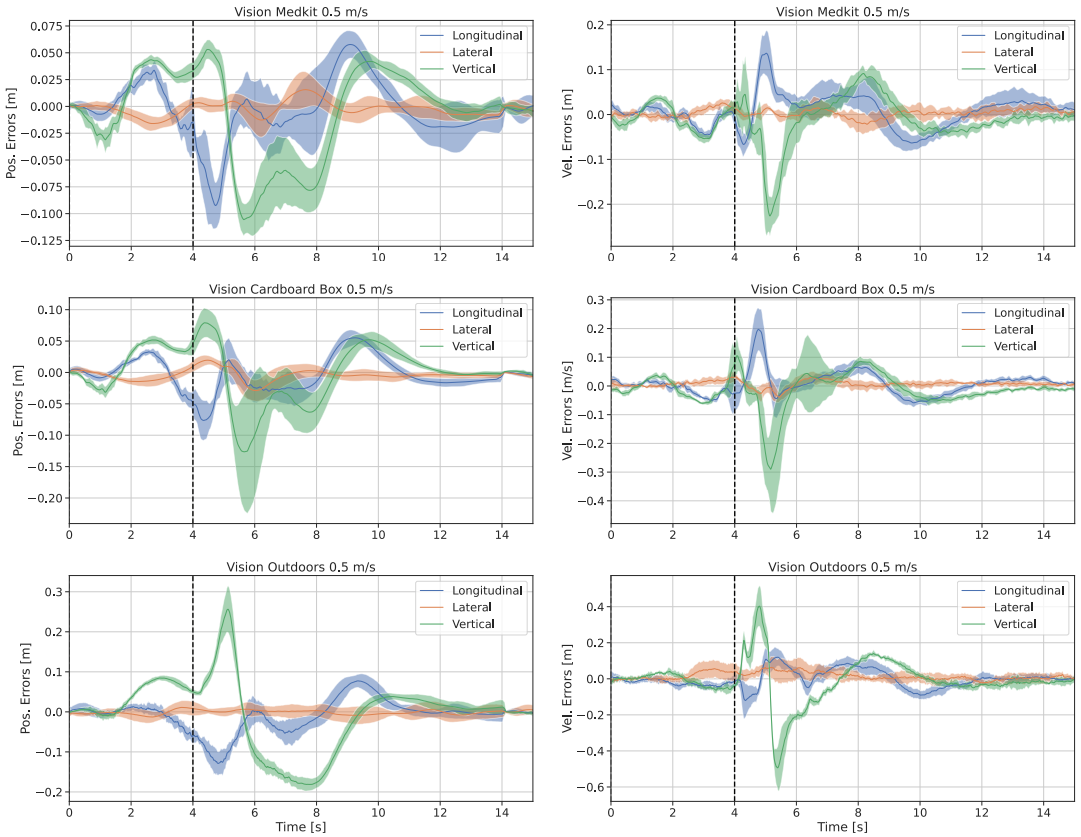


Figure A.1: Tracking error breakdown for experiments with static target (continued below). Longitudinal, lateral, and vertical position and velocity tracking errors for the vision-based experiments with static target. Before grasp, vertical errors become more positive due to ground effect. Post-grasp, the mass of the object results in more vertical errors, that are gradually compensated for.

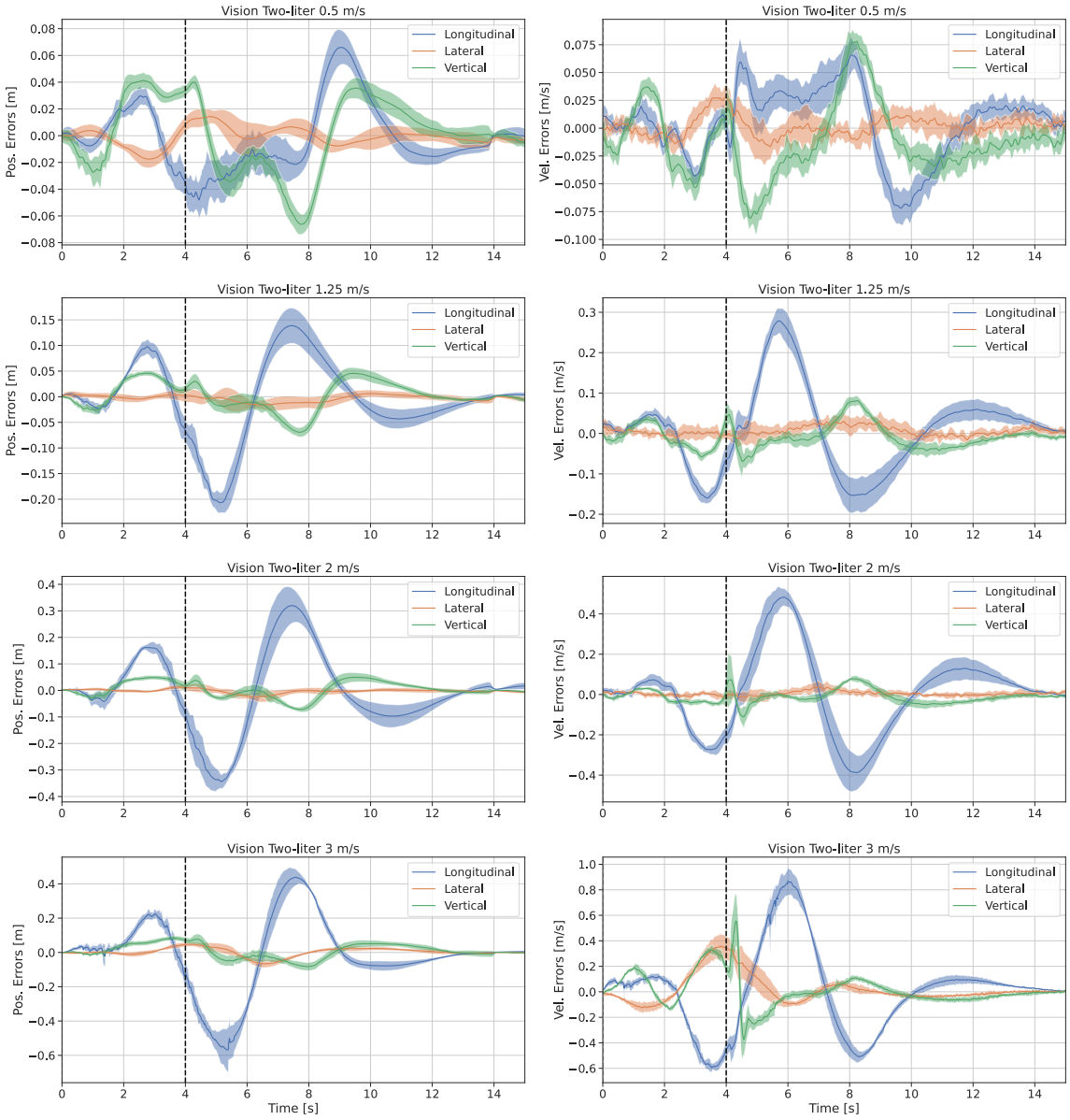


Figure A.1: Tracking error breakdown for experiments with static target (continued). As the velocity increases, we see that longitudinal errors dominate while lateral and vertical errors increase slightly. Large longitudinal tracking errors have little effect on grasp success, as the timing of the gripper closure is based on the drone's actual position with respect to the target. It is important, however, that the lateral and vertical errors are minimal at the time of grasp closure.

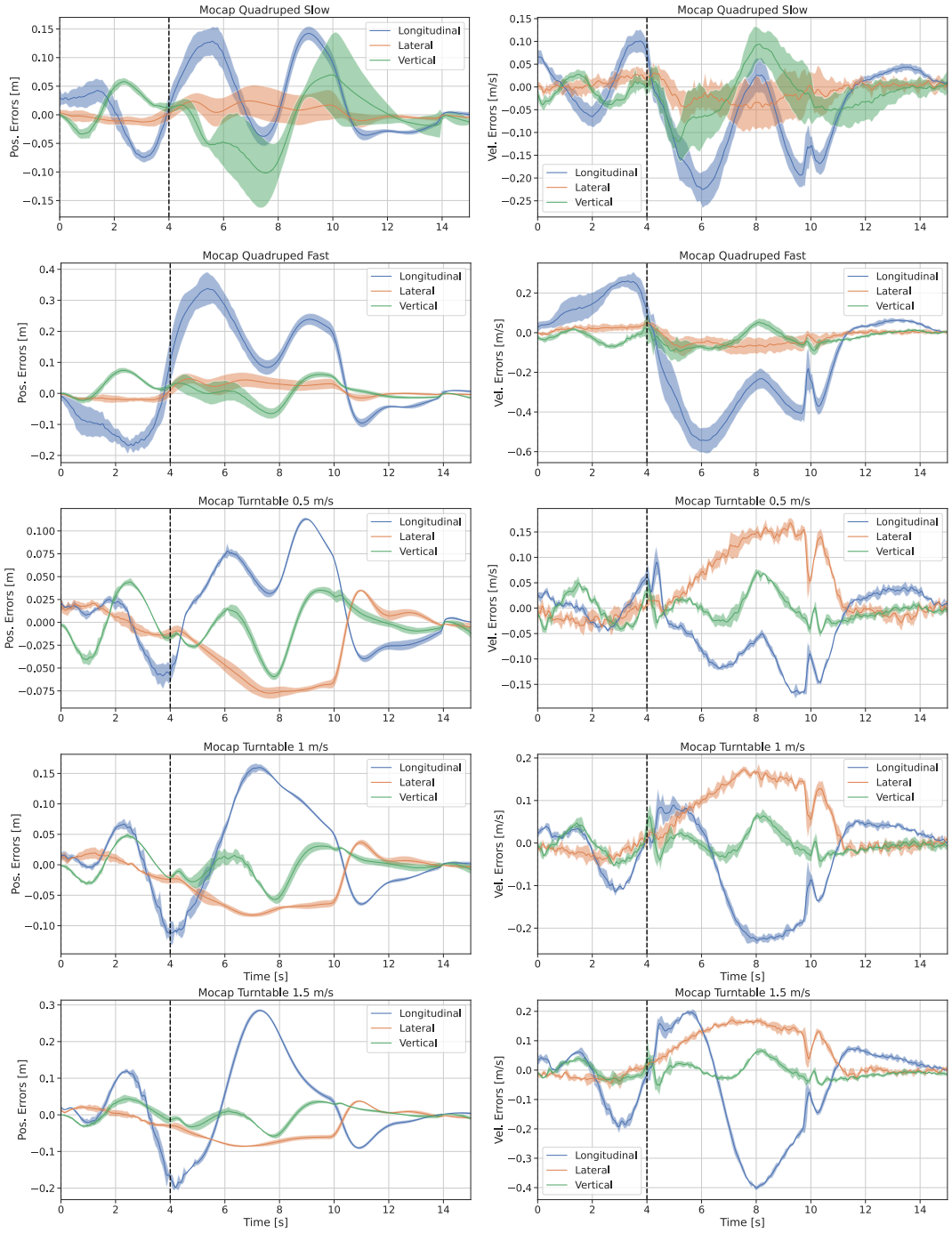


Figure A.2: Tracking error breakdown for experiments with moving target. Longitudinal, lateral, and vertical position and velocity tracking errors for the motion-capture-based experiments with moving targets. For the quadruped experiments, longitudinal errors increase greatly with increased linear speed of the target. Lateral and vertical errors also see slight increases with speed, due to either the higher variation of the target's motion or worsening of the drone's control authority at this speed. We see that there is increased lateral errors when grasping a rotating target, due to the desired grasp direction also rotating. These errors increase with increased relative grasp velocity.

A

Object	Longitudinal [cm]	Lateral [cm]	Vertical [cm]
Medkit	17.5	23.5	9
Cardboard Box	12.5	25.5	17.5
Two-liter	10	31	10

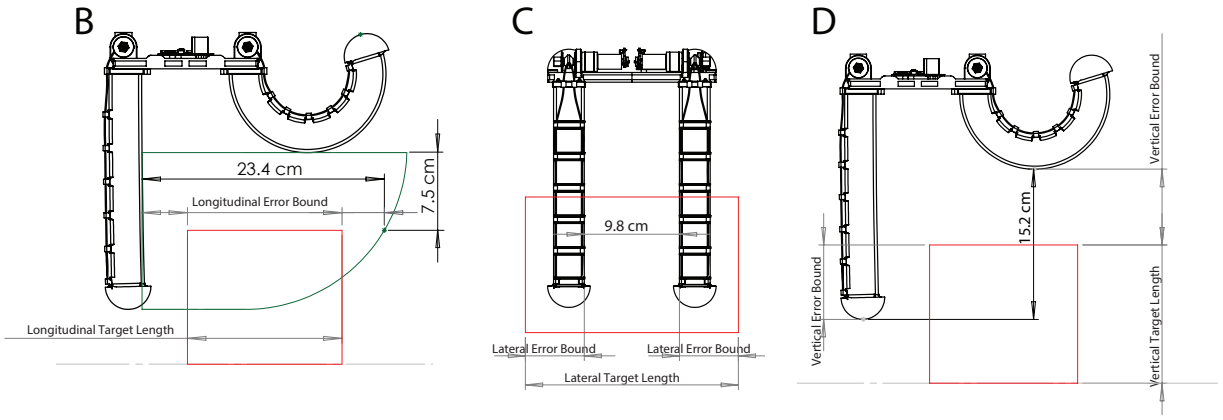


Figure A.3: Gripper dimensions and error bounds. **(A)** The table shows the dimensions of the three objects. **(B)** The longitudinal error bound. Because the fingers close in roughly a circular arc, the longitudinal gripper length decreases as it closes. Assuming the lowest point of the front finger is 7.5 cm above the top face of the target, our effective longitudinal gripper length can be computed when the finger reaches the same height as the target's top face. This effective gripper length minus the target's longitudinal length, halved, gives us our error bound. **(C)** The lateral error bound. If the drone has greater lateral error than this bound, then one half of the gripper will not make contact with the target. **(D)** The vertical error bound. If there exists high error pointing downward, the front fingers will collide with target. Conversely, if there is high error pointing upward, the rear fingers will not make contact.

Experiment	Pose Error [cm]	VIO Error [cm]	Tracking Error [cm]	Total Error [cm]	Effective Total Error [cm]	Max Allowed Error [cm]
Vision Medkit 0.5 m/s	0.42± 0.2	1.06± 0.2	2.84± 0.96	4.32± 1.09	1.48 ± 0.28	2.95
Vision Carboard Box 0.5 m/s	2.39± 0.37	1.01± 0.08	4.87± 1.71	8.27± 1.79	3.4 ± 0.38	5.45
Vision 2-liter 0.5 m/s	0.93± 0.3	1.08± 0.07	3.66± 0.93	5.67± 1.1	2.01 ± 0.31	6.7
Vision 2-liter 1.25 m/s	1.13± 0.71	2.21± 0.32	6.67± 2.09	10.02± 2.36	3.34± 0.78	6.7
Vision 2-liter 2 m/s	0.47± 0.32	3.6± 0.42	8.94± 3.64	13.02± 3.89	4.07 ± 0.53	6.7
Vision 2-liter 3 m/s	0.47± 0.32	6.74± 0.98	12.42± 5.8	19.63± 5.89	7.21 ± 1.03	6.7
Mocap Quadruped Slow	-	-	1.22± 2.24	1.22± 2.24	-	2.95
Mocap Quadruped Fast	-	-	10.18± 7.45	10.18± 7.45	-	2.95
Mocap Turntable 0.5 m/s	-	-	5.46± 1.09	5.46± 1.09	-	6.7
Mocap Turntable 1.0 m/s	-	-	11.11± 1.88	11.11± 1.88	-	6.7
Mocap Turntable 1.5 m/s	-	-	16.94± 1.15	16.94± 1.15	-	6.7

Table A.1: Longitudinal errors at the time of grasp. Longitudinal pose estimate, VIO, tracking errors, total error, and maximum allowed error for each experiment. As the majority of the grasp's motion happens in this axis, we see much higher tracking errors compared to the lateral and vertical axes. These errors increase greatly with increased velocity, due to limited control authority. However, these tracking errors have little effect on the success of the grasp. If the drone is aligned vertically and laterally, the grasp will succeed if it is triggered at the proper time, which is affected by VIO drift and our initial pose estimate. We can then compute our effective total error, which is the sum of only these two sources. We see that in the majority of our experiments, the effective total error remains within the maximum allowed longitudinal error bound derived in Appendix B.

Experiment	Pose Error [cm]	VIO Error [cm]	Tracking Error [cm]	Total Error [cm]	Max Allowed Error [cm]
Vision Medkit 0.5 m/s	3.98± 0.13	0.3± 0.12	0.49± 0.22	4.78± 0.54	6.85
Vision Carboard Box 0.5 m/s	4.07± 0.34	0.41± 0.11	1.03± 0.76	5.51± 0.92	7.85
Vision 2-liter 0.5 m/s	3.6± 0.26	0.3± 0.1	0.83± 0.54	4.72± 0.78	10.6
Vision 2-liter 1.25 m/s	3.73± 0.58	0.45± 0.32	1.12± 0.52	5.3± 1.14	10.6
Vision 2-liter 2 m/s	3.07± 0.36	0.56± 0.38	1.29± 0.73	4.92± 1.59	10.6
Vision 2-liter 3 m/s	3.07± 0.36	0.74± 0.18	4.58± 1.14	8.39 ± 1.21	10.6
Mocap Quadruped Slow	-	-	0.21± 1.26	0.21± 1.26	6.85
Mocap Quadruped Fast	-	-	1.01± 2.0	1.01± 2.0	6.85
Mocap Turntable 0.5 m/s	-	-	1.22± 0.6	1.22± 0.6	10.6
Mocap Turntable 1.0 m/s	-	-	2.37± 0.76	2.37± 0.76	10.6
Mocap Turntable 1.5 m/s	-	-	3.03± 0.71	3.03± 0.71	10.6

Table A.2: Lateral errors at the time of grasp. Lateral pose estimate, VIO, tracking errors, total error, and maximum allowed error for each experiment. We see high lateral error in our pose estimate, which we note could be accounted for by alignment errors in the ground truth pose. VIO is minimal for all speeds, while tracking errors increase with speed and are effected by the rotational motion of the turntable experiments. For all experiments, the total lateral error remains within the maximum allowed lateral error bound derived in Appendix B. Despite this, we observe complex grasp interactions such as buckling or twisting of the fingers which result in fingers failing to make secure contact in the lateral direction, and, consequently, failed grasps.

Experiment	Pose Error [cm]	VIO Error [cm]	Tracking Error [cm]	Total Error [cm]	Max Allowed Error [cm]
Vision Medkit 0.5 m/s	1.44± 0.15	1.31± 0.29	3.33± 0.76	6.1± 0.91	7.6
Vision Carboard Box 0.5 m/s	1.65± 0.23	1.35± 0.15	4.58± 1.69	7.58± 1.76	7.6
Vision 2-liter 0.5 m/s	1.32± 0.19	1.46± 0.15	3.36± 0.39	6.14± 0.66	7.6
Vision 2-liter 1.25 m/s	3.71± 0.98	0.83± 0.41	1.52± 0.76	6.06± 1.49	7.6
Vision 2-liter 2 m/s	2.57± 0.74	1.55± 0.67	1.94± 1.26	6.07± 2.0	7.6
Vision 2-liter 3 m/s	2.57± 0.74	1.21± 0.38	7.17± 1.53	10.95± 1.74	7.6
Mocap Quadruped Slow	-	-	1.26± 0.86	1.26± 0.86	7.6
Mocap Quadruped Fast	-	-	2.41± 0.84	2.41± 0.84	7.6
Mocap Turntable 0.5 m/s	-	-	1.61± 0.4	1.61± 0.4	7.6
Mocap Turntable 1.0 m/s	-	-	2.1± 0.3	2.1± 0.3	7.6
Mocap Turntable 1.5 m/s	-	-	1.33± 0.93	1.33± 0.9	7.6

Table A.3: Vertical errors at the time of grasp. Vertical pose estimate, VIO, tracking errors, total error, and maximum allowed error for each experiment. We see that pose and VIO errors are minimal for all experiments. Differences in the pose estimate for the two-liter experiments are explained by slight alignment differences when defining the ground truth, motion capture pose for the bottle. Tracking errors at the point of grasp are explained by ground effect, unintended pre-grasp interaction with the target, and general control authority limitations. In particular, we see high vertical tracking error at the fastest speed. This could be due to the coupling of forward velocity and altitude control for quadrotors, where our altitude authority worsens at higher speeds. The remaining experiments remain within our vertical error bounds (derived in Appendix B), albeit on the boundary in some cases. Indeed, vertical misalignment is a common failure mode.

Component	Specification
Frame	Carbon Fiber, 3mm and 1mm
Camera Mounts	3D printed PLA
Motor	Yuneec Intel Aero Ready to Fly
Propeller	Yuneec 9x4.5"
Electronic Speed Controller	Tekko32 F4 4 in 1 Mini 45A
Power Distribution Board	Holybro PM06 v2.0
Flight Controller	Pixhawk 4 Mini
RC Receiver	Spektrum FPV Racing Serial Receiver
Battery	4-cell, 3200mAh, 50C LiPo Spektrum G2 smart battery
Target Camera	Intel Realsense D455
VIO Camera	Intel Realsense T265 Tracking Camera
Onboard Compute	Jetson Xavier NX (6 ARM cores, 8GB unified RAM/VRAM)
Jetson Carrier Board	Leetop A203 v2

Table A.4: Drone specifications. Component list for the drone platform and the perception system.

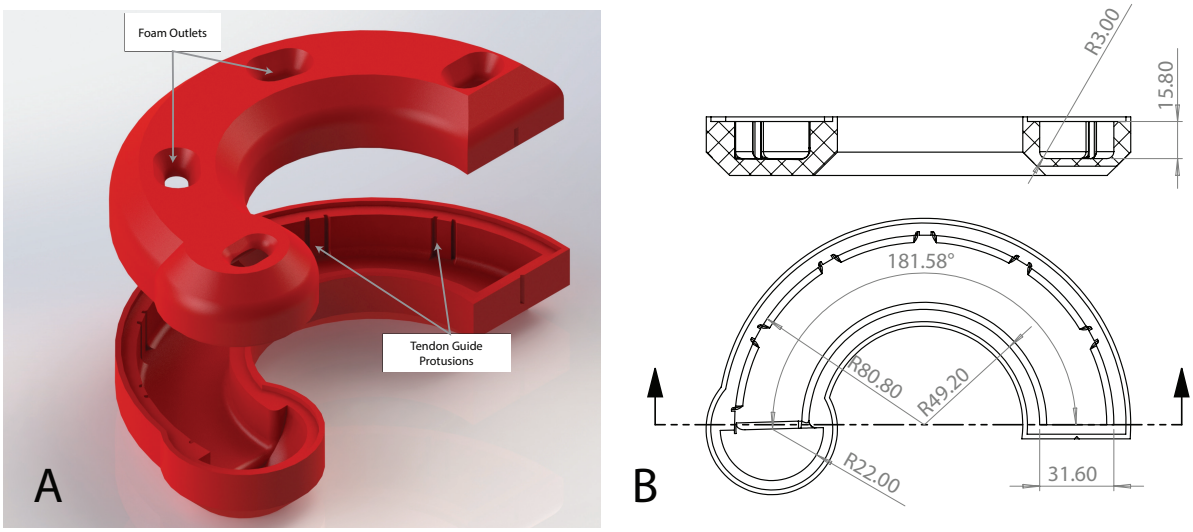


Figure A.4: Passively closed finger mold. (**A**) The mold is 3D printed into a circular arc and consists of a top and bottom, such that the mold can be opened to remove the finger after it has cured. The top piece has outlets for excess liquid foam to escape during the curing process. Evenly spaced slits are placed in the interior of the mold; this leaves small foam protrusions on the cured finger that help mark the placement of the tendon guides. (**B**) Annotated drawing of the bottom mold, detailing the mold's radius and cross-sectional dimensions.

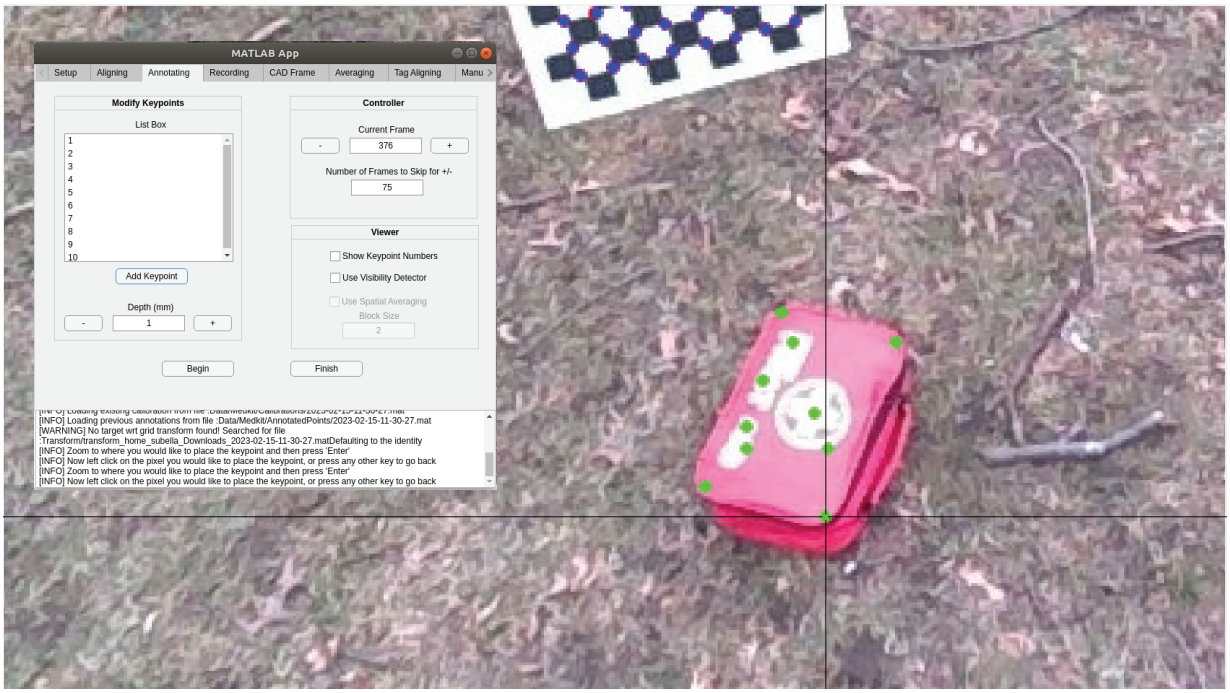


Figure A.5: Automated data annotation tool. Example of the automated keypoint annotation Matlab GUI. The tool automatically parses RGB and depth images from a Rosbag and computes the camera pose for each image. Here, the user is manually placing a keypoint on the corner of the medkit. With known camera poses and leveraging the depth channel of the camera, this keypoint can then be projected back to pixel space for all subsequent frames. Once all the keypoints are labelled and refined, the user can switch to the *recording* tab, which will annotate the pixel locations of keypoints in every image and format the data to be ready to train the network.

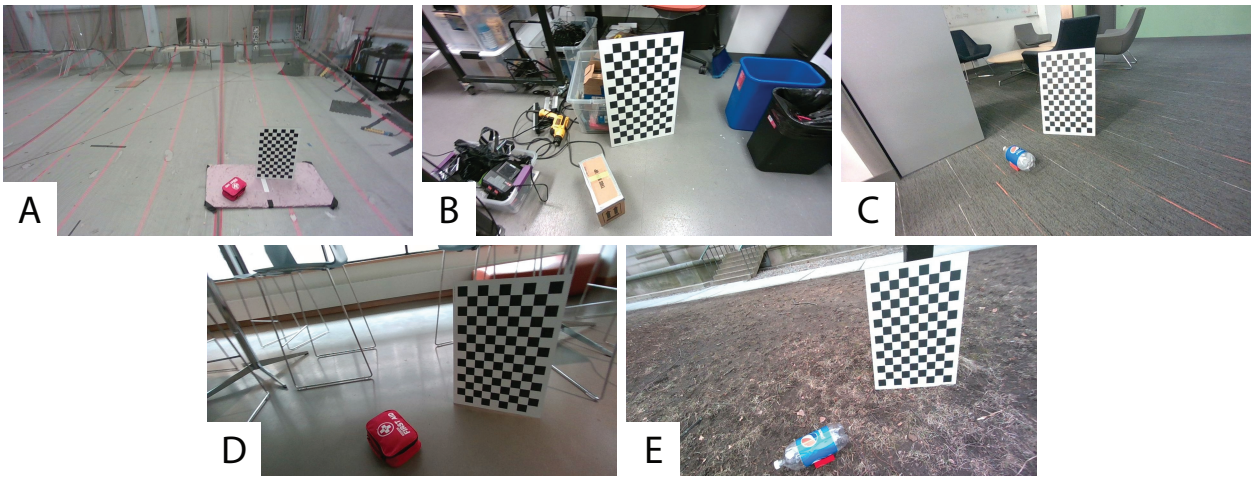


Figure A.6: Keypoint detector training environments. To increase the generalizability of our keypoint detector, we collect training data in several environments: (**A**) motion capture room, (**B**) cluttered workshop, (**C**) lounge space, (**D**) foyer, and (**E**) outdoor field. We can quickly generate training data for new environments and targets with the aid of our automated data annotation tool.

Pipeline Parameter	Value
Model	resnet18_baseline_att
Optimizer	Adam
Learning Rate	1×10^{-3}
Random Angle	[-0.2, 0.2]
Random Scale	[0.5, 2.0]
Random Translate	[-0.2, 0.2]
Brightness	0.15
Contrast	0.1
Saturation	0.1
Hue	0.1

Table A.5: Keypoint detector training parameters. All relevant, custom parameters used to train the model. The parameters correspond to values specified in trt-pose's training pipeline.

$$\Omega_i^{-1} = \begin{cases} \Sigma_d + \begin{bmatrix} t_1\, I_3 & 0 \\ 0 & r_1\, I_3 \end{bmatrix} \\ \begin{bmatrix} t_i\, I_3 & 0 \\ 0 & r_i\, I_3 \end{bmatrix} \quad \text{i=2,3,4} \end{cases} \tag{A.1}$$

Symbol	Name	Value
t_1	Observation Translation Variance	0.2 [m]²
r_1	Observation Rotation Variance	0.05 [rad]²
t_2	Motion Model Position Error Variance	0.005 [m]²
r_2	Motion Model Rotation Error Variance	0.005 [rad]²
t_3	Motion Model Velocity Error Variance	0.01 [m/s]²
r_3	Motion Model Angular Velocity Variance	0.01 [rad/s]²
t_4	Velocity Prior Variance (regularization)	10 [m/s]²
r_4	Angular Velocity Prior Variance (regularization)	1 [rad/s]²

Table A.6: Fixed-lag smoother parameters. Eq. A.1 defines the weight matrices $\Omega_1, \ldots, \Omega_4$ in the fixed-lag smoother (Eq. 3.3) based on the the parameters in this table. Ω_1 scales the penalty on error between predicted and observed object pose, where Σ_d is the covariance matrix of the drone's odometry estimate. Note that Eq. A.1 assumes the measurement noise of the object relative to the camera is isotropic. In general, the observation noise must be rotated into the global frame. Ω_2 scales the penalty on the object pose estimate differing from a constant velocity model. Ω_3 scales the penalty associated with estimating higher changes in velocity (i.e., how constant we expect the velocity to be). Ω_4 is a small regularization term reflecting a prior on the velocity and angular velocity not being too large. Note that our chosen values of $t_4 = 10$ (m/s)^2 and $r_4 = 1$ (rad/s)^2 are quite reasonable, as any target we can hope to pick up will be moving at much slower speeds. This term is included to help ensure reliable initialization of the smoother.

Appendix B

Design Guidelines and Error Analysis

We compute an approximate geometry-based bound on the error that can be tolerated for a successful grasp. The bound can be used to predict which objects are more likely to be grasped by a given gripper design and can inform future redesigns of the soft gripper for other applications.

Towards this goal, we measured the dimensions of the smallest enclosing 3D bounding box around each object; the results are shown in Fig. A.3-A. In the best case, we assume that any positioning error that still enables all four fingers to contact this bounding box could still result in a successful grasp, but positioning errors outside this range will likely fail. We can then approximate longitudinal, lateral, and vertical error bounds by using the gripper dimensions in its pre-grasp configuration and computing the maximum tolerable error (Fig. A.3-B-D) that still allows the four fingers to make contact. For our analysis, we assume that the gripper's center is aligned with the target's center laterally, and the top face of the target is vertically centered between the lowest points on the rear and front fingers.

For the longitudinal error bound (Fig. A.3-B), the solid green region depicts the approximate area the gripper encapsulates as it closes. Because the front fingers close in a circular arc, the effective longitudinal gripper length decreases as the fingers close. Given our assumed drone's vertical position, the effective longitudinal gripper length at the moment the front fingers would contact the target is $\delta_1 = 23.4$ cm. Our nominal control policy aims to have closed the gripper

when the target is at the center of this longitudinal grasp length. If the gripper closes too early, the front fingers will contact the top of the target, rather than the side, and the grasp may fail. If the gripper closes too late, the gripper is more robust as the rear fingers can carry the target while the front fingers continue to close. Calling ℓ_1 the longitudinal target length, the maximum tolerable longitudinal error can be computed as $\frac{\delta_1 - \ell_1}{2}$.

The lateral error bound (Fig. A.3-C) can be found similarly. If there exists so much lateral error the one half of the gripper does not make contact with the target, the grasp is unlikely to succeed. We measure the interior lateral gripper length between two pairs of fingers to be $\delta_2 = 9.8$ cm, and —calling ℓ_2 the lateral target length— compute the lateral error bound as $\frac{\ell_2 - \delta_2}{2}$

For the vertical error bound (Fig. A.3-D), there exists two major failure modes: the rear fingers are too high to secure a grasp, or the front fingers are too low such that they collide with the target before grasp. We are less concerned with the rear fingers contacting the ground thanks to their compliance. We measure the vertical gripper length between the lowest point on the rear fingers and the lowest point on the front fingers as $\delta_3 = 15.2$ cm. As the top face of the target is assumed to be centered in this interval, the vertical error bound is simply $\delta_3/2$ for all targets. We note that in practice, we tune the vertical offset between the drone and the target at the grasp point empirically, rather than centering it.

We report these error bounds, as well as our worst-case, averaged observed errors in the longitudinal, lateral, and vertical axes in Tables A.1, A.2, A.3. To obtain our worst case total error for our experiments, we assume that the pose estimation error, VIO drift, and tracking error across each axis sum together in a way that maximizes error.

In general, we see a bias in the pose estimation errors, particularly in the lateral axis. We conjecture that this bias is due to a slight misalignment between the vision and motion capture frames, and not an issue in our approach. Additionally, we note that our VIO metric is not a true reflection of the exact amount of drift at the time of grasp (see Section 3.3.4). Because the timing of the gripper closure only depends on the VIO state estimate and initial pose estimate of the target, the longitudinal tracking errors have a minimal effect on grasp performance, but are still reported

in the total error metric. Finally, we do not have ground truth target pose data available for the two-liter 3 m/s experiment, and instead use the estimates from the two-liter 2 m/s experiment as there should be minimal difference.

We note that most experiments remained within our theoretical error bounds. We remark that these bounds are conservative: a grasp is unlikely to succeed outside of these bounds, but is not guaranteed to succeed inside, due to the complex interaction between target and soft gripper.

Appendix C

Evaluating Reference Polynomial in Object Frame

Minimum-snap polynomials are widely used for generating quadrotor reference trajectories between waypoints. As the system is differentially flat [33], [51] in position and yaw, a reference trajectory will be feasible if it has a continuous 4th derivative (subject to actuator limits). Each axis of the reference trajectory can be solved independently. We solve for a separate polynomial for each position axis, resulting in $p_x(t), p_y(t), p_z(t)$, by solving the following optimization problem:

$$\begin{aligned} &\min_{p} \int_0^{t_f} ||p^4(t)||^2 dt \\ &\quad s.t. \\ &\quad p(0) = x_0, \quad \dot{p}(0) \quad = 0, \quad \ddot{p}(0) = 0, \\ &\quad p(t_g) = x_g, \quad \dot{p}(t_g) \quad = v_g, \quad \ddot{p}(t_g) = 0, \\ &\quad p(t_f) = x_f, \quad \dot{p}(t_f) \quad = 0, \quad \ddot{p}(t_f) = 0, \end{aligned} \tag{C.1}$$

where p is a 10th degree polynomial, $p^4(t)$ is its fourth derivative with respect to t, x_0 is the drone's current position in the relevant axis, x_f is the desired post-grasp hover point, t_g is the desired time

of grasp, x_g is the grasp waypoint, and v_g is the grasp speed. This optimization can be solved as a quadratic program (QP), although contrary to [33], [52] we solve for a single polynomial to represent both segments of the trajectory. We use CVXOPT to solve the QP [53].

The curve

$$\boldsymbol{x}(t) = [p_x(t), p_y(t), p_z(t)], \quad t \in [0, t_f]$$

represents the position setpoint over time of the drone during a grasp attempt. Similarly,

$$\boldsymbol{v}(t) = \dot{\boldsymbol{x}}(t) = [\dot{p}_x(t), \dot{p}_y(t), \dot{p}_z(t)], \quad t \in [0, t_f]$$
$$\boldsymbol{a}(t) = \ddot{\boldsymbol{x}}(t) = [\ddot{p}_x(t), \ddot{p}_y(t), \ddot{p}_z(t)], \quad t \in [0, t_f].$$

represent its velocity and acceleration. The yaw of the drone is set such that the drone points toward its estimate of the object position at all times before grasp, and it maintains its yaw at the grasp point for times after t_g.

To enable the drone to grasp moving targets, we evaluate the grasp trajectory relative to the moving target frame. As the lower-level controller expects a position, velocity, and acceleration setpoint in the drone's fixed odometry frame, we must transform $[\boldsymbol{x}(t),\ \boldsymbol{v}(t),\ \boldsymbol{a}(t)]$ from the object's moving frame to the fixed frame. We define the following variables:

The equations for transforming the setpoint into the global reference frame are then

$$\boldsymbol{x}_o^f = \boldsymbol{x}_m^f + \boldsymbol{x}_o^m \tag{C.2}$$
$$\boldsymbol{v}_o^f = \boldsymbol{v}_m^f + \boldsymbol{\omega}_m^f \times \boldsymbol{x}_o^m + \boldsymbol{v}_o^m \tag{C.3}$$
$$\boldsymbol{a}_o^f = \boldsymbol{a}_m^f + \boldsymbol{\alpha}_m^f \times \boldsymbol{x}_o^m + \boldsymbol{\omega}_m^f \times \left[\boldsymbol{\omega}_m^f \times \boldsymbol{x}_o^m\right] + 2\boldsymbol{\omega}_m^f \times \boldsymbol{v}_o^m + \boldsymbol{a}_o^m. \tag{C.4}$$

See, *e.g.,* [54, Chapter 4], for more exposition on this transformation. In our experiments, we assume the moving object's angular acceleration is zero, setting $\boldsymbol{\alpha}_m^f = \mathbf{0}$.

Variable Names	Description
$\boldsymbol{x}_o^f, \boldsymbol{v}_o^f, \boldsymbol{a}_o^f$	Position, velocity, and acceleration setpoints for the drone in the fixed odometry frame, expressed in the basis of the fixed odometry frame.
$\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a}$	Position, velocity, and acceleration setpoints for the drone relative to the moving target frame, expressed in the moving object frame.
$\boldsymbol{x}_o^m, \boldsymbol{v}_o^m, \boldsymbol{a}_o^m$	Position, velocity, and acceleration setpoints for the drone relative to the moving target frame, expressed in the basis of the fixed odometry frame. For a rotation $\boldsymbol{R}_m^f$ from the moving frame to the fixed frame, $[\boldsymbol{x}_o^m, \boldsymbol{v}_o^m, \boldsymbol{a}_o^m] = \boldsymbol{R}_m^f[\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{a}]$
$\boldsymbol{x}_m^f, \boldsymbol{v}_m^f, \boldsymbol{a}_m^f,$ $\boldsymbol{\omega}_m^f, \boldsymbol{\alpha}_m^f$	Position, velocity, acceleration, angular velocity, and angular acceleration of the moving target frame relative to the fixed odometry frame, expressed in the fixed odometry frame.

References

[1] M. Idrissi, M. R. Salami, and F. Annaz, "A review of quadrotor unmanned aerial vehicles: Applications, architectural design and control algorithms," *Journal of Intelligent and Robotic Systems*, vol. 104, Jan. 2022. DOI: 10.1007/s10846-021-01527-7.

[2] N. Joubert, J. L. E, D. B. Goldman, F. Berthouzoz, M. Roberts, J. A. Landay, and P. Hanrahan, *Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles*, 2016. arXiv: 1610.01691 [cs.GR].

[3] M. Hassanalian and A. Abdelkefi, "Classifications, applications, and design challenges of drones: A review," *Progress in Aerospace Sciences*, vol. 91, pp. 99–131, 2017, ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2017.04.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042116301348.

[4] S. Gupte, P. Mohandas, and J. Conrad, "A survey of quadrotor unmanned aerial vehicles," Mar. 2012, pp. 1–6, ISBN: 978-1-4673-1374-2. DOI: 10.1109/SECon.2012.6196930.

[5] P. H. Nguyen, K. Patnaik, S. Mishra, P. Polygerinos, and W. Zhang, "A soft-bodied aerial robot for collision resilience and contact-reactive perching," *Soft Robotics*, 2023.

[6] R. Zufferey, J. Tormo-Barbero, D. Feliu-Talegón, S. R. Nekoo, J. Á. Acosta, and A. Ollero, "How ornithopters can perch autonomously on a branch," *Nature Communications*, vol. 13, no. 1, p. 7713, 2022.

[7] W. R. Roderick, M. R. Cutkosky, and D. Lentink, "Bird-inspired dynamic grasping and perching in arboreal environments," *Science Robotics*, vol. 6, no. 61, eabj7562, 2021.

[8] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. Angst, R. Siegwart, and J. Nieto, "An omnidirectional aerial manipulation platform for contact-based inspection," *Arxiv*, 2019. [Online]. Available: https://arxiv.org/pdf/1905.03502.pdf.

[9] E. Aucone, S. Kirchgeorg, A. Valentini, L. Pellissier, K. Deiner, and S. Mintchev, "Drone-assisted collection of environmental dna from tree branches for biodiversity monitoring," *Science Robotics*, vol. 8, no. 74, eadd5762, 2023. DOI: 10.1126/scirobotics.add5762. eprint: https://www.science.org/doi/pdf/10.1126/scirobotics.add5762. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.add5762.

[10] F. Augugliaro, A. Mirjan, F. Gramazio, M. Kohler, and R. D'Andrea, "Building tensile structures with flying machines," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3487–3492. DOI: 10.1109/IROS.2013.6696853.

[11] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014. DOI: 10.1109/MCS.2014.2320359.

[12] M. Xu, S. Huang, R. He, D. Yu, and H. Wang, "Aerial shooting manipulator for distant grasping," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 1991–1998, 2023. DOI: 10.1109/LRA.2023.3245399.

[13] A. X. Appius, E. Bauer, M. Blöchlinger, A. Kalra, R. Oberson, A. Raayatsanati, P. Strauch, S. Suresh, M. von Salis, and R. K. Katzschmann, "Raptor: Rapid aerial pickup and transport of objects by robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 349–355. DOI: 10.1109/IROS47612.2022.9981668.

[14] P. E. Pounds, D. R. Bersak, and A. M. Dollar, "The Yale Aerial Manipulator: Grasping in flight," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011.

[15] M. Ryll and R. K. Katzschmann, "Smors: A soft multirotor uav for multimodal locomotion and robust interaction," in *2022 International Conference on Robotics and Automation (ICRA)*, IEEE, 2022, pp. 2010–2016.

[16] J. Thomas, J. Polin, K. Sreenath, and V. Kumar, "Avian-inspired grasping for quadrotor micro uavs," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, vol. 55935, 2013, V06AT07A014.

[17] R. Spica, A. Franchi, G. Oriolo, H. H. Bülthoff, and P. R. Giordano, "Aerial grasping of a moving target with a quadrotor uav," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 4985–4992.

[18] R. Rossi, A. Santamaria-Navarro, J. Andrade-Cetto, and P. Rocco, "Trajectory Generation for Unmanned Aerial Manipulators Through Quadratic Programming," *IEEE Robotics and Automation Letters*, 2017.

[19] S. Kannan, S. Quintanar-Guzman, J. Dentler, M. A. Olivares-Mendez, and H. Voos, "Control of aerial manipulation vehicle in operational space," in *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, 2016, pp. 1–4. DOI: 10.1109/ECAI.2016.7861157.

[20] H. Yu, P. Wang, J. Wang, J. Ji, Z. Zheng, J. Tu, G. Lu, J. Meng, M. Zhu, S. Shen, *et al.*, "Catch planner: Catching high-speed targets in the flight," *arXiv preprint arXiv:2302.04387*, 2023.

[21] T. G. Chen, K. A. W. Hoffmann, J. E. Low, K. Nagami, D. Lentink, and M. R. Cutkosky, "Aerial grasping and the velocity sufficiency region," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 009–10 016, 2022. DOI: 10.1109/LRA.2022.3192652.

[22] J. Sun, Y. Wang, M. Feng, D. Wang, J. Zhao, C. Stachniss, and X. Chen, "Ick-track: A category-level 6-dof pose tracker using inter-frame consistent keypoints for aerial

manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1556–1563. DOI: 10.1109/IROS47612.2022.9982183.

[23] J. Fishman, S. Ubellacker, N. Hughes, and L. Carlone, "Dynamic grasping with a "soft" drone: From theory to practice," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (pdf), 2021.

[24] E. Bauer, B. G. Cangan, and R. K. Katzschmann, "Autonomous vision-based rapid aerial grasping," *arXiv preprint arXiv:2211.13093*, 2022.

[25] L. Lin, Y. Yang, H. Cheng, and X. Chen, "Autonomous vision-based aerial grasping for rotorcraft unmanned aerial vehicles," *Sensors*, vol. 19, no. 15, p. 3410, 2019.

[26] P. Ramon-Soria, B. C. Arrue, and A. Ollero, "Grasp planning and visual servoing for an outdoors aerial dual manipulator," *Engineering*, vol. 6, no. 1, pp. 77–88, 2020, ISSN: 2095-8099. DOI: https://doi.org/10.1016/j.eng.2019.11.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2095809919308653.

[27] H. Seo, S. Kim, and H. J. Kim, "Aerial grasping of cylindrical object using visual servoing based on stochastic model predictive control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 6362–6368. DOI: 10.1109/ICRA.2017.7989751.

[28] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2113–2118. DOI: 10.1109/ICRA.2014.6907149.

[29] J. Bern, P. Banzet, R. Poranne, and S. Coros, "Trajectory optimization for cable-driven soft robot locomotion," in *Robotics: Science and Systems (RSS)*, 2019.

[30] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, 2015.

[31] A. A. Stokes, R. F. Shepherd, S. A. Morin, F. Ilievski, and G. M. Whitesides, "A hybrid combining hard and soft robots," *Soft Robotics*, vol. 1, no. 1, pp. 70–74, 2014.

[32] G. Huang, "Visual-inertial navigation: A concise review," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 9572–9582. DOI: 10.1109/ICRA.2019.8793604.

[33] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.

[34] F. Goodarzi, D. Lee, and T. Lee, "Geometric adaptive tracking control of a quadrotor unmanned aerial vehicle on SE(3) for agile maneuvers," *J. Dyn. Sys., Meas., Control.*, no. 9, 137 2015.

[35] J. M. Bern, G. Kumagai, and S. Coros, "Fabrication, modeling, and control of plush robots," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2017.

[36] N. Kastor, R. Mukherjee, E. Cohen, V. Vikas, B. A. Trimmer, and R. D. White, "Design and manufacturing of tendon-driven soft foam robots," *Robotica*, vol. 38, no. 1, pp. 88–105, 2020.

[37] Nvidia, *trt_pose*, https://github.com/NVIDIA-AI-IOT/trt_pose, 2023.

[38] H. Yang, J. Shi, and L. Carlone, "TEASER: Fast and Certifiable Point Cloud Registration," *IEEE Trans. Robotics*, vol. 37, no. 2, pp. 314–333, 2020, extended arXiv version 2001.07715 (pdf).

[39] D. Del Cont Bernard, M. Giurato, F. Riccardi, and M. Lovera, "Ground effect analysis for a quadrotor platform," in *Advances in Aerospace Guidance, Navigation and Control*, 2018, pp. 351–367.

[40] Smooth On, *FlexFoam-iT! X*, https://www.smooth-on.com/products/flexfoam-it-x/, 2023.

[41] K. Blomqvist, J. J. Chung, L. Ott, and R. Siegwart, *Semi-automatic 3d object keypoint annotation and detection for the masses*, 2022. arXiv: 2201.07665 [cs.CV].

[42] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige, *Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects*, 2020. arXiv: 1912.02805 [cs.CV].

[43] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications To Tracking and Navigation*. John Wiley and Sons, 2001.

[44] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, 2017, arxiv preprint: 1512.02363, (pdf), technical report GT-IRIM-CP&R-2015-001.

[45] F. Dellaert and G. Contributors, *Borglab/gtsam*, May 2022. DOI: 10.5281/zenodo.5794541. [Online]. Available: https://github.com/borglab/gtsam).

[46] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 217–236, 2 2012.

[47] M. Li, Z. Ferguson, T. Schneider, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics.," 2020.

[48] J. M. Bern and D. Rus, "Soft ik with stiffness control," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, IEEE, 2021, pp. 465–471.

[49] M. Grupp, *Evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo, 2017.

[50] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, "Kpam: Keypoint affordances for category-level robotic manipulation," in *The International Symposium of Robotics Research*, Springer, 2019, pp. 132–157.

[51] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 8, no. 11, pp. 995–1020, 1998.

[52] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*, Springer, 2016, pp. 649–666.

[53] M. S. Andersen, J. Dahl, and L. Vandenberghe, *Cvxopt: A python package for convex optimization, version 1.2.6*, version 1.2.6. [Online]. Available: https://cvxopt.org/.

[54] A. Kelly, *Mobile robotics: mathematics, models, and methods*. Cambridge University Press, 2013, ch. 10.