2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Robust, Perception Based Control with Quadrotors

Laura Jarin-Lipschitz[†], Rebecca Li[†], Ty Nguyen, Vijay Kumar, and Nikolai Matni

Abstract— **Traditionally, controllers and state estimators in robotic systems are designed independently. Controllers are often designed assuming perfect state estimation. However, state estimation methods such as Visual Inertial Odometry (VIO) drift over time and can cause the system to misbehave. While state estimation error can be corrected with the aid of GPS or motion capture, these complementary sensors are not always available or reliable. Recent work has shown that this issue can be dealt with by synthesizing robust controllers using a data-driven characterization of the perception error, and can bound the system's response to state estimation error using a robustness constraint. We investigate the application of this robust perception-based approach to a quadrotor model using VIO for state estimation and demonstrate the benefits and drawbacks of using this technique in simulation and hardware. Additionally, to make tuning easier, we introduce a new cost function to use in the control synthesis which allows one to take an existing controller and "robustify" it. To the best of our knowledge, this is the first *robust* perception-based controller implemented in real hardware, as well as one utilizing a data-driven perception model. We believe this as an important step towards safe, robust robots that explicitly account for the inherent dependence between perception and control.**

I. INTRODUCTION

In real-world robotics with size, weight, and power constraints, a vision sensor is a cheap and lightweight sensor solution to localize a robot in its environment. In particular, vision-based localization using Visual-Inertial Odometry (VIO) has been an important technique in bringing multi-rotor robots out of the lab and into real-world environments [1]. However, visual perception systems can produce state estimates that drift over time, accumulating error that ultimately limits the capability of the robotic system. Poor lighting conditions and a lack of features to track in the environment can make visual state estimation even worse.

While many robotic systems seek to solve this using near-ideal sensors such as motion capture or GPS, this is often impractical or impossible. For lightweight or low-cost robotic systems, adding GPS or other ground truth localization sensors might not be possible. Additionally, there are many cases such as GPS-denied environments or underground environments where robots may not get GPS signal updates in order to correct state estimation error growth. Localization using motion capture systems such as VICON is only possible in lab and indoor environments, significantly limiting use in the real world.

Historically, roboticists have assumed the state estimation of their system to be "good enough," and proceeded to perform control and planning while assuming perfect state information. Theoretically, this is supported by the separation principle [2], which says that for certain linear stochastic systems, optimal control and optimal state estimation can be decoupled and still remain optimal. The assumptions needed for these guarantees (e.g. quadratic cost, unbiased Gaussian noise) are rarely, if ever satisfied by real-world vision-based systems. Instead, control actions taken by the system can significantly impact the quality and content of the images used for state estimation. In this work, we explicitly account for the interplay between perception and control, and furthermore leverage the controller to reduce perception error in a robotic system.

The approach we take in this work is to make the system robust to state estimation error through the use of a robust, perception based controller. We follow the approach outlined in [3] to incorporate a data-driven perception error model into an optimal controller synthesis procedure. By utilizing this procedure, we can guarantee that the perception error will remain bounded given a well-characterized perception map. Specifically, we implement the robust, perception based controller on a quadrotor, a platform where VIO is preferable to other state estimators due to its size, weight, and cost constraints. We demonstrate that for a quadrotor, the state estimation and tracking for the robust, perception based controller is considerably better than an L_1 controller, and matches that of a tuned PD controller on typical conditions. We also demonstrate that the robust, perception based controller remains robust to perturbations in the environment such as weaker ambient light and fewer texture features while the PD control suffers. In short, our contributions include:

- Introduction and implementation of a robust, perception based controller with a data-driven error model on a quadrotor with VIO, in simulation and reality. To the best of our knowledge, we are the first to implement a *robust*, perception-based controller on a hardware system.
- Introduction of a new cost function, the *imitation cost*, which allows the roboticist to use an existing controller and "robustify" it.
- Evaluation of the robust, perception based controller with a data-driven error model using traditional quadratic cost functions and the imitation cost function, in comparison to Proportional-Derivative (PD) and quadratic cost L1 optimal controllers. We evaluate these controllers in ideal perception and degraded perception conditions.

[†] These authors contributed equally to the work
The authors are a part of the GRASP Lab, University of Pennsylvania, PA, 19104, USA {laurajar, robot, kumar, tynguyen, nmatni}@seas.upenn.edu.

II. Related Work

Controllers that incorporate perception into control and planning have been an important part of enabling robotic systems to move into unstructured environments. For quadrotor platforms, Visual-Inertial Odometry has been a popular technique to fuse vision with IMU measurements for accurate state estimation [1], [4]. These utilize Kalman filter based estimators to produce good state estimates. However, state estimation error with these algorithms grows significantly over time. Simultaneous Localization and Mapping (SLAM) is a tool used to match current perception measurements to a constructed map, and is used widely in UAV platforms [5], [6]. Unfortunately, SLAM has extremely high computational and memory load, which makes it unsuitable for small or low-cost platforms.

Another approach is to explicitly consider perception in the control or planning. In [7], a Model Predictive Controller (MPC) controller is presented which, in addition to having trajectory following, also has perception objectives. Perception-aware planning is yet another approach, which can avoid parts of the space where localization may be lost [8]. Visual servoing, or explicitly controlling given vision input, has incorporated robustness in [9] and [10] by explicitly take into account model error, but not perception error in their models.

More accurate perception models can be achieved through learning. In [11], a rover robot was able to use a learned perception model to navigate safely through a novel environment. Racing drones with learning perception systems were also demonstrated by [12]. However, none of these systems provide any control-theoretic guarantees, nor do they attempt to bound the perception or tracking error of the system. In contrast, we define the explicit consideration and presence of guarantees to constitute robust control, which is extremely important in safety-critical applications. In [3], the authors introduce a robust, perception based controller, but only demonstrate it on a relatively simple double integrator system. In this work, we seek to get much closer to bridging the gap between the simplistic models that are often used in robotics, and real, complex systems by incorporating robust guarantees informed by data-driven perception models.

III. Problem Formulation

A. Notation

We use lower case letters x to denote vectors, and capital letters A to denote matrices. x_k denotes the discrete signal x at time k, while $x_{0:k}$ denotes the discrete signal up to time k, $\{x_0, x_1, ..., x_k\}$. Bolded letters $\mathbf{y} = \mathbf{K}\mathbf{x}$ denote infinite horizon signals, and linear convolution operators, such that $y_k = \sum_{t=0}^{k} K_t x_{k-t}$. The norm operator $||\cdot||$ is overloaded to apply to signals and convolution operators: $||\mathbf{x}|| = \sup_k ||x_k||$ and $||\mathbf{\Phi}|| = \sup_{||\mathbf{w}|| \le 1} ||\mathbf{\Phi}\mathbf{w}||$. The norm on linear operators $||\mathbf{\Phi}||$ is also the induced norm. If unmarked, the norm is assumed to be the ∞-norm.

B. LTI System with Perception Model

We consider a LTI system, following the setup in [3]:

$$x_{k+1} = Ax_k + Bu_k + Hw_k \tag{1}$$
$$z_k = q(x_k) \tag{2}$$
$$y_k = p(z_k) = Cx_k + e_k \tag{3}$$
$$u_k = \mathbf{K}(y_{0:k}) \tag{4}$$

In this system, the state at time k is $x_k \in \mathbb{R}^6$, the input is $u_k \in \mathbb{R}^3$, the high-dimensional pixels and IMU measurement is z_k, the measurement after utilizing the perception map is $y_k \in \mathbb{R}^6$, and the system is perturbed by noise w_k with $||H|| = 1$. The matrices $A, B, C,$ and H are all known and described in Section III-E. The linear time invariant output feedback controller $\mathbf{K}$ is described later in this section.

The high-dimensional measurement z_k represents the raw sensors' measurements which depend on the system's state according to an observation process $q(x_k)$. In a VIO system, z_k corresponds to raw images and IMU measurements. The VIO algorithm takes z_k as an input and outputs lower dimensional state measurement y_k, according to the VIO perception map $p(z_k)$. We model the output of the perception map $p(z_k)$ as being composed of two terms: the first, Cx_k, corresponds to an idealized sensor output (e.g., the full system state), whereas the second, $e_k := p(z_k) - Cx_k$, is a state-dependent error term. With this perception model in place, we can ignore the intermediate measurement z_k, resulting in a direct noisy observation model $y_k = Cx_k + e_k$, as VIO systems directly provide an approximate system state measurement. At a high level, our goal is to ensure that the system response to perception errors e_k remains bounded. System Level Synthesis (SLS) [13] provides a framework for doing so. SLS provides a parameterization of the achievable closed-loop behaviors of the LTI system (1) under the feedback policy $\mathbf{u} = \mathbf{K}\mathbf{y}$ that makes explicit the effects of process noise $H\mathbf{w}$ and measurement errors $\mathbf{e}$ on system behavior. This means that we can write the state and input as result of a convolution of the state noise, perception noise and the closed-loop system responses $\{\Phi_{xw}(k), \Phi_{xe}(k), \Phi_{uw}(k), \Phi_{ue}(k)\}$,

$$\begin{bmatrix} x_k \\ u_k \end{bmatrix} = \sum_{t=1}^{k} \begin{bmatrix} \Phi_{xw}(t) & \Phi_{xe}(t) \\ \Phi_{uw}(t) & \Phi_{ue}(t) \end{bmatrix} \begin{bmatrix} Hw_{k-t} \\ e_{k-t} \end{bmatrix}, \tag{5}$$

which we can compactly represent in the z-domain as:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi_{xw}} & \mathbf{\Phi_{xe}} \\ \mathbf{\Phi_{uw}} & \mathbf{\Phi_{ue}} \end{bmatrix} \begin{bmatrix} H\mathbf{w} \\ \mathbf{e} \end{bmatrix} \tag{6}$$

In [13], it is shown that to characterize all possible system responses achievable by a stabilizing feedback controller $\mathbf{K}$, it is necessary and sufficient to constrain these system responses to be stable, and satisfy the affine constraints

$$\begin{gathered} \begin{bmatrix} zI - A & -B \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_{xw}} & \mathbf{\Phi_{xe}} \\ \mathbf{\Phi_{uw}} & \mathbf{\Phi_{ue}} \end{bmatrix} = \begin{bmatrix} I & 0 \end{bmatrix} \\ \begin{bmatrix} \mathbf{\Phi_{xw}} & \mathbf{\Phi_{xe}} \\ \mathbf{\Phi_{uw}} & \mathbf{\Phi_{ue}} \end{bmatrix} \begin{bmatrix} zI - A \\ -C \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix}. \end{gathered} \tag{7}$$

Then, given any system responses satisfying these constraints, the controller $\mathbf{K} = \mathbf{\Phi_{ue}} - \mathbf{\Phi_{uw}}\mathbf{\Phi_{xw}}^{-1}\mathbf{\Phi_{xe}}$ will achieve the desired behavior in system (1). We point the reader to [13] for a thorough explanation of SLS, and Section 4.1 of [3] for a more targeted overview of SLS as applied to the perception based problem considered in this paper.

From (6), we see that $\mathbf{x} = \mathbf{\Phi_{xw}} H\mathbf{w} + \mathbf{\Phi_{xe}}\mathbf{e}$. Thus, the closed loop map from perception error $\mathbf{e}$ to state $\mathbf{x}$ is $\mathbf{\Phi_{xe}}$. Intuitively, the size of the system response $||\mathbf{\Phi_{xe}}||$ captures how aggressively the system can react to the perception error $\mathbf{e}$ – therefore, in order to bound the system's response to perception errors, we need to bound $||\mathbf{\Phi_{xe}}||$. This bound will be influenced by the characteristics of $\mathbf{e}$; if $\mathbf{e}$ is large, $||\mathbf{\Phi_{xe}}||$ must be small, roughly speaking. As such, we first characterize an error model $\mathbf{e}$ according to (3). Specifically, using ground truth data measurements y_k and deriving C from the system, we can then characterize the nonlinear error $||\mathbf{e}|| = ||p(\mathbf{z}) - C\mathbf{x}||$ from training data. Then, we use the properties of $\mathbf{e}$ to constrain $||\mathbf{\Phi_{xe}}||$ in the controller optimization problem. This constraint is the *robustness constraint* that endows the resulting controllers with robustness guarantees, as described at the end of this section.

C. Data-Driven Error Model & Robust Synthesis

In order to craft a robustness constraint, we need to characterize our perception error model $\mathbf{e}$. Our assumption of a static observation model $z_k = q(x_k)$ implies that the perception error model fit on training data remains valid if those states are visited again at test time. Under suitable smoothness assumptions on the observation model q and perception map p, we expect the error model to be approximately valid on states near to those visited during training. But how close do we need to stay in order for our perception error model to be approximately valid, and what rate does its accuracy degrade? The answer from [3] is to quantify the so-called S-slope of the error model in a neighborhood of the training data to quantify this degradation. One can think of the S-slope as a worst-case bound on the rate at which the perception error model degrades as the system moves away from states found in the training data. Formally, we define the data-driven estimate $\hat{S}$ of the S-slope with radius r around the training data $\{x_1, ..., x_n\}$ as:

$$\hat{S} = \max_{x_i \in \mathcal{S}_{x_d}} \frac{||e(x_i) - e(x_d)||}{||x_i - x_d||} \tag{8}$$

$$\mathcal{S}_{x_d} = \{x_1, ..., x_n : ||x_i - x_d|| < r\} \tag{9}$$

It is shown in [3] that using both the S-slope and a norm bound $||\mathbf{e}||$ of the error model, a robustness constraint can be added to the control synthesis problem to guarantee a bounded perception error. In particular, we utilize the SLS problem formulation for a robust, perception based controller from [3]:

$$\min_{\mathbf{\Phi}} \quad c(\mathbf{\Phi_{xw}}, \mathbf{\Phi_{uw}}, \mathbf{\Phi_{xe}}, \mathbf{\Phi_{ue}}) \tag{10}$$

$$\text{s.t.} \quad \text{Equation (7)}, \tag{11}$$

$$||\mathbf{\Phi_{xe}}|| < \frac{1 - \frac{1}{r}||\mathbf{\Phi_{xw}} H\mathbf{w} - \mathbf{x_d}||}{S + \frac{\epsilon_e}{r}} \tag{12}$$

from which we obtain the system level responses $\{\mathbf{\Phi_{xw}}, \mathbf{\Phi_{xe}}, \mathbf{\Phi_{uw}}, \mathbf{\Phi_{ue}}\}$, and can construct the controller $\mathbf{u} = \mathbf{K}\mathbf{y}$, with $\mathbf{K} = \mathbf{\Phi_{ue}} - \mathbf{\Phi_{uw}}\mathbf{\Phi_{xw}}^{-1}\mathbf{\Phi_{xe}}$. We note that the formulation shown here is for an infinite horizon time response. In practice we utilize a Finite Impulse Response approximation of this procedure, outlined in [13] and [3].

As described in Theorem 4.2 from [3], the resulting controller guarantees that the system remains within a bounded distance of states visited during training, and consequently, we can guarantee that our perception errors $||\mathbf{e}|| = ||p(\mathbf{z}) - C\mathbf{x}||$ also remain bounded:

$$||\mathbf{e}|| = ||p(\mathbf{z}) - C\mathbf{x}|| \leq \gamma := \frac{||\hat{\mathbf{x}} - \mathbf{x}_d|| + R_0}{1 - S\,||\mathbf{\Phi_{xe}}||} \tag{13}$$

Our goal in this work is to demonstrate the practical relevance of this robustness guarantee in a quadrotor system.

D. Imitation Cost Function

In order to synthesize a controller, we need a cost function. A typical choice is the quadratic cost c_{QR} with state cost weights Q and input cost weights R [3]. The subsequent controller is called LQG or L_1 controllers, depending on choice of norm in the cost function (L_2 or L_∞):

$$\begin{aligned} &c_{QR}(\mathbf{\Phi_{xw}}, \mathbf{\Phi_{uw}}, \mathbf{\Phi_{xe}}, \mathbf{\Phi_{ue}}) \\ &= \left|\left|\begin{bmatrix} Q^{1/2} & \\ & R^{1/2} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_{xw}} & \mathbf{\Phi_{xe}} \\ \mathbf{\Phi_{uw}} & \mathbf{\Phi_{ue}} \end{bmatrix} \begin{bmatrix} \varepsilon_w H \\ \varepsilon_e I \end{bmatrix}\right|\right| \end{aligned} \tag{14}$$

One problem with this cost function is that if the controller need to be adjusted for better performance, it can be difficult to tune this cost function with Q and R. While there are formal rules of thumb in crafting Q and R, small changes can result in wildly different performance. Additionally, the difference between the quadratic cost controller with and without the robustness constraint (12) can be significant.

In this work, we introduce a new cost function called *imitation cost* c_{im}. The intuition behind this cost is that for most applications of robust control, an existing, non-robust controller already exists and works well. We do not want to synthesize a controller with completely different behavior from this nominal controller, but instead would simply like to have a controller which "imitates" the nominal controller, while still being robust. Hence, the cost should be to minimize the difference between the robust controller and the desired controller while subject to the robustness constraint (12).

We define the nominal controller system responses as $\mathbf{\Phi_{xw,n}}, \mathbf{\Phi_{uw,n}}, \mathbf{\Phi_{xe,n}}, \mathbf{\Phi_{ue,n}}$, and still retain the quadratic cost structure such that it is a convex program. In practice, we find this controller much easier to "tune", since we may simply tune our nominal controller, which could be as simple and intuitive as a PD controller, and then impose the robustness constraint afterwards through optimization. For this cost, we found the L_1 yielded the best performance.

$$\begin{aligned} &c_{im}(\mathbf{\Phi_{xw}}, \mathbf{\Phi_{uw}}, \mathbf{\Phi_{xe}}, \mathbf{\Phi_{ue}}) \\ &= \left|\left|\begin{bmatrix} Q^{1/2} & \\ & R^{1/2} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi_{xw}} - \mathbf{\Phi_{xw,n}} & \mathbf{\Phi_{xe}} - \mathbf{\Phi_{xe,n}} \\ \mathbf{\Phi_{uw}} - \mathbf{\Phi_{uw,n}} & \mathbf{\Phi_{ue}} - \mathbf{\Phi_{ue,n}} \end{bmatrix}\right|\right| \end{aligned} \tag{15}$$

E. Linear Quadrotor Model

In order to use the results from [3], we require an LTI model of our quadrotor. We assume an underlying attitude controller as described in [14], and that we are not flying aggressively. Thus we can linearize our quadrotor around hover while holding yaw constant to get an LTI model.

We define the states of the quadrotor to be the position and velocity of the center of mass: $\mathbf{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$. The inputs for the quadrotor are total propeller thrust and instantaneous attitude in Euler angles as ψ, θ and ϕ, or yaw, pitch and roll. Yaw is held constant at $\psi = 0$, and thus dropped from the input space. Thus, our final control inputs are pitch, roll, and total thrust or $\mathbf{u} = [\theta\ \phi\ f_t]^T$.

The quadrotor system is linearized around hover as $\bar{\mathbf{x}} = [0\ 0\ 0\ 0\ 0\ 0]^T$, $\bar{\mathbf{u}} = [0\ 0\ mg]^T$, where g is acceleration due to gravity and m is the mass of the quadrotor. Finally, we can describe our continuous LTI system:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{16}$$

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -g & 0 & 0 \\ 0 & g & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix} \tag{17}$$

The perception map of our system $p(z_k)$ from VIO returns measurements of position and velocity, so C is simply the identity:

$$y_k = p(z_k) = Cx_k + e_k \tag{18}$$

$$C = I_{6\times 6} \tag{19}$$

To match both our simulation and hardware setups, we convert our system to discrete time with $\Delta T = 0.1$ seconds.

IV. Experimental Setup

Our experiments consist of flying around a trajectory near hover, with access to both ground truth and real state estimation of our quadrotor system. To generate the desired trajectory for the position controller, we utilize a time-parameterized circular trajectory at a constant height. For state estimation, we use VIO which functions as the perception map. In the experiment, we first fly our trajectory with a non-linear SO_3 controller from [14] in order to gather data to train the perception error model. From the error model, we synthesize controllers using SLS, then fly the same circle trajectory to compare results.

A. Simulation Platform

ROS and Gazebo function as the simulation environment for a World Electronics Dragon Drone Development quadrotor. To simulate the visual perception, we utilize a Multi-state Constraint Kalman Filter with Visual-Inertial Odometry (MSCKF-VIO) [4], which is integrated with simulated IMU and stereo camera signals from Gazebo in order to produce a VIO measurement. We designed this simulation platform to best mimic real platforms while still having access to perfect ground truth in order to verify our guarantees.

B. Hardware Platform

The quadrotor platform is based on the quadrotor used in [15]. It is a 0.16 m diameter, 256 g quadrotor using a Qualcomm®, Snapdragon™ Flight™ with a 7.4 V LiPo battery. The hardware platform is depicted in Figure 1. The visual measurement system consists of a downward facing camera with a fish-eye lens and 107°. field of view. The downwards facing camera is the main differentiator from the simulated VIO system, but otherwise is very similar to the simulated quadrotor. For the ground truth, we utilize the VICON Motion Capture system.

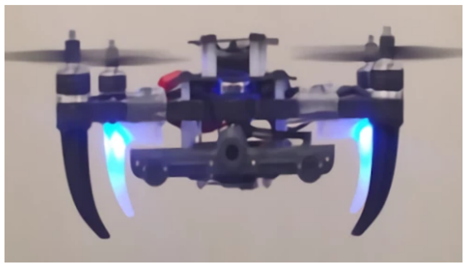

Fig. 1: The Snapdragon platform we used in our hardware experiments. It features a downward facing camera and an on-board VIO state estimator.

V. Results

In order to generate the robust controllers, we first characterize the error profile of the perception system to generate a robust, perception based controller in Section V-A. We then test our controllers in simulation and reality. Unfortunately, only the PD and L_1 Robust controllers were sufficiently safe to realize on the hardware platform. We compare controller performance in Sections V-C and V-D. State estimation of controllers as well as comparison to the theoretical bound is discussed in Section V-F.

A. Perception Error Profile

For the error model, we wish to characterize the perception map $p(z_k)$ by fitting an error profile to match $e_k = p(z_k) - Cx_k$ using training data. The training data were generated by flying a circular trajectory using a high performance non-linear SO_3 controller. The norm and local S-slope bounds $||e_k|| \leq \epsilon_e$, and S are utilized in the SLS procedure described in equations (10). The S-slope represents how quickly the perception error can change, and thus if S is large, then the controller must be extremely conservative as the perception bias may grow extremely quickly away from training data points.

While theoretically sound, unfortunately, the empirical local S-slope of the error profile was found to be an over-conservative bound of reality. Our characterization of the error profile resulted in $S = 11.7$ in simulation and $S = 14.4$ in real experiments. This resulted in too stringent a robustness constraint, rendering the SLS optimization problem (10) infeasible. In reality, only a few points in each data set exhibit such a high S-slope - meaning that a few measurements have high noise. This is expected, as there are often "bad measurements" (especially in velocity) which are largely outliers and not representative of the perception bias growing extremely quickly. Therefore, we take our estimate of the S-slope for the purposes of our robust controller to be far less conservative such that the SLS procedure is feasible. We further justify it by evaluating in our results how well our robustness guarantee was maintained, even with a less conservative S-slope on the error profile, as seen in the next Section V-F. In future work, we hope to address this problem with a more appropriate perception model.

B. Controllers

We compared the following four controllers, all of which were synthesized using optimization problem (10), except for the PD controller:

- Well-tuned PD controller. This controller was tuned to track very well under ideal perception conditions.
- Nominal L_1 controller. This controller was synthesized using the cost function from equation (14) with the L_1 norm, and without enforcing the robustness constraint (12). We choose to use this instead of a Linear Quadratic Gaussian due to superior performance. The only difference is the use of the $\infty \to \infty$-induced-norm or Frobenius norm in the cost of the controller synthesis problem.
- L_1 Robust controller with the normal cost. This controller was synthesized using the cost function from equation (14).
- L_1 Robust imitation controller. This controller was synthesized using the cost function from equation (15). The nominal controller it was imitating is the L_1 controller.

In our hardware experiments, only the $x-y$ position control of the L_1 robust imitation controller was stable enough to test against the PD controller. The z control of the PD controller was used instead of the z control for the L_1 robust imitation controller for safety. We have matched our simulated data to also use the z-control of the PD controller for a more direct comparison.

C. Simulated Controller Performance

The trajectory tracking was done around a circular trajectory. The simulated resulting position tracking error can be seen in Figure 2, while the estimation errors can be seen in Figure 4. In the simulated experiments, the controllers all tracked the trajectory, with the PD controller tracking the best. Qualitatively, the robust controllers reacted more conservatively to errors in state estimation, as seen in Figure 3. We see that at the beginning of the trajectory between 0-4 seconds, there is a disturbance in the state estimation to which the L_1 robust controller reacts quickly, as we can see with the spike, but quickly damps out to a more conservative control than the PD controller. This results in a "softer" controller overall, but is important in real-life when erroneous measurements make their way into the control. Indeed, there is a trade-off in how aggressively a controller can react to changes in sensor measurements, and how robust it can be to perception errors. Our robust controller gives a principled way to navigate this trade-off, allowing us to balance between over-reacting to perception errors and maintaining good tracking.

D. Hardware Controller Performance

The position tracking in the hardware experiments is shown in Figure 2. The PD controller and L_1 controller are on par in tracking performance, however, the PD control looks slightly smoother. The perception error is also about the same for both controllers, as seen in Figure 4.

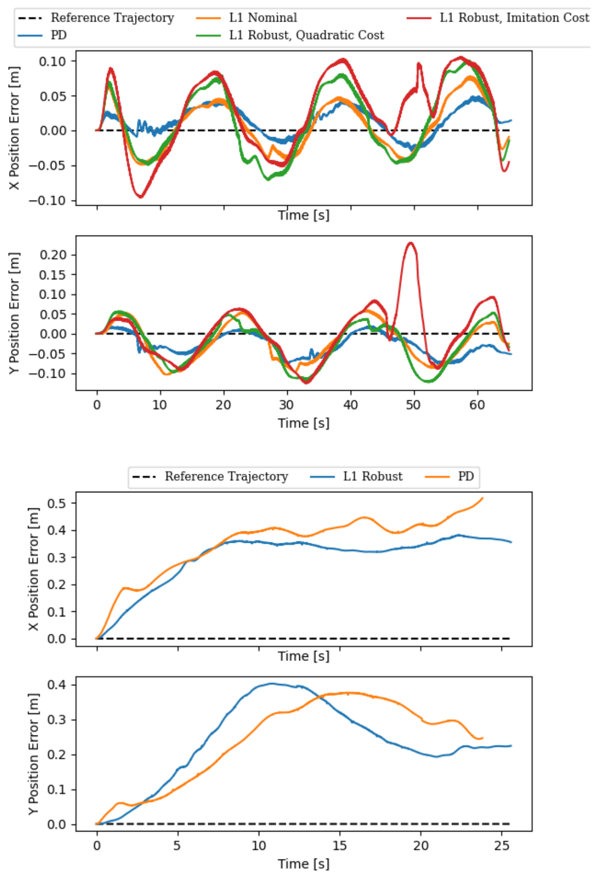

Fig. 2: The position tracking of the controllers around the circle trajectory. (Top) The controllers in simulation did three laps, and all controllers perform similarly well.
(Bottom) Position tracking of the L_1 Robust imitation controller and PD controller on the hardware system. Both had fairly similar performance qualitatively, though the PD controller tracked slightly closer than the L_1 robust controller.

E. Sources of Difficulty in Controller Performance

We suspect the difficulty in synthesizing the optimal controllers for the real quadrotor system lies in the fact that the quadrotor is linearized in hover, but is not truly a linear system. In the simulation, where process noise is closer to Gaussian, the controllers are able to stabilize the system as expected. However, in real life, the process noise may be more due to model error, which may interfere with the synthesized controllers in a pathological way.

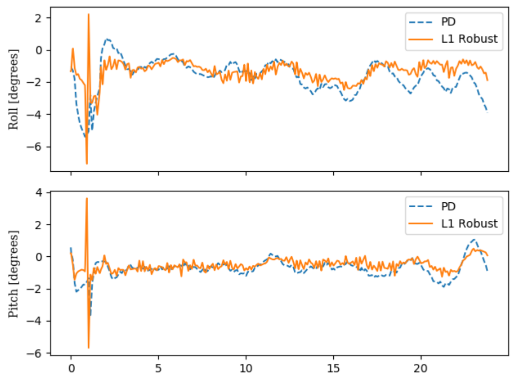

Fig. 3: The control input on a L1 robust imitation control run compared to what the PD control from the same state estimate.

For the simulation, we still experienced significant difficulty in tuning controllers to be performant enough to satisfy the task. The synthesized controllers often let the quadrotor drift farther from the desired trajectory than the PD controller. Only after much tuning did we achieve tracking error similar to PD as seen in Figure 2. We hypothesize that with a more stable system, the robust controllers as well as nominal L_1 controllers would be adequate controllers. However, even with our generous interpretation of the error profile S-slope, the trade-off between minimizing perception error and minimizing tracking error is too far in the direction of minimizing perception error. Ultimately, the quadrotor needs to track well enough to perform its task. Qualitatively speaking, this highlights a deficiency in this robust, perception based controller formulation that the robustness constraint gives a hard limit on how much the roboticist tune her preference for minimizing perception error or tracking error while remaining robust. The "tuning knobs" available in the cost function given in equation (14) are the Q and R matrices, which in our experience was insufficient to produce a controller good enough to use on a real hardware platform.

F. Bounded Perception Error Guarantee

The first guarantee in equation (13) simply states that our state estimation error will remain bounded. We can confirm this directly in simulation and reality by comparing our ground truth (simulation or VICON motion capture) to the VIO state estimate.

We compare the perception error to the theoretical bound in equation (13), with a slight modification. We compare instead to a slightly stricter bound, which removes the dependence on the distance to the training data $||\hat{\mathbf{x}} - \mathbf{x}_d||$, which is difficult to calculate.

$$||p(\mathbf{z}) - C\mathbf{x}|| \leq \frac{R_0}{1 - S\,||\mathbf{\Phi}_{\mathbf{xe}}||} \tag{20}$$

In this case, the perception error still stays below the theoretical limit, for all controllers, in simulation and reality, as seen in Figures 4. The numerical values for these limits are 1.602 in simulation and 3.172 in our hardware experiments, calculated using the true S-slope of the true error profiles. We note that these are an order of magnitude larger than the errors we see in either simulation or reality. While our controllers satisfy these theoretical bounds, they are hardly reassuring to consider that a quadrotor may drift by almost a meter in position (since most of the state estimation error bias comes from position), which is not acceptable for many applications.

Additionally, we performed the same long-trajectory tests with increased noise in the virtual camera sensor, which further highlighted the difference between the naive and robust controller performance. The perception error in these trials can be seen in Figure 4. In this case, the L_1 quadratic cost robust and nominal controllers performed significantly better than the PD and L_1 robust imitation cost controllers. This suggests that the quadratic cost controllers may perform significantly better than controllers tuned for ideal perception conditions. In particular, the L_1 robust controller with quadratic cost appears to reduce the perception error the most. However, once again, all of the controllers stay far below the perception error theoretical bound.

VI. Discussion

In this work, we implemented a robust, perception based controller on a quadrotor platform. With this controller design, the control is aware of the perception error, and drives the quadrotor state to keep the perception error bounded. We implement the first version of a *robust*, perception-based controller on a simulated and hardware quadrotor platform. In implementing and testing the control synthesis procedure with a quadrotor perception system, we encountered issues in the controller design relying on our perception map being unrealistically smooth, causing the control synthesis problem to be infeasible. In reality, using a slightly less conservative characterization of the perception map resulted in a feasible but still empirically robust perception-based controller.

We also introduced a new cost function to aid with tuning the robust, perception based controller. It takes an existing good controller and "robustifies" it with the SLS procedure, which in both simulation and hardware we found to be a more reliable way to produce good controllers.

We tested the robust, perception based controller against a well-tuned PD controller and L_1 controller and saw that the robust, perception based controller and PD controller performed equally well, significantly better than the L_1 controller. Both PD and the robust, perception based controller satisfied the robustness constraint.

In degraded perception conditions mimicking adverse lighting conditions, the robust, perception based controller adjusted its control to maintain better perception. Overall, we found that the robust control was overly conservative for well-lit conditions, and only for degraded perception does its advantages come into play. This suggests further experimentation in more challenging state estimation conditions.

VII. Future Work

Moving forward, we would like to explore new controller formulations using System Level Synthesis which do not

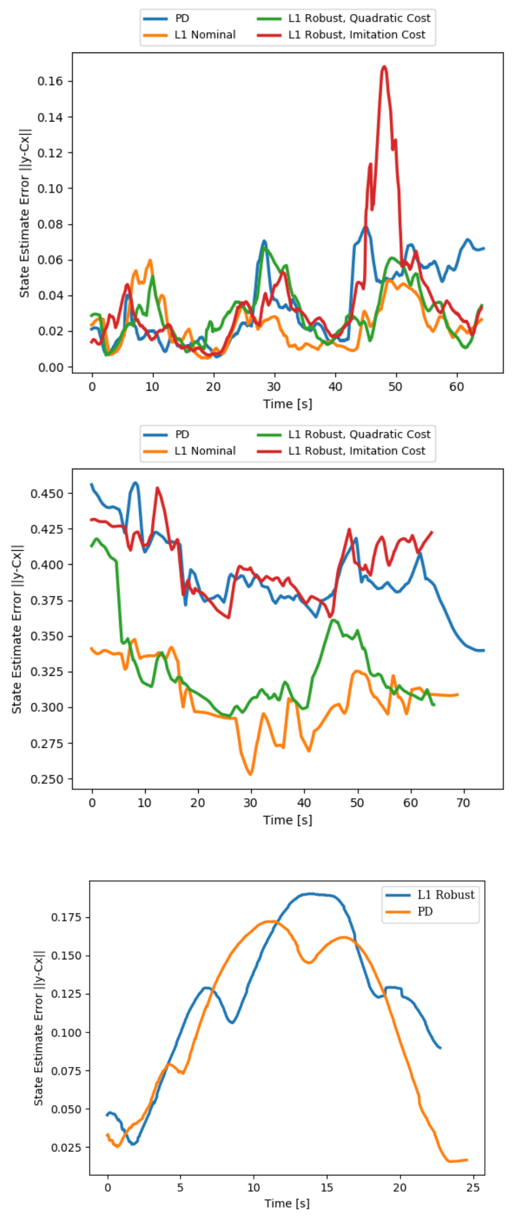

Fig. 4: The norm of the perception error over time from simulation with regular perception (Top), simulation with degraded perception (Middle) and our hardware experiments (Bottom). The perception error stays below the theoretical bound for all controllers. The signals has been averaged with a time constant of 1 second for readability.

rely upon overly-conservative characterizations of perception noise such as S-slope. Instead, we believe interesting work lies in characterizing parts of the state space as having better or worse state estimation, and using that information to formulate a robust, perception aware controller. We would also like to incorporate a linear time-varying (LTV) version of the robust perception-based control synthesis, based on [16]. This will allow us to incorporate yaw back into the controller for the quadrotor, which is important for forward-facing camera scenarios and would be a robust complement to current research that investigates active sensing with yaw.

We also see promise in maintaining the robustness constraint. This would be applicable in patrol or inspection situations, where a vehicle needs to be robust but still operate in a noisy environment. The robust controller could also be used in combination with a safe exploration policy, where reliance on perfect perception is unrealistic, yet safety and robustness are still needed.

ACKNOWLEDGMENTS

We would like to acknowledge Dinesh Thakur and Alex Zhou for their help in using the Snapdragon Quadrotor platform and the VICON Motion capture system, and Ke Sun for his help with the simulated perception.

This work was supported in part by the Semiconductor Research Corporation (SRC) and DARPA.

REFERENCES

[1] G. Loianno, M. Watterson, and V. Kumar, “Visual inertial odometry for quadrotors on SE(3),” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2016-June, no. 3, pp. 1544–1551, 2016.

[2] D. P. Joseph and T. J. Tou, “On linear control theory,” *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, vol. 80, pp. 193–196, Sep. 1961.

[3] S. Dean, N. Matni, B. Recht, and V. Ye, “Robust guarantees for perception-based control,” in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120 of *Proceedings of Machine Learning Research*, pp. 350–360, PMLR, 10–11 Jun 2020.

[4] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, “Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[5] S. Lynen, B. Zeisl, D. Aiger, M. Bosse, J. Hesch, M. Pollefeys, R. Siegwart, and T. Sattler, “Large-scale, real-time visual-inertial localization revisited,” *arXiv preprint arXiv:1907.00338*, 2019.

[6] R. Mur-Artal and J. D. Tardos, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” *IEEE Transactions on Robotics*, vol. 33, p. 1255–1262, Oct 2017.

[7] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “Pampc: Perception-aware model predictive control for quadrotors,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8, Oct 2018.

[8] B. Ichter, B. Landry, E. Schmerling, and M. Pavone, “Perception-aware motion planning via multiobjective search on gpus,” in *Robotics Research*, pp. 895–912, Springer, 2020.

[9] W. Zhao, H. Liu, F. L. Lewis, K. P. Valavanis, and X. Wang, “Robust visual servoing control for ground target tracking of quadrotors,” *IEEE Transactions on Control Systems Technology*, pp. 1–8, 2019.

[10] A. Dib, N. Zaidi, and H. Siguerdidjane, “Robust control and visual servoing of an uav,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 5730–5735, 2008.

[11] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, “Combining optimal control and learning for visual navigation in novel environments,” in *Conference on Robot Learning*, pp. 420–429, 2020.

[12] E. Kaufmann, M. Gehrig, P. Foehn, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, “Beauty and the beast: Optimal methods meet learning for drone racing,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 690–696, IEEE, 2019.

[13] J. Anderson, J. C. Doyle, S. H. Low, and N. Matni, “System level synthesis,” *Annual Reviews in Control*, 2019.

[14] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2520–2525, 2011.

[15] D. Thakur, G. Loianno, L. Jarin-Lipschitz, A. Zhou, and V. Kumar, “Autonomous inspection of a containment vessel using a micro aerial vehicle,” in *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 1–7, Sep. 2019.

[16] H. Wang, S. Chen, V. M. Preciado, and N. Matni, “Robust Model Predictive Control via System Level Synthesis,” *arXiv preprint on arXiv:1911.06842*, 2019.