4-15-2019

Technical Report: Anytime Computation and Control for Autonomous Systems

Yash Vardhan Pant
University of Pennsylvania, yashpant@seas.upenn.edu

Houssam Abbas
Oregon State University, houssam.abbas@oregonstate.edu

Kartik Mohta
University of Pennsylvania, kmohta@seas.upenn.edu

Rhudii A. Quaye
University of Pennsylvania, quayerhu@seas.upenn.edu

Truong X. Nghiem
Northern Arizona University, truong.nghiem@nau.edu

See next page for additional authors

Technical Report: Anytime Computation and Control for Autonomous Systems

Abstract

The correct and timely completion of the sensing and action loop is of utmost importance in safety critical autonomous systems. A crucial part of the performance of this feedback control loop are the computation time and accuracy of the estimator which produces state estimates used by the controller. These state estimators, especially those used for localization, often use computationally expensive perception algorithms like visual object tracking. With on-board computers on autonomous robots being computationally limited, the computation time of a perception-based estimation algorithm can at times be high enough to result in poor control performance. In this work, we develop a framework for co-design of anytime estimation and robust control algorithms while taking into account computation delays and estimation inaccuracies. This is achieved by constructing a perception-based anytime estimator from an off-the-shelf perception-based estimation algorithm, and in the process we obtain a trade-off curve for its computation time versus estimation error. This information is used in the design of a robust predictive control algorithm that at run-time decides a contract for the estimator, or the mode of operation of estimator, in addition to trying to achieve its control objectives at a reduced computation energy cost. In cases where the estimation delay can result in possibly degraded control performance, we provide an optimal manner in which the controller can use this trade-off curve to reduce estimation delay at the cost of higher inaccuracy, all the while guaranteeing that control objectives are robustly satisfied. Through experiments on a hexrotor platform running a visual odometry algorithm for state estimation, we show how our method results in upto a 10% improvement in control performance while saving 5-6% in computation energy as compared to a method that does not leverage the co-design.

Keywords

Model Predictive Control, Perception, Co-design, Chance Constrained MPC

Disciplines

Computer Engineering | Electrical and Computer Engineering

Author(s)

Yash Vardhan Pant, Houssam Abbas, Kartik Mohta, Rhudii A. Quaye, Truong X. Nghiem, Joseph Devietti, and Rahul Mangharam

Technical Report: Anytime Computation and Control for Autonomous Systems

Yash Vardhan Pant*, Houssam Abbas†, Kartik Mohta*, Rhudii A. Quaye*, Truong X. Nghiem‡, Joseph Devietti§ and Rahul Mangharam*§

*Department of Electrical and Systems Engineering
University of Pennsylvania, Philadelphia, USA
Email: {yashpant, kmohta, quayerhu, rahulm}@seas.upenn.edu

† Electrical Engineering and Computer Science
Oregon State University, Corvallis, USA
Email: houssam.abbas@oregonstate.edu

‡ School of Informatics, Computing, and Cyber Systems
Northern Arizona University, Flagstaff, USA
Email: truong.nghiem@nau.edu

§Department of Computer Science
University of Pennsylvania, Philadelphia, USA
Email: devietti@cis.upenn.edu

***Abstract*—The correct and timely completion of the sensing and action loop is of utmost importance in safety critical autonomous systems. A crucial part of the performance of this feedback control loop are the computation time and accuracy of the estimator which produces state estimates used by the controller. These state estimators, especially those used for localization, often use computationally expensive perception algorithms like visual object tracking. With on-board computers on autonomous robots being computationally limited, the computation time of a perception-based estimation algorithm can at times be high enough to result in poor control performance. In this work, we develop a framework for co-design of anytime estimation and robust control algorithms while taking into account computation delays and estimation inaccuracies. This is achieved by constructing a perception-based anytime estimator from an off-the-shelf perception-based estimation algorithm, and in the process we obtain a trade-off curve for its computation time versus estimation error. This information is used in the design of a robust predictive control algorithm that at run-time decides a *contract* for the estimator, or the mode of operation of estimator, in addition to trying to achieve its control objectives at a reduced computation energy cost. In cases where the estimation delay can result in possibly degraded control performance, we provide an optimal manner in which the controller can use this trade-off curve to reduce estimation delay at the cost of higher inaccuracy, all the while guaranteeing that control objectives are robustly satisfied. Through experiments on a hexrotor platform running a visual odometry algorithm for state estimation, we show how our method results in upto a 10% improvement in control performance while saving 5-6% in computation energy as compared to a method that does not leverage the co-design.**

This work was supported by the US Department of Transportation University Transportation Center (Mobility21).

This work was done when Houssam Abbas and Truong X. Nghiem were at the University of Pennsylvania.

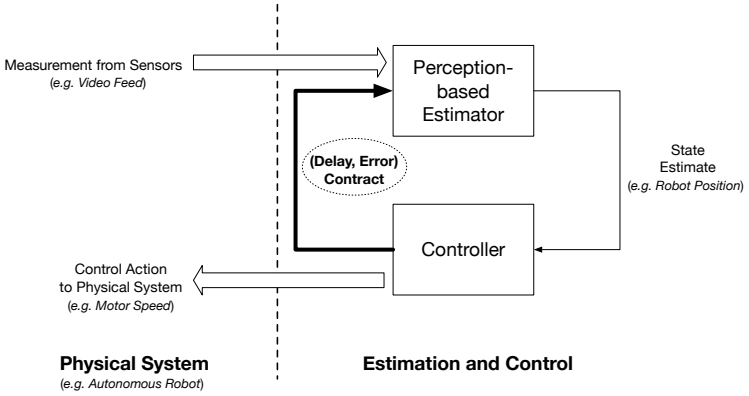

Fig. 1. Contract-driven controller and estimator. Add co-design here.

I. Introduction

The real-time control of many autonomous robots, e.g. self-driving cars and Unmanned Aerial Systems (UASs), usually includes closed loops between the controller that drives the actuation, and the estimator that computes state estimates which are used by the controller. Of particular importance in this traditional feedback control architecture are: a) the delay in the control action due to the time taken by estimator for computing the state estimate and, b) the inaccuracy in the state estimate. Either of these factors can result in control actions that can drive the system into an unsafe state.

In most conventional feedback control designs, controllers are tasked with realizing the functional goals of the system under simplistic assumptions on the performance of the estimator, in particular, perfect state estimates and negligible computation time. This design principle based on separation of

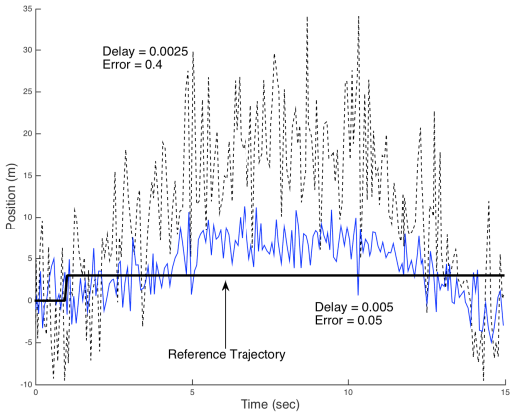

Fig. 2. Effect of delay, error values on control performance of a PID tracker.

concerns simplifies the control design process but often does not accurately reflect real implementations. On the other hand, most perception-based state estimation algorithms (e.g. SVO [1] and ORB-SLAM [2]) do not take into account how their output will be used to close the control loop. More specifically, an estimator will more often than not *run to completion*: i.e., its termination criteria are designed to provide the best quality output (estimate). This can result in large delays in the control action, leading to degraded control performance. It can also result in the computation platform consuming a significant amount of energy, reducing the amount of time the system can operate on a full charge. This is specially of concern in mobile robotic systems like autonomous drones and cars, that operate on batteries with limited capacity.

In this work we focus on these problems, that when the real-time requirements on the closed-loop system become more demanding, this disconnect between the estimator and controller can lead to poor system performance. The following example shows how this problem can manifest in even simple settings.

Example 1. *To illustrate the impact of estimation delay δ and state estimation error ϵ on control performance, we show a simple PID tracker controlling the motion of a point mass in the (x, y) plane. The position of the point mass must follow a reference constant trajectory, whose x dimension is shown in Fig. 2 (the same plot can be obtained for the y position). We simulate two cases of estimation (and therefore actuation) delay and error, where a larger delay value δ implies a smaller estimation error ϵ. As can be noted in Fig. 2, the effect of delay can be non-negligible. In this example, it can be seen that the increased delay causes the tracking performance to worsen. Running an estimation task with a fixed smaller delay but larger estimation error does not necessarily solve the problem of degraded performance, as can be seen in Fig. 2. Therefore, there is a need to rigorously quantify the trade-off between computation time and estimation error, then exploit that trade-off to achieve the best control performance under the problem constraints. Rather than always running the estimation task to completion, it is useful to have several delay/error run-time modes for the estimator. These can then be used at run-time to satisfy the control objectives.* □

The goal of this paper is to develop a rigorous framework for the co-design of the controller and estimation algorithms. In this framework, the estimator has a range of computation time/estimate quality operating modes, and in order to best maintain control performance and reduce energy consumption, the controller at run-time selects one of these modes for the estimator to operate in. This is motivated by the following observations:

1) The traditional engineering approach to account for the estimator's run-time is to gauge the Worst-Case Execution Time (WCET) of the estimation task, and design the system to meet deadlines under the WCET conditions. In practice however, the actual execution time of perception-based estimators can be much less than the WCET and depends on the actual data being processed. Hence, considering the WCET can lead to a conservative design of the system. Additionally, the classical timing analysis alone does not guarantee *functional* correctness of the closed-loop system under control.

2) Moreover, in the context of closed loop control, we do not always require the best quality state estimate: more often than not, a lower quality estimate, computed using lesser energy and time, is acceptable to achieve the control objectives.

3) In the case where obtaining a better quality state estimate requires longer computation time, it can be detrimental to the control performance to require a high quality state estimate all the time. For example, when the on-board computer is overloaded, there may be a need to spend less time computing a state estimate so that not only the control action has less delay, but also so that other processes can access the computation resource as scheduled.

In this paper, we develop the observations above into a *co-design framework* for a real-time control systems, where the controller and estimator are interfaced via *contracts*. A contract is an assurance requested by the controller, and provided by the estimator, that the latter can give an estimate with a certain accuracy ϵ, and within a predefined time deadline δ. The computation time given to the estimator, as well as the quality of the state estimate define the contract. This can be interpreted as turning the estimator into a discretized version of an *anytime algorithm* [3] where its computation can be interrupted at runtime to get a state estimate, usually with a trade-off between the computation time given to the algorithm and the quality of output that it returns. Through this notion of contracts, we show how the controller can vary the computation time of the estimation algorithm to maintain control performance and to reduce energy consumption. The work presented here is focused on estimation algorithms that rely on computationally intensive Computer Vision (CV) algorithms in order to get a state estimate of a dynamical system, e.g. those in autonomous robot navigation with visual (camera, Lidar) sensors. We refer to these as *perception-based estimators*. Through experiments, we show that the computation time of

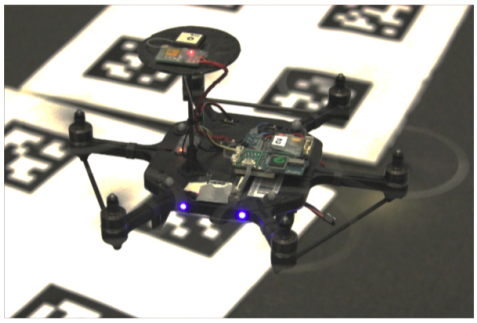

Fig. 3. Autonomous hexrotor with downward-facing camera flying over synthetic features.

such algorithms can be significant (and much greater than that of the control algorithm), resulting in an adverse impact on the closed loop control performance.

The architecture for the co-design framework proposed in this work is shown in Fig. 1. It resembles the conventional closed loop control architecture involving the estimator, the controller, and the system being controlled, but also incorporates the (delay, error) contract as an interface between the controller and the estimation algorithm.

Summary of contributions. In this paper, we build upon our results from [4] and present a framework for the co-design of control and estimation algorithms for the real-time control of dynamical systems. This approach consists of:

- a well-defined interface between control and estimation, in the form of operating modes, or *contracts*, on the accuracy and computation time of the estimator (Section III),
- characterizing the estimator accuracy as either deterministic (worst-case) or stochastic through offline profiling of the perception-based estimator (Section VII),
- a predictive control algorithm that can change the operating mode of the estimator at run-time to achieve control objectives at a lower energy cost (Section IV), while providing guarantees on satisfaction of constraints for both deterministic (Section V) and probabilistic (Section VI) characterizations of the estimation error, and,
- a straightforward, low-touch and low-effort approach to design a contract-driven estimation algorithm starting from an off-the-shelf, run-to-completion version of it (Section VII).
- We demonstrate our method on an autonomous flying robot (shown in Fig. 3) and show its performance and energy gains over a classical controller (Section VIII).

Compared to our previous work [4], which only allows for characterizing the contracts in terms of worst case estimation error, in this work we extend the framework to also allow for a probabilistic representation for estimation error. We also provide guarantees on satisfaction of constraints and recursive feasibility of the new control predictive algorithm resulting from this probabilistic setup. In addition, we also extend the experimental setup and incorporate a real-time implementation of the new control algorithm and evaluate our approaches with two sets of new experiments on a hex-rotor autonomous robot.

II. Related Work

Algorithms, that can be interrupted at any point at run-time and still return an acceptable solution, are called *Anytime Algorithms* [3]. Such algorithms generally return solutions with improving quality of output the longer they run for. A subset of these are Contract Algorithms [5] which can be interrupted only at a finite number of pre-agreed upon times. In this paper we design a Contract-driven perception-based state estimator, but significantly expand the notion of a contract to now include the *quality of the solution* (estimation error in our case) as well as the computation time.

Anytime algorithms have found particular importance in the field of graph search [6], evaluation of belief networks [7] and GPU architectures [8], [9]. With autonomous systems gaining popularity, computationally overloaded systems with real-time requirements are becoming the norm. This has generated interest in the development of anytime algorithms in the field of control theory, with Quevedo and Gupta [10], Bhattacharya and Balas [11], and Fontanelli et al. [12] exploring this line of research. Anytime algorithms have also found widespread use in the field of motion planning [13], [14], [15], [16].

The work presented in this paper contrasts considerably with these efforts as the assumption of anytime computation is not on the controller or planning side but on the perception-based state estimation component of the feedback control loop. The loop is closed by the control algorithm presented here that decides the contract for the anytime state estimator at run-time. Also differing from the works discussed above, which require instantaneous and perfect full state access for the controller, our control algorithm takes into account the computation time and the estimation error of the perception-based estimators that are common in autonomous systems. The recent work of Falanga et al. [17] also tackles the problem of co-designing the perception and the control, but does so as a joint optimization that takes into account both the perception and the control. Our work differs from this significantly as we introduce the notion of contracts to decouple the perception-based estimator's performance and the control optimization. Our method also explicitly incorporates the timing and the estimation performance of the perception-based estimator in the control design, and can be used for the off-the-shelf perception-based estimators (for an example, see section VII-C).

In the domain of real-time systems, Worst Case Analysis, along with Logical Execution Time semantics are used in [18] to imbue a controller with information of the timing characteristics of the closed loop implementation. On the other hand, our approach involves profiling the estimation algorithm in a direct manner to get timing and estimation error characteristics. While [18] involves formally verifying a given controller, we *design* a control algorithm that is correct by construction and takes advantage of delay/accuracy trade-offs in real-time. In the context of autonomous multi-rotor UAVs, the effect of increasing the computation time of task on the overall performance of the system has been analyzed in [19] by

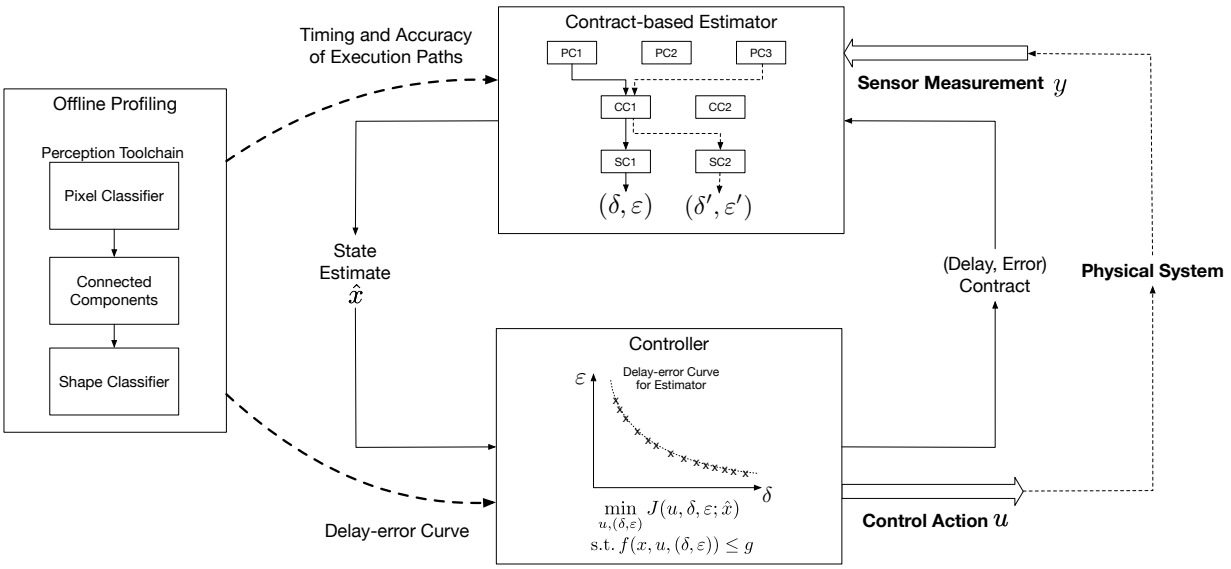

Fig. 4. Contract-driven estimator and controller. With knowledge of the estimator's performance through offline profiling, the controller both actuates the dynamical system and sets *contracts* for the estimator at run-time in order to maximize control performance while guaranteeing that constraints on the system are always satisfied.

using a resource allocation algorithm similar to QRAM [20]. Our approach contrasts with this as we focus on the execution time of a particular task, the perception-based state estimator, which directly impacts the closed loop control performance. In addition to this, we also formulate a controller that provides mathematical guarantees on the system's performance.

Finally, in the area of computer architecture, approximate computing approaches [21], [22], [23] have been explored to get savings in time or energy through performing a computation in an approximate manner, rather than precisely. While anytime algorithms and approximate computing have a common high-level goal, approximate computing methods are run-to-completion and lack a feedback mechanism to permit computation and resources to be balanced dynamically. It is also worth noting that time and energy scale that our approach deals with are much greater than those which concern approximate computing.

III. Co-design of estimation and control

Conventional closed loop control systems are generally designed in a manner where the controller is incognizant of the implementation details of the state estimation module, while the estimation module is designed independent of the requirements of the controller. For example, a feedback controller, that gets state estimates from a camera based visual odometry algorithm, might not be designed to take into account the non-negligible time taken to process the video frames to get a state estimate. We refer to this computation time as the estimation delay. On the other hand, the design of most perception-based estimators does not take into account the varying real-time constraints that the controlled closed-loop system must satisfy. Also of importance, especially in autonomous systems deployed in the field, is the power consumed by the computation platform which can have a significant impact on the duration the system can operate between charging.

Taking these factors into account, we propose the *co-design* of estimation and control to improve the closed loop performance of real-time control on systems with computationally and power limited platforms. This is done through a *contract-driven framework* for both estimator and controller in which the controller asks for a state estimate within a certain deadline δ seconds, with an associated bound on the inaccuracy of the estimation. This inaccuracy can either be in the form of a hard bound ϵ, e.g. an infinite-norm bound on the estimation error vector, or have a probabilistic characterization Σ, e.g. the covariance of the estimation error vector, depending on the application. For the sake of simplicity, we use ϵ for the characterization of the estimation error in the following text.

In our framework, the tuple (δ, ϵ) forms the *contract* between controller and estimator. The estimator is tasked with providing a state estimate that respects the contract. Aware of these contracts, the controller can set the appropriate contract in a time varying manner to adapt the closed-loop system performance in real-time to take into account the control requirements of the physical system. For example, it can decide when an estimate is needed fast (but usually with higher error), and when a more accurate estimate is needed (but with greater delay). Note, the (δ, ϵ) contract can also be thought of as setting an *operating mode* for the perception-based estimator. A high-level view of this setup is shown in Fig. 1.

In order to make sure that the contracts are such that the estimator can indeed fulfill them, the estimator is profiled offline. To do this, the estimator's internal parameters are varied, and for each parameter setting, it is run on a *profiling data set* (with a known ground-truth baseline). This results in a set of (δ, ϵ) values, each one corresponding to a particular setting of

the parameters. These values can be plotted on a curve, which we call the *error-delay curve* made up of discrete points, (δ, ϵ), Examples of such a curve are shown in Figs. 7 and 9. Section VII provides the detailed procedure for obtaining this curve for a perception-based estimator.

During run-time execution, upon receiving a (δ, ϵ) contract request from the controller, the estimator can adapt its parameter settings to fulfil the contract, i.e. to provide a state estimate within the requested deadline δ that also respects the requested error bound ϵ.

The controller, in the co-design framework, is designed with the awareness of the error-delay curve of the estimation algorithm, and requests contracts from that curve. The error-delay curve, thus constitutes the *interface* between the controller and state estimator. The controller leverages the flexible nature of the estimation algorithm to maximize some measure of control performance.

The closed loop architecture in a system with co-design of the estimator and controller is shown in Fig. 4. In this co-designed system, the controller can make the estimation algorithm switch to lower or higher time (and/or energy) consuming modes based on the control objective at the current time step. The main components of the co-design architecture presented in this paper are: a) a contract perception-based estimator, b) a robust control algorithm that computes an input to be sent to the physical system being controlled as well as the contract for the estimator, and c) the interface between them. More details on these components are in the following sections.

IV. Control with Contract-driven Estimation

In this section, we formalize how the error-delay curve of the estimator can be utilized by the control algorithm to optimize the control performance while minimizing the power consumed by the computations for the perception-based estimator.

A. System Model

In order to model the co-design process, consider the closed-loop control of an autonomous hex-rotor robot (more details in VIII), shown in Fig. 3. The state x of the hexrotor consists of its 3D position and 3D velocity, while the input u to the robot consists of the desired pitch and roll angles, and the desired thrust. The hexrotor's task is to fly a pre-defined trajectory given by x_{ref}, where $x_{ref}(t)$ gives the desired position at each time t. The dynamics of the hexrotor, relating the time-evolution of its state to the current state and input, can be linearized around hover and approximated by the following Linear Time-Invariant (LTI) ODE:

$$\dot{x}(t) = A_c x(t) + B_c u(t) + w_c(t) \tag{1}$$

Here, the state vector $x \in \mathbb{R}^n$ is constrained to be within set $X \subset \mathbb{R}^n$, the control input $u \in \mathbb{R}^m$ is constrained within set $U \subset \mathbb{R}^m$, and $w_c \in \mathbb{R}^n$ is the process noise assumed to lie in a (bounded) set $\mathcal{W}_c \subset \mathbb{R}^n$. $A_c \in \mathbb{R}^{n \times n}$ and $B_c \in \mathbb{R}^{n \times m}$ are matrices. LTI ODEs can model a wide range of systems, and our results apply to arbitrary LTI systems of the form given in (1) with compact and convex constraint sets X, U and $\mathcal{W}_c$. The sets X and U are determined by the control designer or by physical constraints on the system. For example, X captures limits on the state to define the region which the hexrotor can fly and the velocity limits on it. The set U restricts the inputs to values that can be supported by the rotors, as well as within which the linearized system provides a good approximation to the true nonlinear dynamics.

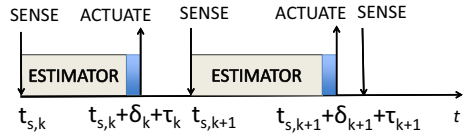

Fig. 5. Time-triggered sensing and actuation. The figure shows the varying execution time for the estimator and the blue area shows the execution time for the controller, which is small.

B. Time-Triggered Sensing and Actuation

For feedback control of the hexrotor, the controller needs to be aware of the hexrotor's current position and speed, i.e. requires an *estimate* of its current state x. This is done via a perception-based estimator, that process video frames (at a fixed rate) obtained through a downward facing camera mounted on the hexrotor. The estimator detects and tracks features across frames, and deduces its own position through the relative motion of these features.

A new frame is captured by the camera every $T > 0$ seconds, which results in periodic measurements at instants $t_{s,k} = kT$, where $k \in \mathbb{N}$. This measurement is used by the estimator to compute the state estimate $\hat{x}_k := \hat{x}(t_{s,k})$ with the desired accuracy ϵ_k *determined by the contract set by the controller in the previous time step*. The controller then acts on this state estimate to compute the control input u_k as well as decide on the perception-based estimator's delay and accuracy contract $(\delta_{k+1}, \epsilon_{k+1})$ for the next time step. The control is then applied to the physical system according to (1) at instant $t_{a,k} = t_{s,k} + \delta_k + \tau_k$, where τ_k is the time it takes to compute the input. See Fig. 5 for the timing diagram of this process.

The controller has access to the delay-error curve, or operating modes Δ of the estimator, and at each time step selects contracts *from that curve*. This curve is obtained offline as explained in Section III, and illustrated in Section VII. Note that at each step $k \geq 0$, the estimation accuracy ϵ_k, and hence the delay δ_k are already decided in the previous time step and known to the controller. For the very first step $k = 0$, the initial estimation mode δ_0, ϵ_0, as well as the the initial control input u_{-1} are chosen by the designer.

C. Control Performance

The controller has a goal that is twofold: it needs to ensure that the reference trajectory is tracked as closely as possible, and that the computation energy consumed to do so is minimized. To capture this, we define two (stage) cost functions: first, $\ell(x, u) = (x - x_{ref})^T Q (x - x_{ref}) + u^T R u$ defines a

weighted sum of the tracking error (first summand) and the input power (second summand). Here, Q and R are positive semidefinite and positive definite matrices respectively. Second, $\pi(\delta)$ captures the average power consumed to perform a perception-based estimation computation duration δ. This power information is collected offline during the estimator profiling phase.

The total cost function for the controller to minimize is $J = \sum_{k=0}^{M} (\ell(x_k, u_k) + \alpha\pi(\delta_k))$, where $M \geq 0$ is the duration of the system's operation.

D. Discretized Dynamics

Due to the time-triggered sensing and actuation of the system (see Sec. IV-B), from time $t_{s,k}$ to $t_{a,k}$, the previous control input u_{k-1} is still being applied. Then at $t_{a,k}$ the new control input u_k is computed and applied by the controller (see Fig. 5). For the sake of simplicity, we assume the computation time for the controller (τ) is small and constant, and so lump it with the time for the estimator (δ). This is justified experimentally for our problem (in Sec.VIII) where the time for the controller is negligible compared to the time taken by the estimation algorithm. The discrete time dynamics for this setup, with a periodic sensing time of T, are given by

$$x_{k+1} = Ax_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + w_k, k \geq 0 \quad (2)$$

in which

$$A = \mathrm{e}^{A_c T}, \quad w_k = \int_0^T \mathrm{e}^{A_c(T-t)} w_c(t_{s,k} + t)dt$$
$$B_1(\delta) = \int_0^\delta \mathrm{e}^{A_c(T-t)} B_c dt,\ B_2(\delta) = \int_\delta^T \mathrm{e}^{A_c(T-t)} B_c dt.$$

Here, w_k is the process noise accumulated during the interval. It is constrained to lie in a compact convex set $\mathcal{W}$ since $w_c(t)$ lies in the compact convex set $\mathcal{W}_c$ and T is finite. As explained above, both the current control u_k and the previous control u_{k-1} appear in (2). In addition, the input matrices $B_1(\delta_k)$ and $B_2(\delta_k)$ depend on the delay δ_k. The estimation accuracy ϵ_k, indirectly affects the dynamics via the control input, which is computed using the state estimate $\hat{x}_k$. These discrete time dynamics therefore show how the operation mode of the estimator (δ, ϵ) affects the dynamics of the system.

V. Robust Model Predictive Control Solution

In this section we give an overview of the *Robust Adaptive Model Predictive Controller* (RAMPC) that we use in the contract-driven setup of Fig. 4. Here, we consider the estimation errors to be bounded, and use these worst-case bounds in the controller formulation. The mathematical details and derivations are available in the appendix. Experiments confirm that the following controller can be run in real-time, and its computation uses a negligible amount of time relative to the estimation delay.

A. Solution overview

Recall the operation of the contract-driven control and estimation framework as presented in Section III and Fig. 4. First, the estimator is profiled offline to obtain its delay-error curve, which we denote by Δ. The curve Δ represents a finite number of (δ, ϵ) contracts that the estimator can satisfy. At every time step k, the controller receives a state estimate $\hat{x}_k$ and uses it to compute the control input u_k to be applied to the physical system at time $t_{a,k}$ and the contract $(\delta_{k+1}, \epsilon_{k+1}) \in \Delta$ that will be requested from the estimator at the next step. At $k+1$, the estimator provides an estimate with error at most ϵ_{k+1} and within delay δ_{k+1}. Finally, recall that $J = \sum_{k=0}^{M} (\ell(x_k, u_k) + \alpha\pi(\delta_k))$ combines tracking error and input power in the ℓ terms, and estimation power consumption in the π terms. The scalar α quantifies the importance of power consumption to the overall performance of the system.

The contract-driven controller's task is to find a sequence of inputs $u_k \in U$ and of contracts $(\delta_k, \epsilon_k) \in \Delta$ such that the cost J is minimized, and the state x_k is always in the set X. The challenge in finding the control inputs is that the controller does not have access to the real state x_k, but only to an estimate $\hat{x}_k$. The norm of the error $e_k = \hat{x}_k - x_k$ is bounded by the contractual ϵ_k, which varies at each time step.

Let us fix the *prediction horizon* $N \geq 1$. Assume that the current contract (under which the current estimate $\hat{x}_k$ was obtained) is (δ_k, ϵ_k), and that the previously applied input is u_{k-1}. To compute the new input value u_k and next contract $(\delta_{k+1}, \epsilon_{k+1})$, the proposed **Robust Adaptive Model Predictive Controller (RAMPC)** seeks to solve the following optimization problem which we denote by $\mathbb{P}_\Delta(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$:

$$\begin{aligned} J^*[0\!:\!N] = \min_{\mathbf{u},\mathbf{x},\underline{\delta},\underline{\epsilon}} & \sum_{j=0}^{N} (\ell(x_{k+j}, u_{k+j}) + \alpha\pi(\delta_k)) \quad (3) \\ \text{s.t.} \quad & \forall j \in \{0, \ldots, N\} \\ & x_{k+j+1} = Ax_{k+j} + B_1(\delta_k)u_{k+j-1} + B_2(\delta_k)u_{k+j} \\ & [x_{k+j+1},\ u_{k+j}]' \in X \times U \end{aligned}$$

Here, RAMPC needs to find the optimal length-N input sequence $\mathbf{u}^* = (u_k^*, \ldots, u_{k+N}^*) \in U^N$, corresponding state sequence $\mathbf{x} = (x_k, \ldots, x_{k+N}) \in X^N$, delay sequence $\underline{\delta} = (\delta_k, \ldots, \delta_{k+N})$ and error sequence $\underline{\epsilon} = (\epsilon_k, \ldots, \epsilon_{k+N})$ such that $(\delta_k, \epsilon_k) \in \Delta$, which minimize the N-step cost $J[0:N]$. The matrices that make up the system dynamics are defined in Section IV-D. As in regular MPC [24], once a solution $\mathbf{u}^*$ is found, only the *first* input value u_k^* is applied to the physical system, thus yielding the next state x_{k+1} as per (2). At the next time step $k+1$, RAMPC sets up the new optimization $\mathbb{P}_\Delta(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_{k+1-1})$ and solves it again.

To make this problem tractable, we first assume that the mode is fixed throughout the N-step horizon, i.e. $(\delta_{k+j}, \epsilon_{k+j}) = (\delta, \epsilon)$ for all $1 \leq j \leq N$. Thus for every value (δ, ϵ) in Δ, we can setup a different problem (3) and solve it. Let $J^*_{(\delta,\epsilon)}$ be the corresponding optimum. The solution with the smallest objective function value yields the input value u_k^* to be applied and the next contract (δ^*, ϵ^*).

Because RAMPC only has access to the state estimate, we extend the RMPC approach in [25], [26]. Namely, the problem is solved for the *nominal dynamics* which assume zero process and observation noise ($w_{k+j} = 0$) and zero estimation error ($\hat{x}_{k+j} = x_{k+j}$) over the prediction horizon.

Let $\overline{x}$ be the state of the system under nominal conditions. To compensate for the use of nominal dynamics, RMPC replaces the constraint $(\overline{x}_{k+j}, u_{k-1+j}) \in X \times U := \mathcal{Z}$ by $(\overline{x}_{k+j}, u_{k+j}) \in \mathcal{Z}_j(\epsilon_k, \epsilon)$, where $\mathcal{Z}_j(\epsilon_k, \epsilon) \subset \mathcal{Z}$ is $\mathcal{Z}$ 'shrunk' by an amount corresponding to ϵ, as explained in the appendix. Intuitively, by forcing $(\overline{x}_{k+j}, u_{k-1+j})$ to lie in the reduced set $\mathcal{Z}_j(\epsilon_k, \epsilon)$, the bounded estimation error and process noise are guaranteed not to cause the true state and input to exit the constraint sets X and U. The tractable optimization for a given (δ, ϵ), denoted by $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$, is then

$$
\begin{aligned}
J^*_{(\delta,\epsilon)} = \min_{\mathbf{u},\mathbf{x}} & \sum_{j=0}^{N} \left(\ell(\overline{x}_{k+j}, u_{k+j}) + \alpha\pi(\delta)\right) \\
\text{s.t.} \quad & \forall j \in \{0, \ldots, N\} \\
& \overline{x}_{k+j+1} = A\overline{x}_{k+j} + B_1(\delta)u_{k+j-1} + B_2(\delta)u_{k+j} \\
& (\overline{x}_{k+j}, u_{k+j}) \in \mathcal{Z}_j(\epsilon_k, \epsilon)
\end{aligned} \tag{4}
$$

Algorithm 1 summarizes the RAMPC algorithm.

Algorithm 1 Robust Adaptive MPC algorithm with Anytime Estimation.

1: (δ_0, ϵ_0) and u_{-1} specified by designer
2: Apply u_{-1}
3: **for** $k = 0, 1, \ldots, M$ **do**
4: Estimate $\hat{x}_k$ with guarantee (δ_k, ϵ_k)
5: **for** each $(\delta, \epsilon) \in \Delta$ **do**
6: $(u^*_k, J^*_{(\delta,\epsilon)}) \leftarrow$ Solve $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$
7: **end for**
8: $(\delta^*, \epsilon^*, u^*_k) \leftarrow \text{argmin}_{(\delta,\epsilon)} J^*_{(\delta,\epsilon)}$
9: Apply control input $u_k = u^*_k$ and estimation mode $(\delta_{k+1}, \epsilon_{k+1}) = (\delta^*, \epsilon^*)$
10: **end for**

We prove the following result in the appendix:

Theorem V.1. *If at the initial time step there exists a contract value $(\delta, \epsilon) \in \Delta$, an initial state estimate $\hat{x}_0 \in X$, and an input value $u_{-1} \in U$, such that $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_0, \delta_0, \epsilon_0, u_{0-1})$ is feasible then the system (2) controlled by Alg. 1 and subjected to disturbances constrained by $w_k \in \mathcal{W}$ robustly satisfies the state constraint $x \in X$ and the control input constraint $u \in U$, and all subsequent iterations of the algorithm are feasible.*

VI. Stochastic Model Predictive Control Solution

The control algorithm developed in section V assumes that the state-estimation error e lies in a bounded set, E. In practice, this can result in a very conservative approximation. Assuming instead that the error arises from a random distribution allows us to develop a *chance constrained* formulation for the controller, outlined in this section. We call this control algorithm the *Stochastic Adaptive Model Predictive Controller* (SAMPC). Here, the constraints on the state have to be satisfied with some probability $1-\zeta$, rather than in a deterministic manner as in the RAMPC formulation.

A. Solution overview

Starting from the contract-driven control and estimation framework of Sec. III, we denote the profiled delay-error curve of the estimator by Δ. This curve Δ consists of a finite number of contract options (δ, Σ) that the estimator can satisfy at run-time. Here, $\Sigma \in \mathbb{R}^{n\times n}$ is the positive semi-definite co-variance matrix associated with the now stochastic state-estimation error e. It can be obtained through profiling the performance of the estimator as outlined in Sec. VII. We assume that the mean of the estimation errors is zero in all the contracts, but the formulation and analysis that follows also extends to distributions with non-zero means. δ is again the computation time the estimator takes in a particular mode of operation.

The SAMPC works in a manner similar to the RAMPC. At each time step k, the controller receives a state estimate $\hat{x}_k$ and uses it to compute: a) the control signal u_k, as well as b) the contract $(\delta_{k+1}, \Sigma_{k+1}) \in \Delta$ that will be met by the estimator at the following time step. Following this, at time step $k+1$ the estimator give a state estimate $\hat{x}_{k+1}$ with error $e_{k+1} = \hat{x}_{k+1} - x_{k+1}$ drawn from a distribution with co-variance Σ_{k+1} and within time δ_{k+1}.

The cost function to be minimized is $J = \sum_{k=0}^{M}(l(x_k, u_k) + \alpha\pi(\delta_k))$ that combines the tracking error and input power through the l term and the estimator power consumption through the π terms. The SAMPC control algorithm then finds a sequence of control signals u_k and the contracts at each time step $(\delta_k, \Sigma_k) \in \Delta$ such that J is minimized and the state x_k and input u_k respect chance constraints of the form:

$$
P([x_k, u_k] \in X \times U) \geq 1 - \zeta \, \forall k \tag{5}
$$

Here, $0 < \zeta \leq 1$ is a design parameter that decides the lower bound on the constraint satisfaction probability. To achieve these objectives, the **Stochastic Adaptive Model Predictive Controller (SAMPC)** aims to solve the following optimization (with horizon $N \geq 1$), denoted by $\widetilde{\mathbb{P}}_{\Delta}(\hat{x}_k, \delta_k, \Sigma_k, u_{k-1})$, at each time step k:

$$
\begin{aligned}
J^*[0:N] \quad = \quad & \min_{\mathbf{u},\mathbf{x},\underline{\delta},\underline{\Sigma}} \sum_{j=0}^{N} \left(\ell(x_{k+j}, u_{k+j}) + \alpha\pi(\delta_k)\right) \\
\text{s.t.} \quad & \forall j \in \{0, \ldots, N\} \\
& x_{k+j+1} = Ax_{k+j} + B_1(\delta_k)u_{k+j-1} + B_2(\delta_k)u_{k+j} \\
& P([x_k, u_k] \in X \times U) \geq 1 - \zeta
\end{aligned} \tag{6}
$$

Similar to the RAMPC, the SAMPC needs to find the optimal length-N input sequence $\mathbf{u} = (u^*_k, \ldots, u^*_{k+N})$, the corresponding state sequence $\mathbf{x} = (x_{k+1}, \ldots, x_{k+N+1})$, the delay sequence $\underline{\delta} = (\delta_k, \ldots, \delta_{k+N})$ and associated error co-variance sequence $\underline{\Sigma} = (\Sigma_k, \ldots, \Sigma_{k+1})$ (such that $(\delta_k, \Sigma_k) \in \Delta$) which minimize the the N-step cost $J[0:N]$ and ensuring the chance constraint of (5) is satisfied.

Consistent with regular MPC framework, once a solution $\mathbf{u}$ is found, only the first input u_k is applied to the system, resulting in state x_{k+1}. At the next time step, after receiving

the state estimate $\hat{x}_{k+1}$ from the estimator based on the contract of step k, the SAMPC sets up the new optimization $\widetilde{\mathbb{P}}_{\Delta}(\hat{x}_{k+1}, \delta_{k+1}, \Sigma_{k+1}, u_{k+1-1})$ and solves it, repeating the process at each subsequent time step.

Similar to RAMPC, the SAMPC only has access to the state estimate, we extend the Stochastic MPC (SMPC) approach in [27]. Namely, the problem is solved for the *nominal dynamics* which assume zero process and observation noise ($w_{k+j} = 0$) and zero estimation error ($\hat{x}_{k+j} = x_{k+j}$) over the prediction horizon. Let $\overline{x}$ be the state of the system under nominal conditions. To compensate for the use of nominal dynamics, SMPC replaces the constraint of (5) by $(\overline{x}_{k+j}, u_{k+j}) \in \widetilde{\mathcal{Z}}_j(\Sigma_k, \Sigma)$, where $\widetilde{\mathcal{Z}}_j(\Sigma_k, \Sigma) \subset \mathcal{Z}$ is $\mathcal{Z} = X \times U$ 'shrunk' by an amount corresponding to Σ, as explained in the appendix.Intuitively, by forcing $(\overline{x}_{k+j}, u_{k-1+j})$ to lie in the reduced set $\widetilde{\mathcal{Z}}_j(\Sigma_k, \Sigma)$, the stochastic estimation error and process noise are guaranteed to be such that that the state and the input respect the joint chance constraint of (5).

The tractable optimization for a given (δ, Σ), denoted by $\widetilde{\mathbb{P}}_{(\delta,\Sigma)}(\hat{x}_k, \delta_k, \Sigma_k, u_{k-1})$, is then

$$
\begin{aligned}
J^*_{(\delta,\Sigma)} \quad &= \quad \min_{\mathbf{u},\mathbf{x}} \sum_{j=0}^{N} \left(\ell(\overline{x}_{k+j}, u_{k+j}) + \alpha\pi(\delta_k)\right) \qquad (7)\\
\text{s.t.} \quad & \quad \forall j \in \{0, \ldots, N\} \\
& \quad \overline{x}_{k+j+1} = A\overline{x}_{k+j} + B_1(\delta_k)u_{k+j-1} + B_2(\delta_k)u_{k+j} \\
& \quad (\overline{x}_{k+j}, u_{k+j}) \in \widetilde{\mathcal{Z}}_j(\Sigma_k, \Sigma)
\end{aligned}
$$

Construction of the shrunk constraint sets $\widetilde{\mathcal{Z}}_j$ is covered in the appendix. In practice, we solve the optimization for each $(\delta, \Sigma) \in \Delta$ in parallel and pick the optimal contract and the corresponding control signal as outlined in Algorithm 1 (solving $J^*_{(\delta,\Sigma)}$ instead of $J^*_{(\delta,\epsilon)}$ in this case). The following theorem (proven in the appendix) states the guarantees of this control algorithm:

Theorem VI.1. *For any estimation mode (δ, Σ), if $\widetilde{\mathbb{P}}_{(\delta,\Sigma)}(\hat{x}_k, \delta_k, \Sigma_k, u_{k-1})$ is feasible then the system* (2) *controlled by the SAMPC and subjected to disturbances constrained by $w_k \in \mathcal{W}$ satisfies, with probability at least $1 - \zeta$, the state constraint $x_k \in X$ and control input constraint $u_k \in U$, and the subsequent optimization $\widetilde{\mathbb{P}}_{(\delta,\Sigma)}(\hat{x}_(, \delta_(, \Sigma_(, u_{(-1)})\hat{x}_{k+1}, \delta[k], \Sigma[k], u_k)$, are feasible with Probability* 1.

VII. Contract based perception algorithms

We presupposed, in Section III, the existence of an Estimation Error vs Computation Delay curve Δ for the state estimator. The controller uses this curve at each discrete time step to select the operating mode (δ, ϵ) for the estimator at the next time step, as seen in Sec. V. In this section, we show how this curve can be obtained for particular applications, as well as ways for the contract based estimation algorithm to realize the points on the curve at runtime.

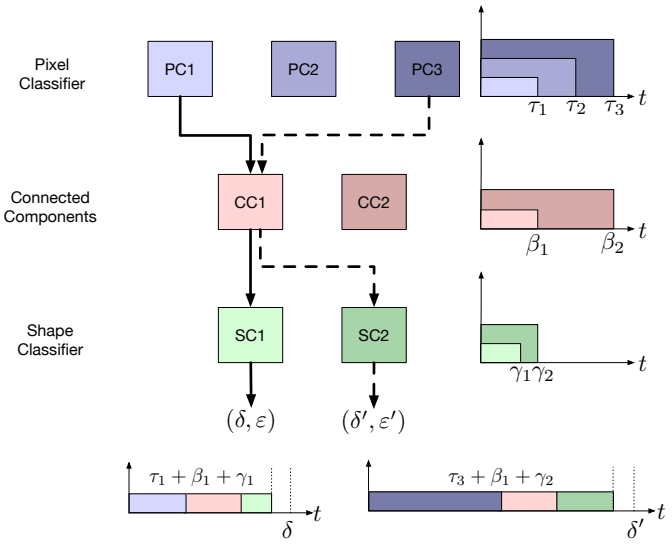

Fig. 6. Illustration of the building blocks used to compose the Contract Object Detector and their representation as real-time tasks. For a given (δ, ϵ) contract, knob settings are chosen at run-time resulting in a schedule to execute these sequential components, or tasks, to respect the contract.

A. Profiling And Creating an Anytime Contract Based Perception-and-Estimation Algorithm

In order to profile a contract estimator, we first need to identify the distinct building blocks (or tasks) of the perception algorithm. Next, we need to find the relevant parameters used in each task, such that varying these parameters results in corresponding changes in the computation time and the quality of the overall output of the estimation algorithm. This can be done, e.g. by varying the number of iterations of a loop [21] such that the resulting computation time δ and estimation quality ϵ are different. We refer to these parameters as *knobs* of the components of the estimation algorithm.

This procedure is tested through implementation on a Computer Vision (CV)-based object detection tool chain, an overview of which is shown in Fig. 6. This object recognition tool chain is tasked with tracking an Object of Interest (OOI) across the frames of a video stream. The first level of this is a pixel classifier that assigns a probability for each pixel being a part of the OOI. After thresholding over some minimum probability, a we obtain a binary image with the pixels of interest taking a value 1, others being 0. The second level involves denoising the binary image, and then finding the Connected Components (CC), i.e. collecting adjacent pixels of interest into (possibly disconnected) objects. The third and final level is a shape classifier that is run on the output of the connected components to determine whether each object from it is of interest or not.

Our implementation uses a Gaussian Mixture Model (GMM) classifier as the pixel classifier. The *knob* here is the number of Gaussian distributions in the GMM. A smaller number of Gaussians will result in a faster, but possibly inaccurate classifier. On the other hand, more Gaussians can result in improved performance, but at the cost of higher

computation time. As is typically done, knob values that result in an overfit are identified and rejected via cross-validation during the training process.

The filtering for denoising the binary image, and the Connected Components algorithm form the second level of the object recognition tool chain and the knob here consists of selecting either a 4-connected or 8-connected implementation.

We use a GMM for the shape classifier, but unlike the first level, the knob here is the number of features used to define the shape of the object of interest (e.g.eccentricity, linear eccentricity and major and minor axis lengths for ellipsoidal objects). In this implementation, the number of knob settings for the object recognition tool chain is K = (#Gaussians for pixel classifier $\times$ #neighbors for CC $\times$ #features for shape classifier), and has a total of $3 \times 2 \times 2 = 12$ values.

The trade-off curve for the entire toolchain is obtained by profiling all 12 knob settings by running it on a data set for profiling. Through this process we obtain, for each of the different knob values: a) the output quality error ϵ, and b) the computation times δ for the entire tool chain. This offline gathering of information gives us the information to be used at run-time in the co-design framework. The profiled performance of the CV-based object recognition toolchain considered here is shown in Fig. 7.

It should be noted that for each block of the tool chain, the relation between knob value and quality of output is not necessarily monotonic. The GMM based classifiers must be trained on a data set before deployment and like all machine learning algorithms, their output quality for a given knob setting will depend on the specific data set. This also holds for the output quality of the entire chain, and is reflected in Fig. 7 which shows the mean perception error[1] and the 90^{th} percentile execution time for the different knob settings. While the trend is that perception error decreases with increasing execution time, there are some knob settings leading to both larger perception error and larger execution time, which is seen in the non-monotonic behavior seen in Fig. 7.

B. Run-time execution of the contract-driven perception algorithm

After profiling the contract-driven estimator, we can use the information at run-time to choose which knob settings are needed to respect a given (δ, ϵ) contract. This is tantamount to choosing altered versions of tasks and scheduling them to execute one after the other in a pre-defined manner to optimally perform the job of detecting an object of interest. Fig. 6 shows the various tasks and their different versions for every knob setting and the resulting task schedules.

C. Visual Odometry

An example of a vision based state estimation algorithm is Semi-Direct Monocular Visual Odometry (SVO) [1], which we will use in Sec. VIII to get state estimates for control of the hexrotor robot. SVO detects *corners* in an image, and tracks them across consecutive frames of a video feed in order to localize the moving robot and generate a state estimate. Since this state estimate is used for closed loop control of the hexrotor, SVO has to run in real-time at a frame rate that is fast enough for the purpose of controlling a flying robot. The number of corners $\#C$ (as well as their quality) being tracked from frame to frame affects the computation time of the localization algorithm and the resulting quality of the state estimate. In general, assuming that the camera is looking at a feature rich environment, detecting and tracking a higher number of corners results in better localization accuracy but also takes larger computation time. For the profiling of SVO, the number of corners $\#C$ is the only knob and is varied to obtain an error-delay curve of the localization performance.

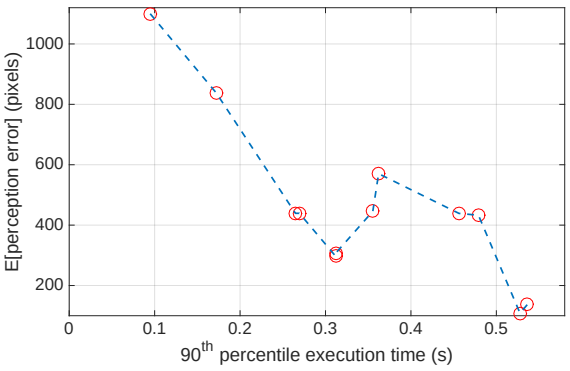

Fig. 7. Profiled delay-error curve for the object detection tool chain run at different parameter settings.

1) Profiling SVO performance: Fig. 8 outlines the profiling process for SVO. We start with the hexrotor, running ROS, flying (either manually or autonomously) in an environment with a *Vicon* motion-capture system [28]. Throughout the flight, the downward facing monocular camera captures frames at the desired rate of 20 HZ. We also log the IMU data, as well as the high-accuracy 6-DOF pose estimate generated by the motion capture system, which we will use as the ground truth for the hexrotor positions and velocities. We collected data, recorded as rosbags, over 15 minutes of flights with randomly chosen paths, flown both manually and autonomously.

After collecting the data from our flights, in order to profile the estimation performance of SVO for a particular setting of the number of corners used, we playback this recorded rosbag, accurately recreating the in-flight environment that is present for the visual odometry algorithm. We process the camera frames with SVO running at the desired setting of $\#C$, and use the SVO generated position estimate along with the corresponding time-stamped IMU data to generate an estimate of the hexrotor's position and velocity (the hexrotor's state, see fig. 10) at that time instant. By comparing this the state estimate to the VICON measurement at that time instant, we get the state estimation error of SVO. By doing so for the entire recorded data set, we can get the estimation error characteristics of SVO operating at this knob setting. We repeat this process for all knob settings of SVO, going from 50 to 350

[1]Error is the distance between the true centroid and the estimated centroid of the OOI

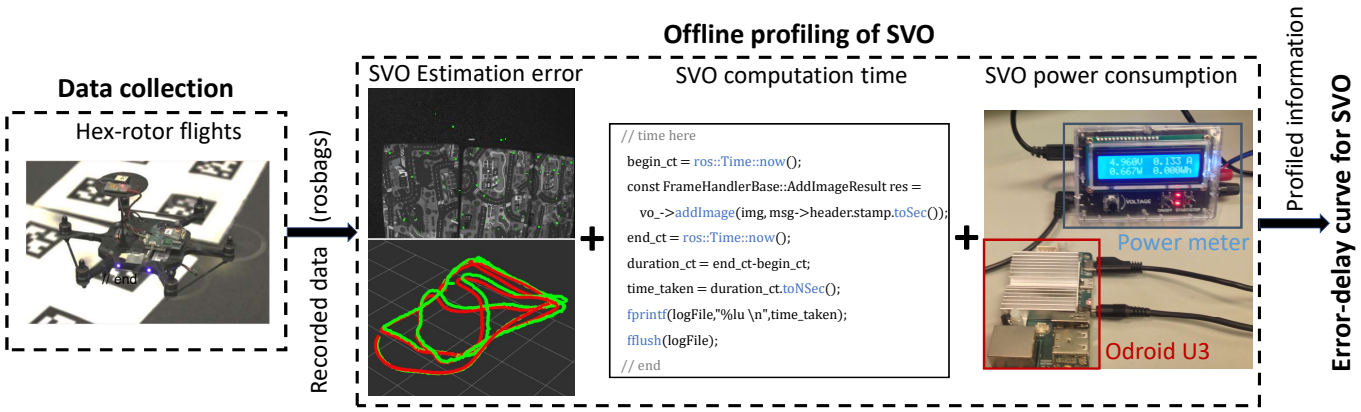

Fig. 8. The profiling process to characterize the performance of SVO in terms of estimation error, computation time and power consumption. Sensor and ground truth data is logged from flights of the hexrotor, and then played back and processed offline to generate the error-delay curve (shown in Fig. 9) for SVO. The code snippet shows how little modification is needed to the SVO code base to be able to profile its timing characteristics. Through this offline profiling process, we avoid the need of performing separate flights for each knob setting of SVO.

corners, and through this get the estimation error profile for SVO across all its operating modes. Fig. 8 shows an overlay of the position estimates from SVO (in green) and those from VICON (in red) for a segment of the profiling data set. It also shows a frame captured from the downward facing camera on the hexrotor, and the corners (green dots) that SVO is tracking in that particular frame.

We also need to measure the timing and power consumption of SVO for each knob setting. For the former, we insert C++ code for timing how long SVO takes to process each frame, i.e. the time from receiving a frame from the camera to generating a position estimate. We do this for each frame in the profiling data set, log this data, and repeat the process for each knob setting of SVO. Fig. 17 shows the cumulative distribution function for this computation time across the entire profiling data set, for each knob setting of SVO.

For the power consumption of SVO at different values of the knob $\#C$, we record power measurements made using the Odroid Smart Power meter [29], which measures consumption to milliwatt precision. By playing back the logged data and running SVO offline for profiling, we avoid the physical challenges of fitting the power meter onto our hexrotor platform and can measure the power consumption of the Odroid board on the ground, while running the workloads as it does during flight. We measure the power consumption of the entire Odroid board, including CPU and DRAM power consumption. Since the profiling of power is done offline with other peripherals plugged into the odroid (e.g. a monitor and keyboard), we measure the idle power of the Odroid and subtract that from the power measurements when the SVO algorithm is running on it in different modes. This gives us a more accurate measure of the workload due to the visual odometry task.

Through this offline profiling process, we avoid having to fly separate flights to get profiling information for every knob setting ($\#C$), and the result of this profiling is used in the formulation of the controller and used at run-time by it to generate contracts for the contract-driven estimator (Fig.4).

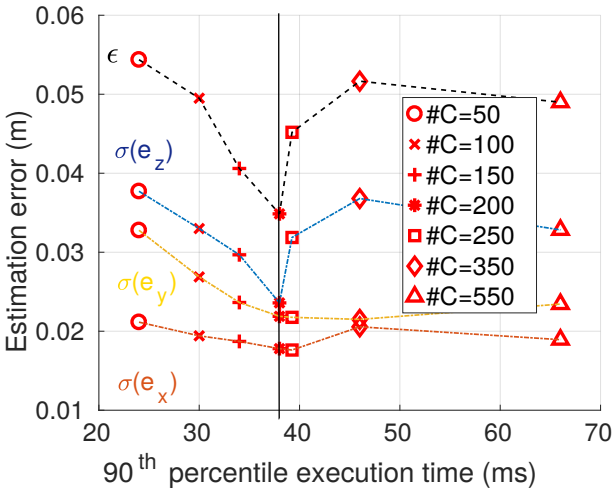

Fig. 9. (Color online) Error-delay curve for the SVO algorithm running on the Odroid-U3 with different settings of maximum number of features ($\#C$) to detect and track. The vertical line shows the cut-off for maximum delay and the SVO settings that are allowable (upto $\#C = 200$) for closed loop control of a hexrotor at 20Hz. No value of $\#C$ is used above this as it results in the delay approaching the sampling period of the controller.

2) The error-delay curve for SVO: Obtained from the profiling process outlined above, Fig. 9 shows the error-delay curve(s) of the localization error (in positions) of the hexrotor with SVO running on an Odroid-U3 [30], the on-board computation platform of the hexrotor robot. The curve, obtained through data collected over multiple flights in a fixed environment, shows the worst case error ϵ (over all flights and all components of the 3D position, used in Sec. V), as well the the standard deviation of the error for all components of the 3D position (used in the stochastic control formulation of Sec. VI) versus the computation time δ for varying number of corners being tracked $\#C$. δ is obtained by considering the 90^{th} percentile of computation times, while ϵ is obtained by computing the infinite norm of the 90^{th} percentile error over the 3 components (x, y, z) of the position. Note that as the number of corners being tracked increases, the computation

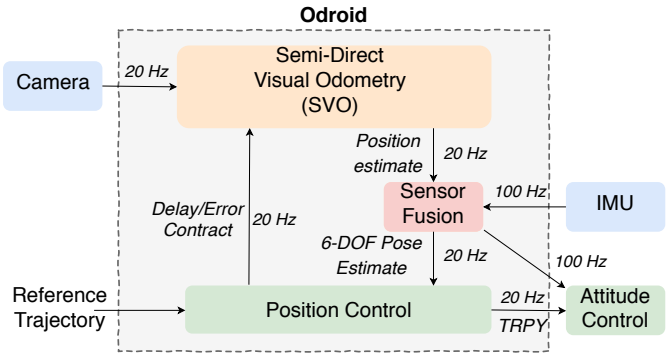

Fig. 10. The control and computational components on-board the hex-rotor.

time increases and the estimation error decreases as expected, but only up to a point. At $\#C = 250$, the estimation error increases. We hypothesize this is due to the decreasing quality of the corners in the environment now being tracked. This is because if the scene is not particularly feature rich, and a sizable fraction of the $\#C$ corners are of poor quality (i.e., unstable or hard to track across frames), and we can expect the localization error to increase as the poor quality of the corners detected adds noise to the visual odometry estimates.

VIII. Case Study: Feedback Control of a Hex-Rotor Robot

To evaluate the performance of our proposed methods, we implemented the contract-driven estimator and control scheme on a KMel robotics hex-rotor robot [31]. The hex-rotor is equipped with a downward facing camera, allowing us to use SVO for localization. The on-board computation platform is an Odroid U-3 [30] computer running Ubuntu as the operating system. The computer also runs Robot Operating System (ROS) [32] which is responsible for executing the estimation and control algorithm at a fixed rate, as well as the communication between them.

A. Experimental Setup

Fig. 10 shows the feedback control loop and flow of information on-board the hex-rotor. Camera images are processed via SVO to generate position estimates, which are used along with IMU information to generate a 6-Degree of Freedom (linear and angular positions and velocities) pose estimate via Unscented Kalman Filtering. The linear components of this state estimate are used by the position control algorithm (RAMPC). The position controller, tasked with tracking a given reference trajectory, generates desired thrust, roll and pitch to be tracked by the low level attitude controller running at a high-rate. The RAMPC also generates the Delay/Error (δ, ϵ) contract for the Contract SVO algorithm to respect at the next discrete time step. More details on the experimental setup are in the appendix.

B. Experiment design

To compare the performance of the RAMPC and SAMPC algorithms developed in this work with that of a MPC that does not leverage co-design, we task the controllers with following two pre-defined reference trajectories, shown in Fig. 11. The reference trajectories are generated using the jerk minimizing trajectory generator of [33].

TABLE I
SVO Modes used in the experiments

Mode	$\#C$	$\delta\,(ms)$	$\epsilon\,(m)$	$\sigma(e_x)$	$\sigma(e_y)$	$\sigma(e_z)$	P (mW)
0	50	24	0.054	0.021	0.033	0.038	778
1	100	30	0.049	0.019	0.027	0.033	862
2	150	34	0.041	0.019	0.024	0.030	870
3	200	38	0.035	0.018	0.022	0.024	951

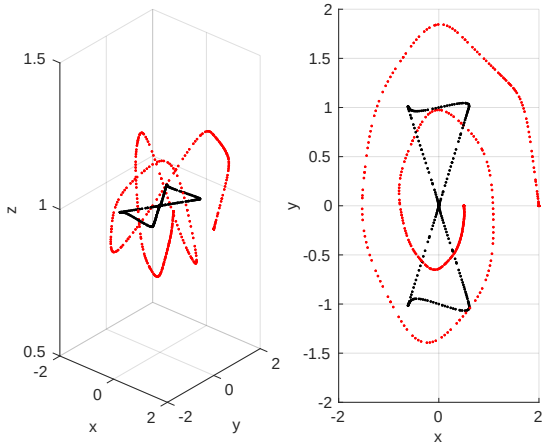

Fig. 11. The two reference trajectories, the spiral is in dashed red and the hourglass is in solid black (color in online version). The figure on the right shows the trajectories projected on the x,y plane. Note, the spiral starts on the outside and ends inwards while the hourglass trajectory starts and ends at $(0, 0, 1)$.

1) **The hourglass trajectory**: This trajectory involves flying straight lines between the desired waypoints, as shown in Fig. 11. In order to get the straight lines, the waypoints are associated with desired velocities of zero (in each axis). The duration of this trajectory is around $14s$. The entire trajectory is flown at a constant height of $1m$. A video of the hex-rotor flying this trajectory can be found at https://youtu.be/-ltJO2gVxWs
2) **Spiral in x, y with sinusoidal variations in z**: This trajectory consists of smooth curves between waypoints, with the waypoints such that in the x,y plane the trajectory looks like a spiral converging towards the origin, while in the z-axis it consists of sinusoidal variations along a reference height of $1m$. The duration of this trajectory is $17s$. A video of the hex-rotor flying this trajectory is at https://youtu.be/hmTRxrq4NJg

These trajectories are flown with: a) the **baseline**, a Robust MPC formulation that does not leverage the co-design of computation and control, with all four chosen modes of SVO used for the state feedback, b) the **RAMPC** algorithm with varying values of α, the weight for the computation power in the optimization, c) the **SAMPC** (with $\zeta = 0.82$) with varying

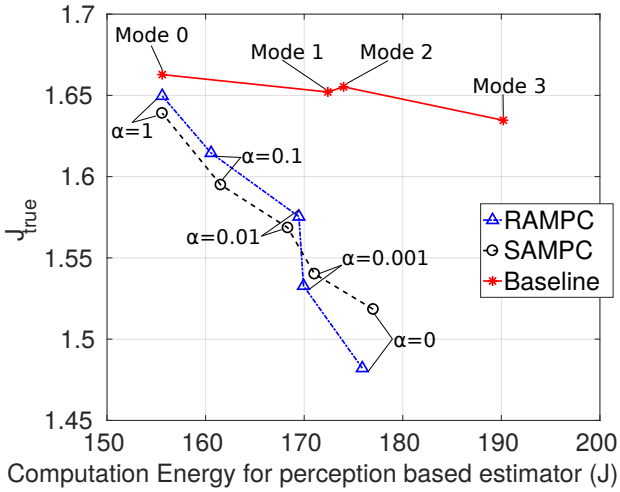

Fig. 12. Performance, hourglass trajectory. The vertical axis has the average control performance (eq. 8) over the flights for the labeled settings,with lower values implying better control performance. The horizontal axis shows the computation power (in Joules) consumed by SVO to perform the state estimation task. The figure shows how our methods (RAMPC/SAMPC) leveraging the co-design have both better control performance while consuming less computation power than the baseline method.

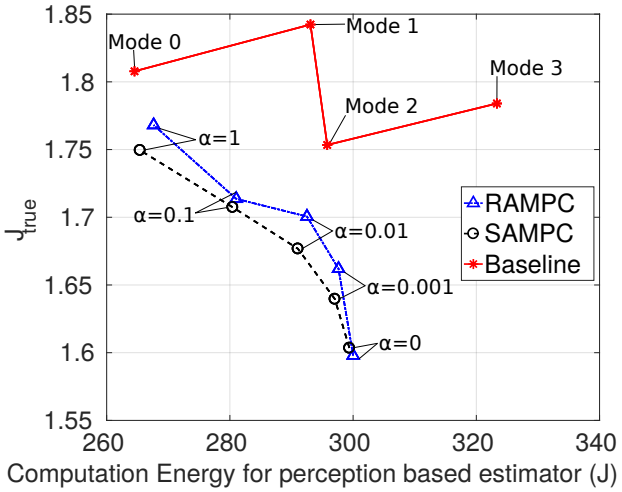

Fig. 13. Performance, spiral trajectory. The vertical axis has the average control performance (eq. (8)) over the flights for the labeled settings,with lower values implying better control performance. The horizontal axis shows the computation power (in Joules) consumed by SVO to perform the state estimation task. Similar to the case for the hourglass trajectory, our methods outperform the baseline.

values of α. Each trajectory is flown twice for each one of these settings to get a comparison of control performance and computation energy consumption. This lead to a total of 56 flights to gather the data presented in this case study.

C. Experimental Results

To measure the performance of the controllers in a standardized manner, we used the following measure of control performance:

$$J_{\text{true}} = \frac{1}{T_{\max}} \sum_{k=0}^{T_{\max}/h} (x_k - x_k^{\text{ref}})^T Q (x_k - x_k^{\text{ref}}) + u_k^T R u_k \qquad (8)$$

Here, x^{ref} is the desired trajectory, and Q and R are the matrices used in the cost of MPC/RAMPC/SAMPC, h is the sampling time ($50ms$) and $T_{\max}$ is the duration of the particular trajectory flown. J_{true} can be accurately evaluated as we have access to the true state, x_k, of the hex-rotor from the Vicon system.

1) Comparison to the baseline: Fig. 12 shows the control performance and the SVO energy consumption for the hourglass trajectory for the baseline RMPC, RAMPC, and SAMPC for different settings. The SAMPC and RAMPC result in lower (average across flights) values of J_{true} than the baseline controller, i.e. better control performance. As the value of α increases, the power consumption decreases and the control performance degrades for the RAMPC and SAMPC. This is expected as α is the weight for the computation power in the overall optimization cost of (3) (and (6)) and increasing it would make computation power more important relative to the control performance. Fig. 13 shows a similar behavior for the spiral trajectory. The notable exception is in the baseline performance, where the most accurate mode (mode 3) of SVO does not result in the best control performance of the fixed mode RMPC controller. This is possibly because the spiral trajectory is more aggressive than the hourglass trajectory, which involves stopping at each corner waypoint of the trajectory, and spending time in mode 3 comes with a computation delay that degrades the control performance despite the increases accuracy of the state estimate. It should be noted that for either trajectory, SAMPC and RAMPC give a better control performance than the baseline for the corresponding computation energy consumption. For both cases, the control performance of SAMPC and RAMPC are close to each other, with the SAMPC slightly outperforming the RAMPC for the spiral trajectory.

Summary: Across both the trajectories, the best case control performance of our methods results in about a 10% improvement compared to that of the baseline. To achieve this performance, our methods result in SVO using about $5-6\%$ and less computation energy compared to the baseline (at the setting resulting in best control performance). This clearly demonstrates the benefit of the co-design between the perception-based estimation and the control algorithms.

2) Impact of the weight for computation power (α): As α takes on a high value, the control performance of RAMPC and SAMPC for the hourglass trajectory approaches that of the baseline RAMPC with SVO mode fixed to 0. This is backed up the observation of tables II, III which show that for $\alpha = 1$, the RAMPC and SAMPC select mode 0, the low-power but high estimation error mode, of SVO all the time. The tables II, III show the fraction of time spent in each mode of SVO as α changes. Note that as α, the weight for the computation power, increases the time spent in the low power mode 0 also increases while the time spent in the more accurate but higher power modes accordingly decreases. Similar behavior is noted

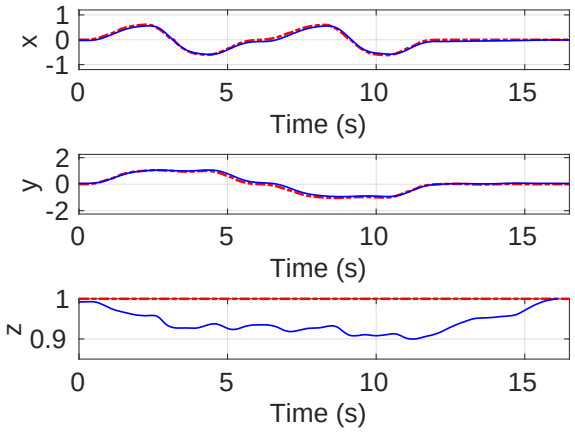

Fig. 14. (Color online) Reference positions (dashed red) and actual positions (blue) of the hex-rotor flying the hourglass trajectory while being controlled by the SAMPC ($\alpha = 0$).

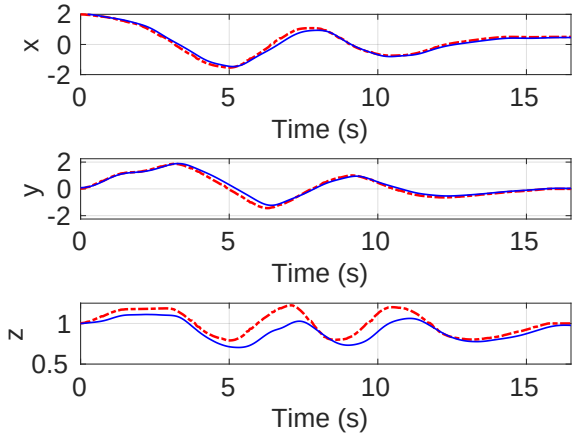

Fig. 15. (Color online) Reference positions (dashed red) and actual positions (blue) of the hex-rotor flying the spiral trajectory while being controlled by the RAMPC ($\alpha = 0.1$).

TABLE II
FRACTION OF TIME SPENT IN MODES: HOURGLASS TRAJECTORY, RAMPC

	Mode 0	Mode 1	Mode 2	Mode 3
$\alpha = 0$	0.398	0.008	0.024	0.570
$\alpha = 0.001$	0.523	0.004	0.024	0.440
$\alpha = 0.01$	0.557	0.000	0.067	0.374
$\alpha = 0.1$	0.820	0.000	0.055	0.123
$\alpha = 1$	1.000	0.000	0.000	0.000

for the spiral trajectory, and tables IV and V show the fraction of time spent in the different SVO modes as α changes for RAMPC and SAMPC flying the spiral trajectory respectively.

3) Snapshots of the control performance of RAMPC and SAMPC: Fig. 14 shows the reference and actual positions of the hex-rotor (in x,y and z co-ordiantes) as function of time for the hourglass trajectory controlled by the SAMPC ($\alpha = 0$). Note the near perfect tracking in x and y. The small dip in the height (z co-ordinate) is due to combination of model error (due to inaccuracy of the mass) as well the effect of linearization around hover. Fig. 15 shows the reference and actual positions versus time for the RAMPC ($\alpha = 0.1$) flying the spiral trajectory, showing similarly good tracking performance as in the hourglass trajectory.

Finally, Fig. 16 shows the selected mode of the SVO (with SAMPC, for the spiral trajectory flown by the SAMPC ($\alpha = 0.001$) changing over the discrete time steps, as well as the evolution of the tracking cost at each time step.

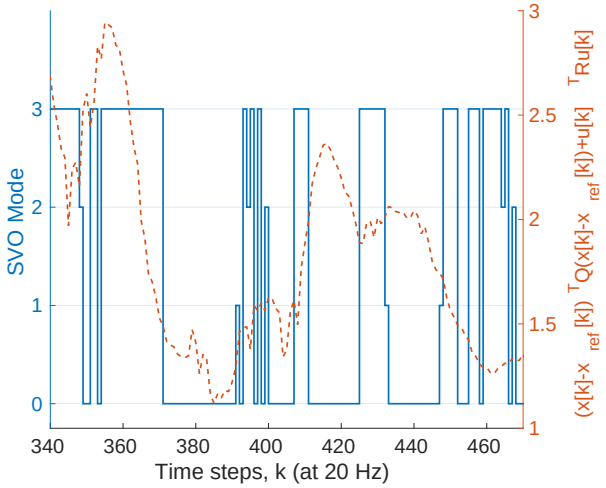

Fig. 16. SVO Mode and control cost over time for the spiral trajectory flown with SAMPC at $\alpha = 0.001$.

IX. CONCLUSION

In this paper we presented a contract-driven methodology for co-design of estimation and control for autonomous systems. The basic idea is that the control algorithm requests a delay and estimation error (δ, ϵ) contract that the perception-and-estimation algorithm realizes. The control algorithm we designed aims to set time-varying contracts to maximise a per-

TABLE III
FRACTION OF TIME SPENT IN MODES: HOURGLASS TRAJECTORY, SAMPC

	Mode 0	Mode 1	Mode 2	Mode 3
$\alpha = 0$	0.374	0.000	0.004	0.621
$\alpha = 0.001$	0.514	0.016	0.051	0.418
$\alpha = 0.01$	0.617	0.000	0.032	0.351
$\alpha = 0.1$	0.793	0.000	0.076	0.131
$\alpha = 1$	1.000	0.000	0.000	0.000

TABLE IV
FRACTION OF TIME SPENT IN MODES: SPIRAL TRAJECTORY, RAMPC

	Mode 0	Mode 1	Mode 2	Mode 3
$\alpha = 0$	0.381	0.015	0.015	0.589
$\alpha = 0.001$	0.422	0.012	0.018	0.548
$\alpha = 0.01$	0.504	0.000	0.041	0.455
$\alpha = 0.1$	0.680	0.000	0.082	0.238
$\alpha = 1$	0.995	0.000	0.015	0.000

TABLE V
FRACTION OF TIME SPENT IN MODES: SPIRAL TRAJECTORY, SAMPC

	Mode 0	Mode 1	Mode 2	Mode 3
$\alpha = 0$	0.396	0.003	0.018	0.584
$\alpha = 0.001$	0.434	0.009	0.018	0.540
$\alpha = 0.01$	0.531	0.000	0.038	0.431
$\alpha = 0.1$	0.695	0.000	0.073	0.232
$\alpha = 1$	0.971	0.000	0.029	0.000

formance function while respecting feasibility constraints and stability under the time varying execution delay and estimation error from the estimator. We also illustrate how the contract-driven perception-and-estimation algorithm is designed offline and used at run-time to best meet the (δ, ϵ) contracts set for it. Through a case study on a flying hexrotor, we showed the applicability of our scheme to real-time closed loop system. The experimental results show the good performance of our scheme and how it outperforms regular Model Predictive Control which does not leverage co-design. A key result showed how our closed loop solution is more energy efficient than MPC while achieving better tracking performance. A focus of ongoing research is to overcome the necessity of the contracts always being met by the estimator. Another focus is on an automated tool chain to profile perception algorithms commonly used in autonomous systems.

ACKNOWLEDGEMENTS

We would like to thank Kuk Jang for his help in creating several of the diagrams in this paper.

REFERENCES

[1] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in *Robotics and Automation (ICRA), 2014 IEEE Intl. Conf. on*. IEEE, 2014.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct 2015.

[3] M. Boddy and T. Dean, "Solving Time-dependent Planning Problems," *Joint Conf. on AI*, pp. 979–984, 1989.

[4] Y. V. Pant, H. Abbas, K. Mohta, T. X. Nghiem, J. Devietti, and R. Mangharam, "Co-design of anytime computation and robust control," in *2015 IEEE Real-Time Systems Symposium*, Dec 2015, pp. 43–52.

[5] S. Zilberstein, "Using anytime algorithms in intelligent systems," *AI Magazine*, vol. 17, no. 3, 1996.

[6] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime Search in Dynamic Graphs," *Artif. Intell.*, vol. 172, no. 14, pp. 1613–1643, 2008.

[7] M. Wellman and C. L. Liu, "State-Space Abstraction for Anytime Evaluation of Probabilistic Networks," *Conf. on Uncertainty in AI*, 1994.

[8] R. Mangharam and A. Saba, "Anytime Algorithms for GPU Architectures," in *Proc. of the IEEE Real-Time Systems Symposium*, 2011.

[9] Y. V. Pant, H. Abbas, K. N. Nischal, P. Kelkar, D. Kumar, J. Devietti, and R. Mangharam, "Power-efficient algorithms for autonomous navigation," in *International Conference on Complex Systems Engineering*, 2015.

[10] D. Quevedo and V. Gupta, "Sequence-based anytime control," *IEEE Trans. Autom. Control*, vol. 58, no. 2, pp. 377–390, Feb 2013.

[11] R. Bhattacharya and G. J. Balas, "Anytime control algorithm: Model reduction approach," *Journal of Guidance and Control*, vol. 27, no. 5, pp. 767–776, 2004.

[12] D. Fontanelli, L. Greco, and A. Bicchi, "Anytime control algorithms for embedded real-time systems," in *Hybrid Systems: Computation and Control*. Springer, 2008, pp. 158–171.

[13] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[14] D. K. Jha, M. Zhu, Y. Wang, and A. Ray, "Data-driven anytime algorithms for motion planning with safety guarantees," in *American Control Conference*, 2016.

[15] S. Choudhury, "Anytime geometric motion planning on large dense roadmaps," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, July 2017.

[16] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. J. Teller, "Anytime motion planning using the rrt*," 2011.

[17] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[18] G. Frehse, A. Hamann, S. Quinton, and M. Woehrle, "Formal analysis of timing effects on closed-loop properties of control software," in *Real-Time Systems Symposium (RTSS), 2014 IEEE*, Dec 2014, pp. 53–62.

[19] D. de Niz, L. Wrage, N. Storer, A. Rowe, and R. Rajukar, "On Resource Overbooking in an Unmanned Aerial Vehicle," *IEEE/ACM Third International Conference on Cyber-Physical Systems*, 2012.

[20] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek, "A Resource Allocation Model for QoS Mgmt." *IEEE RTSS*, 1997.

[21] S. Sidiroglou-Douskos, S. Misailovic, H. Hoffmann, and M. Rinard, "Managing performance vs. accuracy trade-offs with loop perforation," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE '11, 2011.

[22] M. Carbin, S. Misailovic, and M. C. Rinard, "Verifying quantitative reliability for programs that execute on unreliable hardware," in *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages and Applications*, 2013.

[23] R. St. Amant, A. Yazdanbakhsh, J. Park, B. Thwaites, H. Esmaeilzadeh, A. Hassibi, L. Ceze, and D. Burger, "General-purpose code acceleration with limited-precision analog computation," in *Proc. of the 41st Annual International Symposium on Computer Architecuture*, ser. ISCA '14, 2014.

[24] E. Camacho and C. Bordons, *Model predictive control*. Springer Verlag, 2004.

[25] A. Richards and J. How, "Robust model predictive control with imperfect information," in *American Control Conference*, 2005, pp. 268–273.

[26] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[27] B. Kouvaritakis, M. Cannon, and S. V. Rakovic, "Explicit use of probabilistic distributions in linear predictive control," in *UKACC International Conference on Control*, 2010.

[28] "Motion capture systems — vicon," https://www.vicon.com, accessed: 2018-10-30.

[29] "ODROID Smart Power," http://odroid.com/, accessed: 2015-05-13.

[30] "ODROID-U3," http://odroid.com/, accessed: 2015-05-13.

[31] "Kmel (qualcomm)," https://www.grasp.upenn.edu/startups/kmel-qualcomm, accessed: 2018-08-13.

[32] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[33] M. W. Mueller, M. Hehn, and R. DÁndrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," in *IEEE Transactions on Robotics*, 2015.

[34] E. C. Kerrigan, "Robust constraint satisfaction: Invariant sets and predictive control," Ph.D. dissertation, University of Cambridge, 2000.

[35] S. Boucheron, G. Lugosi, and P. Massart, *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[36] J. Mattingley and S. Boyd, "Cvxgen: a code generator for embedded convex optimization," *Optimization and Engineering*, 2012.

APPENDIX

X. THE ROBUST CASE

In this appendix we give the detailed mathematical derivation of the results of Section III. The controller is designed using a Robust Model Predictive Control (RMPC) approach via constraint restriction [25], [26], and augments it by an adaptation to the error-delay curve of the estimator. In order to ensure robust safety and feasibility, the key idea of the RMPC approach is to tighten the constraint sets iteratively to account

for possible effect of the disturbances. As time progresses, this "robustness margin" is used in the MPC optimization with the nominal dynamics, i.e., the original dynamics where the disturbances are either removed or replaced by nominal disturbances. Because only the nominal dynamics are used, the complexity of the optimization is the same as for the nominal problem.

Since the controller only has access to the estimated state $\hat{x}$, we need to rewrite the plant's dynamics with respect to $\hat{x}$. The error between x_k and $\hat{x}_k$ is $e_k = x_k - \hat{x}_k$. At time step $k+1$ we have

$$\begin{aligned}\hat{x}_{k+1} &= x_{k+1} - e_{k+1} \\ &= Ax_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + w_k - e_{k+1},\end{aligned}$$

then, by writing $x_k = \hat{x}_k + e_k$, we obtain the dynamics

$$\hat{x}_{k+1} = A\hat{x}_k + B_1(\delta_k)u_{k-1} + B_2(\delta_k)u_k + \hat{w}_k \tag{9}$$

where $\hat{w}_k = w_k + Ae_k - e_{k+1}$. The set of possible values of $\hat{w}_k$ depends on the estimation accuracy at steps k and $k+1$ and is denoted by $\widehat{\mathcal{W}}(\epsilon_k, \epsilon_{k+1})$, i.e., $\widehat{\mathcal{W}}(\epsilon, \epsilon') := \{w + Ae - e' \mid w \in \mathcal{W}, e \in \mathcal{E}(\epsilon), e' \in \mathcal{E}(\epsilon')\}$. Note that $\widehat{\mathcal{W}}(\epsilon_k, \epsilon_{k+1})$ is independent of the time step k. It can be computed as $\widehat{\mathcal{W}}(\epsilon, \epsilon') = \mathcal{W} \oplus A\mathcal{E}(\epsilon) \oplus (-\mathcal{E}(\epsilon'))$ where the symbol $\oplus$ denotes the Minkowski sum of two sets.

The dynamics in (9) has a non-standard form where it depends on both the current and the previous control inputs. However we can expand the state variable to store the previous control input as

$$\hat{z}_k = \begin{bmatrix} \hat{x}_k \\ u_{k-1} \end{bmatrix} \in \mathbb{R}^{n+m}$$

and rewrite the dynamics as, for all $k \geq 0$,

$$\hat{z}_{k+1} = \hat{A}(\delta_k)\hat{z}_k + \hat{B}(\delta_k)u_k + \hat{F}\hat{w}_k. \tag{10}$$

Here, the system matrices are

$$\begin{aligned}\hat{A}(\delta_k) &= \begin{bmatrix} A & B_1(\delta_k) \\ \mathbf{0}_{m\times n} & \mathbf{0}_{m\times m} \end{bmatrix}, \\ \hat{B}(\delta_k) &= \begin{bmatrix} B_2(\delta_k) \\ \mathbb{I}_m \end{bmatrix}, \quad \hat{F} = \begin{bmatrix} \mathbb{I}_n \\ \mathbf{0}_{m\times n} \end{bmatrix}.\end{aligned} \tag{11}$$

Let the actual expanded state be $z_k = \left[x_k^T, u_{k-1}^T\right]^T$. Because the expanded state consists of both the plant's state and the previous control input, the state constraint $x_k \in X$ and the control constraint $u_k \in U$ are equivalent to the joint constraint $z_k \in X \times U$. We can now describe the RAMPC algorithm for the dynamics in (10).

A. Tractable RAMPC Algorithm

Let $N \geq 1$ be the horizon length of the RMPC optimization. Because the system matrices in the state equation (10) depend nonlinearly on the variables δ_k, the RMPC optimization is generally a mixed-integer nonlinear program, which is very hard to solve. To simplify the RMPC optimization to make it tractable, we fix the estimation mode for the entire RMPC horizon.

Let $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ denote the RMPC optimization problem at step $k \geq 0$ where the current state estimate is $\hat{x}_k$, the current estimation mode is $(\delta_k, \epsilon_k) \in \Delta$, the previous control input is u_{k-1}, and the estimation mode for the entire horizon (after step k) is fixed at $(\delta, \epsilon) \in \Delta$. Since the system matrices become constant now, if the stage cost $\ell(\cdot)$ is linear or positive semidefinite quadratic, each optimization problem $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_\cdot, \delta_\cdot, \epsilon_\cdot, u_{\cdot-1})$ is tractable and can be solved efficiently as we will show later. The RAMPC algorithm with Anytime Estimation is stated in Alg. 1 .

B. RMPC Formulation

We formulate the RMPC optimization $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ with respect to the nominal dynamics, which is the original dynamics in Eq. (10) but the disturbances are either removed or replaced by nominal disturbances. To ensure robust feasibility and safety, the state constraint set is tightened after each step using a candidate stabilizing state feedback control, and a terminal constraint is derived. In this RMPC formulation, we extend the approach in [25], [26]. At time step k, given $(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ and for a fixed (δ, ϵ), we solve the following optimization

$$J^*_{\delta,\epsilon}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1}) = \min_{\boldsymbol{u},\boldsymbol{x}} \sum_{j=0}^{N} \ell(\overline{x}_{k+j|k}, u_{k+j|k}) \tag{12a}$$

$$\text{subject to, } \forall j \in \{0, \ldots, N\}$$

$$\overline{z}_{k+j+1|k} = \hat{A}(\delta_{k+j|k})\overline{z}_{k+j|k} + \hat{B}(\delta_{k+j|k})u_{k+j|k} \tag{12b}$$

$$(\delta_{k+j+1|k}, \epsilon_{k+j+1|k}) = (\delta, \epsilon)$$

$$(\delta_{k|k}, \epsilon_{k|k}) = (\delta_k, \epsilon_k) \tag{12c}$$

$$\overline{x}_{k+j|k} = \begin{bmatrix} \mathbb{I}_n & \mathbf{0}_{n\times m} \end{bmatrix} \overline{z}_{k+j|k} \tag{12d}$$

$$\overline{z}_{k|k} = \left[\hat{x}_k^T, u_{k-1}^T\right]^T \tag{12e}$$

$$\overline{z}_{k+j|k} \in \mathcal{Z}_j(\epsilon_k, \epsilon) \tag{12f}$$

$$\overline{z}_{k+N+1|k} \in \mathcal{Z}_f(\epsilon_k, \epsilon) \tag{12g}$$

in which $\overline{z}$ and $\overline{x}$ are the variables of the nominal dynamics. The constraints of the optimization are explained below.

- (12b) is the nominal dynamics.
- (12c) states that the estimation mode is fixed at (δ, ϵ) except for the first time step when it is (δ_k, ϵ_k).
- (12d) extracts the nominal state $\overline{x}$ of the plant from the nominal expanded state $\overline{z}$.
- (12e) initializes the nominal expanded state at time step k by stacking the current state estimate and the previous control input.
- (12f) tightens the admissible set of the nominal expanded states by a sequence of shrinking sets.
- (12g) constrains the terminal expanded state to the terminal constraint set $\mathcal{Z}_f$.

The state constraint $\mathcal{Z}_j$: The tightened state constraint sets $\mathcal{Z}_j(\epsilon_k, \epsilon)$ are parameterized with two parameters ϵ_k and ϵ. They are defined as follows, for all $j \in \{0, \dots, N\}$

$$\mathcal{Z}_0(\epsilon_k, \epsilon) = \mathcal{Z} \ominus \hat{F}\mathcal{E}(\epsilon_k) \tag{13a}$$

$$\mathcal{Z}_{j+1}(\epsilon_k, \epsilon) = \mathcal{Z}_j(\epsilon, \epsilon) \ominus L_j \hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon) \tag{13b}$$

in which the symbol $\ominus$ denotes the Pontryagin difference between two sets. The set $\mathcal{Z}$ combines the constraints for both the plant's state and the control input: $\mathcal{Z} = X \times U$. The matrix L_j is the state transition matrix for the nominal dynamics in (12b) under a candidate state feedback gain $K_j(\delta)$, for $j \in \{0, \dots, N\}$

$$L_0 = \mathbb{I} \tag{14}$$

$$L_{j+1} = (\hat{A}(\delta) + \hat{B}(\delta)K_j(\delta))L_j \tag{15}$$

Note that the possibly time-varying sequence $K_j(\delta)$ is designed for each choice of δ (i.e., the system matrices $\hat{A}(\delta)$ and $\hat{B}(\delta)$), hence L_j depends on δ; however we write L_j for brevity. The candidate control $K_j(\delta)$ is designed to stabilize the nominal system (12b), desirably as fast as possible so that the sets $\mathcal{Z}_j$ are shrunk as little as possible. In particular, if $K_j(\delta)$ renders the nominal system nilpotent after $M < N$ steps then $L_j = \mathbf{0}$ for all $j \geq M$, therefore $\mathcal{Z}_j(\epsilon_k, \epsilon) = \mathcal{Z}_M(\epsilon_k, \epsilon)$ for all $j > M$.

The terminal constraint $\mathcal{Z}_f$: $\mathcal{Z}_f$ is given by

$$\mathcal{Z}_f(\epsilon_k, \epsilon) = \mathcal{C}(\delta, \epsilon) \ominus L_N \hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon) \tag{16}$$

where $\mathcal{C}(\delta, \epsilon)$ is a robust control invariant admissible set for δ [34], i.e., there exists a feedback control law $u = \kappa(z)$ such that $\forall z \in \mathcal{C}(\delta, \epsilon)$ and $\forall w \in \widehat{\mathcal{W}}(\epsilon, \epsilon)$

$$\hat{A}(\delta)z + \hat{B}(\delta)\kappa(z) + L_N \hat{F}w \in \mathcal{C}(\delta, \epsilon) \tag{17}$$

$$z \in \mathcal{Z}_N(\epsilon, \epsilon) \tag{18}$$

We remark that $\mathcal{C}(\delta, \epsilon)$ does not depend on (δ_k, ϵ_k), therefore it can be computed offline for each mode (δ, ϵ).

C. Proofs of Feasibility

The RMPC formulation of the previous section, with a fixed estimation mode $(\delta, \epsilon) \in \Delta$, is designed to ensure that the control problem is robustly feasible, as stated in the following theorem.

Theorem X.1 (Robust Feasibility of RAMPC)**.** *For any estimation mode (δ, ϵ), if $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible then the system (2) controlled by the RAMPC and subjected to disturbances constrained by $w_k \in \mathcal{W}$ robustly satisfies the state constraint $x_k \in X$ and the control input constraint $u_k \in U$, and all subsequent optimizations $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ $\forall k > k_0$, are feasible.*

Proof. We will prove the theorem by recursion. We will show that if at any time step k the RAMPC problem $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible and feasible control input $u_k = u^\star_{k|k}$ is applied with estimation mode $(\delta_{k+1}, \epsilon_{k+1}) = (\delta, \epsilon)$ then u_k is admissible and at the next time step $k+1$, the actual plant's state x_{k+1} is inside X and the optimization $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$ is feasible for all disturbances. Then we can conclude the theorem because, by recursion, feasibility at time step k_0 implies robust constraint satisfaction and feasibility at time step $k_0 + 1$, and so on at all subsequent time steps.

Suppose $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k, \delta_k, \epsilon_k, u_{k-1})$ is feasible. Then it has a feasible solution $\left(\{\overline{z}^\star_{k+j|k}\}_{j=0}^{N+1}, \{u^\star_{k+j|k}\}_{j=0}^{N}\right)$ that satisfies all the constraints in (12). Now we will construct a feasible candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$ at the next time step by shifting the above solution by one step. Consider the following candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$:

$$\overline{z}_{k+j+1|k+1} = \overline{z}^\star_{k+j+1|k} + L_j \hat{F}\hat{w}_k \tag{19a}$$

$$\overline{z}_{k+N+2|k+1} = \hat{A}(\delta)\,\overline{z}_{k+N+1|k+1} + \hat{B}(\delta)\,u_{k+N+1|k+1} \tag{19b}$$

$$u_{k+i+1|k+1} = u^\star_{k+i+1|k} + K_i(\delta)\,L_i \hat{F}\hat{w}_k \tag{19c}$$

$$u_{k+N+1|k+1} = \kappa\left(\overline{z}_{k+N+1|k+1}\right) \tag{19d}$$

in which $j \in \{0, \dots, N\}$, $i \in \{0, \dots, N-1\}$, and $\kappa(\cdot)$ is the feedback control law for the invariant set $\mathcal{C}(\delta, \epsilon)$ that is used in the terminal set. We first show that the input and state constraints are satisfied for u_k and x_{k+1}, then we will prove the feasibility of the above candidate solution for $\mathbb{P}_{\delta,\epsilon}(\hat{x}_{k+1}, \delta_{k+1}, \epsilon_{k+1}, u_k)$.

Validity of the applied input and the next state: The next plant's state is

$$\begin{aligned} x_{k+1} &= Ax_k + B_1(\delta_k)\,u_{k-1} + B_2(\delta_k)\,u_k + w_k \\ &= A(\hat{x}_k + e_k) + B_1(\delta_k)\,u_{k-1} + B_2(\delta_k)\,u^\star_{k|k} + w_k \\ &= \begin{bmatrix} A & B_1(\delta_k) \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ u_{k-1} \end{bmatrix} + B_2(\delta_k)\,u^\star_{k|k} \\ &\qquad + e_{k+1} + (w_k + Ae_k - e_{k+1}) \end{aligned}$$

in which $e_{k+1} \in \mathcal{E}(\epsilon)$ and $(w_k + Ae_k - e_{k+1}) \in \widehat{\mathcal{W}}(\epsilon_k, \epsilon)$. Note that $\overline{z}^\star_{k|k} = \left[\hat{x}_k^T, u_{k-1}^T\right]^T$. Hence we have

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} &= \hat{A}(\delta_k)\overline{z}^\star_{k|k} + \hat{B}(\delta_k)u^\star_{k|k} \\ &\qquad + \hat{F}e_{k+1} + \hat{F}(w_k + Ae_k - e_{k+1}) \\ &= \overline{z}^\star_{k+1|k} + \hat{F}e_{k+1} + \hat{F}(w_k + Ae_k - e_{k+1}) \end{aligned}$$

where we use the dynamics in (12b). From (12f) and (13), $\overline{z}^\star_{k+1|k}$ satisfies $\overline{z}^\star_{k+1|k} \in \mathcal{Z}_1(\epsilon_k, \epsilon) = \mathcal{Z} \ominus \hat{F}\mathcal{E}(\epsilon) \ominus \hat{F}\widehat{\mathcal{W}}(\epsilon_k, \epsilon)$. It follows that $\left[x_{k+1}^T, u_k^T\right]^T \in \mathcal{Z} = X \times U$, therefore $x_{k+1} \in X$ and $u_k \in U$.

Initial condition: We have from (10) that $\hat{z}_{k+1} = \hat{A}(\delta_k)\hat{z}_k + \hat{B}(\delta_k)u_k + \hat{F}\hat{w}_k$. On the other hand, by (19a),

$$\begin{aligned} \overline{z}_{k+1|k+1} &= \overline{z}^\star_{k+1|k} + L_0 \hat{F}\hat{w}_k \\ &= \hat{A}(\delta_k)\overline{z}^\star_{k|k} + \hat{B}(\delta_k)u^\star_{k|k} + L_0 \hat{F}\hat{w}_k. \end{aligned}$$

Note that $\overline{z}^\star_{k|k} = \hat{z}_k$, $u_k = u^\star_{k|k}$, and $L_0 = \mathbb{I}$. Therefore $\overline{z}_{k+1|k+1} = \hat{z}_{k+1}$, hence the initial condition is satisfied.

Dynamics: We show that the candidate solution satisfies the dynamics constraint in Eq. (12b). For $0 \leq j < N$ we have

$$\begin{aligned}
&\overline{z}_{k+j+2|k+1} \\
&= \overline{z}^{\star}_{k+j+2|k} + L_{j+1}\hat{F}\hat{w}_k \\
&= \hat{A}(\delta)\,\overline{z}^{\star}_{k+j+1|k} + \hat{B}(\delta)u^{\star}_{k+j+1|k} + L_{j+1}\hat{F}\hat{w}_k \\
&= \hat{A}(\delta)\left(\overline{z}_{k+j+1|k+1} - L_j\hat{F}\hat{w}_k\right) \\
&\quad + \hat{B}(\delta)\left(u_{k+j+1|k+1} - K_j(\delta)\,L_j\hat{F}\hat{w}_k\right) + L_{j+1}\hat{F}\hat{w}_k \\
&= \hat{A}(\delta)\,\overline{z}_{k+j+1|k+1} + \hat{B}(\delta)u_{k+j+1|k+1} \\
&\quad - \left(\hat{A}(\delta) + \hat{B}(\delta)K_j(\delta)\right)L_j\hat{F}\hat{w}_k + L_{j+1}\hat{F}\hat{w}_k \\
&= \hat{A}(\delta)\,\overline{z}_{k+j+1|k+1} + \hat{B}(\delta)u_{k+j+1|k+1}
\end{aligned}$$

where the equality in (15) is used to derive the last equality. Therefore the dynamics constraint is satisfied for all $0 \leq j < N$. For $j = N$, the constraint is satisfied by construction (19b).
State constraints: We need to show that $\overline{z}_{(k+1)+j|k+1} \in \mathcal{Z}_j(\epsilon,\epsilon)$ for all $j \in \{0,\ldots,N\}$. Consider any $0 \leq j < N$. (13b) states that $\mathcal{Z}_{j+1}(\epsilon_k,\epsilon) = \mathcal{Z}_j(\epsilon,\epsilon) \ominus L_j\hat{F}\widehat{\mathcal{W}}(\epsilon_k,\epsilon)$. From the construction of the candidate solution we have $\overline{z}_{k+j+1|k+1} = \overline{z}^{\star}_{k+j+1|k} + L_j\hat{F}\hat{w}_k$, where $\hat{w}_k \in \widehat{\mathcal{W}}(\epsilon_k,\epsilon)$ and $\overline{z}^{\star}_{k+j+1|k} \in \mathcal{Z}_{j+1}(\epsilon_k,\epsilon)$. By definition of the Pontryagin difference, we conclude that $\overline{z}_{k+j+1|k+1} \in \mathcal{Z}_j(\epsilon,\epsilon)$ for all $j \in \{0,\ldots,N-1\}$.

At $j = N$ the candidate solution in (19a) gives us $\overline{z}_{(k+1)+N|k+1} = \overline{z}^{\star}_{k+N+1|k} + L_N\hat{F}\hat{w}_k$. Because $\overline{z}^{\star}_{k+N+1|k} \in \mathcal{Z}_f(\epsilon_k,\epsilon) = \mathcal{C}(\delta,\epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon_k,\epsilon)$ and $\hat{w}_k \in \widehat{\mathcal{W}}(\epsilon_k,\epsilon)$, we have $\overline{z}_{(k+1)+N|k+1} \in \mathcal{C}(\delta,\epsilon)$. The definition of $\mathcal{C}(\delta,\epsilon)$ in (17) implies $\mathcal{C}(\delta,\epsilon) \subseteq \mathcal{Z}_N(\epsilon,\epsilon)$. Therefore $\overline{z}_{(k+1)+N|k+1} \in \mathcal{Z}_N(\epsilon,\epsilon)$.
Terminal constraint: We need to show that $\overline{z}_{k+N+2|k+1} \in \mathcal{Z}_f(\epsilon,\epsilon) = \mathcal{C}(\delta,\epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon,\epsilon)$. Add $L_N\hat{F}\hat{w}$, for any $\hat{w} \in \widehat{\mathcal{W}}(\epsilon,\epsilon)$, to both sides of (19b) and note that $u_{k+N+1|k+1} = \kappa\left(\overline{z}_{k+N+1|k+1}\right)$, we have

$$\begin{aligned}
\overline{z}_{k+N+2|k+1} + L_N\hat{F}\hat{w} = \hat{A}(\delta)\,\overline{z}_{k+N+1|k+1} \\
+ \hat{B}(\delta)\,\kappa\left(\overline{z}_{k+N+1|k+1}\right) + L_N\hat{F}\hat{w}.
\end{aligned}$$

It follows from $\overline{z}_{k+N+1|k+1} \in \mathcal{C}(\delta,\epsilon)$ and from the definition of the invariant control invariant admissible set $\mathcal{C}(\delta,\epsilon)$ (Eq.(17)) that $\overline{z}_{k+N+2|k+1} + L_N\hat{F}\hat{w} \in \mathcal{C}(\delta,\epsilon)$ for all $w \in \widehat{\mathcal{W}}(\epsilon,\epsilon)$. Then by definition of the Pontryagin difference, we conclude that $\overline{z}_{k+N+2|k+1} \in \mathcal{C}(\delta,\epsilon) \ominus L_N\hat{F}\widehat{\mathcal{W}}(\epsilon,\epsilon) = \mathcal{Z}_f(\epsilon,\epsilon)$.

□

The control algorithm in Alg. 1 , in each time step k, solves $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_k,\delta_k,\epsilon_k,u_{k-1})$ for each estimation mode $(\delta,\epsilon) \in \Delta$ and selects the control input u_k and the next estimation mode $(\delta_{k+1},\epsilon_{k+1})$ corresponding to the best total cost $J_{(\delta,\epsilon)}$. Therefore, during the course of control, the algorithm may switch between the estimation modes in Δ depending on the system's state. Thm. X.2 states that if the control algorithm Alg. 1 is feasible in its first time step then it will be robustly feasible and the state and control input constraints are also robustly satisfied.

Theorem X.2. *If at the initial time step there exists $(\delta,\epsilon) \in \Delta$ such that $\mathbb{P}_{(\delta,\epsilon)}(\hat{x}_0,\delta_0,\epsilon_0,u_{0-1})$ is feasible then the system Eq. 9 controlled by Alg. 1 and subjected to disturbances constrained s.t. $w_k \in \mathcal{W},\forall k \geq 0$ robustly satisfies the state constraint $x_k \in X,\forall k \geq 0$ and the control input constraint $u_k \in U,\forall k \geq 0$, and all subsequent iterations of the algorithm are feasible.*

Proof. The Theorem can be proved by recursively applying Thm. X.1. Indeed, suppose at time step k the algorithm is feasible and results in control input u_k and next estimation mode $(\delta_{k+1},\epsilon_{k+1})$, then $\mathbb{P}_{(\delta_{k+1},\epsilon_{k+1})}(\hat{x}_k,\delta_k,\epsilon_k,u_{k-1})$ is feasible. By Thm. X.1, $u_k \in U$ and at the next time step $k+1$, $x_{k+1} \in X$ and $\mathbb{P}_{(\delta_{k+1},\epsilon_{k+1})}(\hat{x}_{k+1},\delta_{k+1},\epsilon_{k+1},u_{k+1-1})$ is also feasible, hence the algorithm is feasible. Therefore, the Theorem holds by induction. □

XI. The Stochastic case

When the estimation errors are drawn from a distribution, the contracts for the perception algorithm are of the form (δ,Σ) (computation time and estimation error covariance respectively, assume 0 mean distributions w.l.o.g). In the following section we consider the case where the estimation errors come from a general distribution (with bounded second moment) and have bounded support.

The main results are summarized in the following theorem (restated here):

Theorem XI.1 (Robust Feasibility of SAMPC)**.** *For any estimation mode (δ,Σ), if $\widetilde{\mathbb{P}}_{\delta,\Sigma}(\hat{x}_k,\delta_k,\Sigma_k,u_{k-1})$ is feasible then the system (2) controlled by the RAMPC and subjected to disturbances constrained by $w_k \in \mathcal{W}$ satisfies, with probability at least $1-\zeta$, the state constraint $x_k \in X$ and the control input constraint $u_k \in U$, and the subsequent optimization $\widetilde{\mathbb{P}}_{\delta,\Sigma}(\hat{x}_{k+1},\delta_{k+1},\Sigma_{k+1},u_{k+1-1})$, are feasible with Probability* 1.

We begin with a candidate solution similar to the one from the robust (worst case) Anytime MPC case, i.e. (19). Since the proofs are very similar in nature to those in the robust case, we will build on top of those existing proofs, dropping subscripts for mode, time step, and constraint number where necessary for ease of notation.

A. Constraint tightening

Here, we assume that the estimation error e comes from a distribution with a known bounded variance and a known mean (set to 0 w.l.o.g) for each mode of the perception-based estimator (δ,Σ).For the sake of simplicity, we assume that the process noise w is also such a distribution and has a bounded support.

Starting from a chance constraint of the form $P(Hz_{k+j|k} \leq g) \geq 1-\zeta$ with $g \in \mathbb{R}^p$, constraint separation tells us that this constraint is satisfied when $\forall i = 1,\ldots,p$:

$$P(H_i^T z_{k+j|k} \leq g_i) \geq 1-\zeta_i \tag{20}$$

where $\zeta_i|\zeta_i \geq 0 \forall i$, $\sum_{i=1}^{p} \zeta_i = \zeta$. This is satisfied by the candidate solution when:

$$P(H_i^T(\bar{z}_{k+j|k} + \sum_{l=0}^{j-1} L_l \hat{F} \hat{w}_{k+(j-l)} - \hat{F} e_{k+j}) \leq g_i) \geq 1 - \zeta_i \tag{21}$$

Let

$$\lambda_{i,k+j|k} = H_i^T \textstyle\sum_{l=0}^{j-1} L_l \hat{F} \hat{w}_{k+(j-l)} - \hat{F} e_{k+j} \tag{22}$$

then for the optimization formulation we need (21) in a form:

$$H_i^T \bar{z}_{k+j|k} \leq g_i - \gamma_{i,k+j|k} \tag{23a}$$

$$\text{where } \gamma_{i,k+j|k} \text{ is s.t. } P(\lambda_{i,k+j|k} \leq \gamma_{i,k+j|k}) \geq 1 - \zeta_i \tag{23b}$$

Assume $\lambda_{i,k+j|k}$ has variance $\sigma^2_{i,k+j|k}$, which can be computed as w and e are independent and have bounded variances. Now in order to compute such a $\gamma_{i,k+j|k}$, we have the option of using one of multiple concentration inequalities:

Case A: e has unbounded support: In this case, we can use the very commonly used [27], [35] Chebyschev inequality:

$$\begin{aligned} P(\lambda_{i,k+j|k} \geq \gamma_{i,k+j|k}) &\leq \zeta_i \\ &= \sigma^2_{i,k+j|k}/(\sigma^2_{i,k+j|k} + \gamma^2_{i,k+j|k}) \end{aligned} \tag{24}$$

This gives us

$$\gamma^{\text{cheb}}_{i,k+j|k} \geq \sigma_{i,k+j|k} \sqrt{(1-\zeta_i)/\zeta_i} \tag{25}$$

We can use this $\gamma^{\text{cheb}}_{i,k+j|k}$ in (23a) to be used for the constraints of the optimization. In this paper, we do not develop the formulation for this further as strong guarantees on recursive feasibility cannot be achieved when the error distribution does not have a finite support.

Case B: e has bounded support: In this case, we have the option of using either the Hoeffding or the Bernstein concentration inequalities [35] (based on the form of the bound available). We know that $\lambda_{i,k+j|k}$ is formed by a sum of multiple independent random variables (22). Let this sum be $\lambda_{i,k+j|k} = \sum_v \lambda_v$, with λ_v generally referring to elements of the sum in (22). Since e and w have bounded support, so do the λ_v's, let their bounds be $a_v \leq \lambda \leq b_v \forall v$. Also, let their variances be σ_v^2. In this case, we can use the Hoeffding concentration inequality:

$$\begin{aligned} P(\lambda_{i,k+j|k} \geq \gamma_{i,k+j|k}) &\leq \zeta_i \\ &= \exp(-2\gamma^2_{i,k+j|k}/\textstyle\sum_v (b_v - a_v)^2) \end{aligned} \tag{26}$$

Solving this gives us a value of $\gamma_{i,k+j|k}$ to be used in the constraints for the optimization:

$$\gamma^{\text{hoeff}}_{i,k+j|k} \geq (1/\sqrt{2}) \sqrt{\textstyle\sum_v (b_v - a_v)^2 \log(1/\zeta_i)} \tag{27}$$

Another option is to use the Bernstein concentration inequality. In order to use this, define $M = \max_v b_v$ (therefore $\lambda_v \leq M \, \forall v$). With this

$$\begin{aligned} &P(\lambda_{i,k+j|k} \geq \gamma_{i,k+j|k}) \leq \zeta_i \\ = \; &\exp\{-\gamma^2_{i,k+j|k}/((2/3)M\gamma_{i,k+j|k} + 2\textstyle\sum_v \sigma_v^2)\} \end{aligned} \tag{28}$$

Define $c_1 = (-2/3)\log(1/\zeta_i)$ and $c_2 = -2\log(1/\zeta_i)\sum_v \sigma_v^2$. We can compute a $\gamma_{i,k+j|k}$ that can be used in the optimization formulation from the above equation as follows:

$$\gamma^{\text{bernst}}_{i,k+j|k} \geq (1/2)(-c_1 \pm \sqrt{c_1^2 - 4c_2}) \tag{29}$$

Combining these with (23a) results in linear constraints on the optimization variables of the SAMPC such that the chance constraints $P(H z_{k+j|k} \leq g) \geq 1 - \zeta$ are satisfied for all j in the optimization horizon.

The rest of this section will focus on the recursive feasibility of the candidate solution of (19) (as constructed for the robust case) for the case where the estimation error (and process noise) distributions have bounded support.

B. Sketch of proof for recursive feasibility

1) Validity of the applied input and next state, initial condition, dynamics: Again, via construction (as shown in the robust case), these conditions are met by the candidate solution.

2) State Constraints: Similar to the case when e came from a normal distribution, the condition for recursive feasibility takes on the form:

$$H_i^T L_j \hat{F} \hat{w}_{k+1} \leq \gamma_{i,(k+1)+j|k+1} - \gamma_{i,k+j+1|k} \tag{30}$$

This quantity can be computed offline since $\gamma_{i,(k+1)+j|k+1}$, $\gamma_{i,k+j+1|k}$ are computed apriori (in both cases where the mode remains the same from time k to $k+1$, or changes). With this, and given the samples that form the distribution of e (through the profiling step), the probability can be computed via brute force by summing over all combinations of the elements that make up the sum $H_i^T L_j \hat{F} \hat{w}_{k+1}$.

Recall that we assume bounded support of the distributions of e and w. In this case, we also know that $\hat{w}_{k+1} \in \hat{W}$. For such cases, we can prove recursive feasibility with probability 1 by using the approach presented in [27]. A sketch of this proof follows.

For simplicity of notation, denote $\gamma_{i,l+j|l} = \gamma_{i,j}$, and similarly for other variables with the same indexing that follow. First, using the bounded support of the uncertainties, we can compute:

$$\kappa_{i,j} = \max_{\hat{w}_{k+1} \in \hat{W}} H_i^T L_j \hat{F} \hat{w}_{k+1} \tag{31}$$

Now let $\tilde{\gamma}_{i,j}$ be the maximum element of the j^{th} column of the following matrix:

$$\begin{bmatrix} \gamma_{i,1} & \gamma_{i,2} & \gamma_{i,3} & \cdots \\ 0 & \gamma_{i,1} + \kappa_{i,1} & \gamma_{i,2} + \kappa_{i,2} & \cdots \\ \vdots & 0 & \gamma_{i,1} + \kappa_{i,1} + \kappa_{i,2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{32}$$

Replacing γ in (23a) by this new $\tilde{\gamma}$ gives us the constraints:

$$H_i^T \bar{z}_{k+j|k} \leq g_i - \tilde{\gamma}_{i,k+j|k} \tag{33}$$

This added conservativeness turns the recursive feasibility probability to 1. This can be observed by rewriting (30):

$$\begin{aligned} & P(H_i^T L_j \hat{F} \hat{w}_{k+1} \leq \tilde{\gamma}_{i,j} - \tilde{\gamma}_{i,j+1}) \\ &= P(H_i^T L_j \hat{F} \hat{w}_{k+1} \leq \kappa_{i,j}) && \text{(by definition of } \tilde{\gamma}_{i,j}) \\ &= 1 && \text{(by definition of } \kappa_{i,j}) \end{aligned}$$

3) Terminal Constraint: The terminal constraint is recursively feasible by the definition of the invariant set (same formulation as for the robust case) and using the fact that $\hat{w}_{k+1} \in \hat{W}$, where W can be computed because of the bounded support of the disturbances. The proof follows from the deterministic (robust) case.

This concludes the proof sketch to show that the SAMPC of (7) formulated using the set shrinking of Sec. XI-A both satisfies the chance constraints and is recursively feasible with probability 1 (i.e. proves Theorem XI.1).

XII. More details on the experimental setup

To evaluate our methodology on a real platform, we applied it to a hexrotor with the Odroid-U3 as a computation platform, running the Robot Operating System (ROS) [32] in Ubuntu. For the evaluations, the hexrotor is tasked with following the two trajectories shown in Fig. 11. As can be seen in Fig. 17, the visual odometry algorithm can occcasionaly take a long time to give a pose estimate.

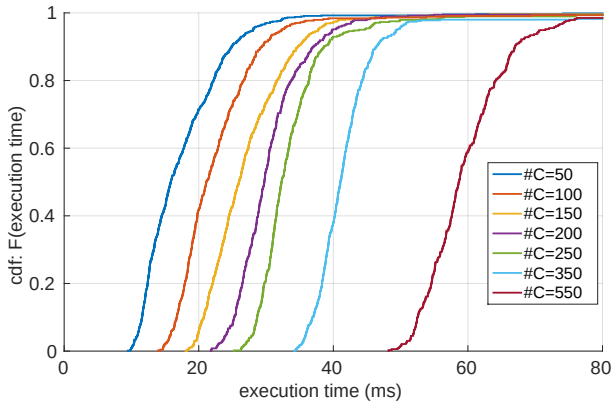

Fig. 17. Cumulative distribution of profiled execution times for visual odometry running on the Odroid-U3 for varying maximum number of corners from the SVO algorithm.

In our formulation we have assumed that the estimator satisfies the (δ, ϵ) contract requested by the controller. Thus, to ensure that the estimator fulfils the contract and that the mathematical guarantees provided by our RAMPC formulation hold, instead of using the visual odometry algorithm to fly the robot, we injected delays and errors into the measurements from Vicon, which is a high accuracy localization system. These delays and errors were selected from the Δ curve obtained by profiling the SVO algorithm (Fig 9). The hexrotor flies using these pose estimates and our control algorithms for both the position/velocity control and setting the time deadline for the next estimate. The RAMPC has the positions and velocities in the 3-axes as its references, x_k^{ref}, to track, and generates control inputs in the form of desired thrust, roll and pitch for a low-level attitude controller to track. The RAMPC and SAMPC are coded in CVXGEN [36] and the generated C Code is integrated in the ROS module for control of the hexrotor, running at 20Hz. The constraint sets for the RAMPC and SAMPC are computed offline in MATLAB and then used in CVXGEN as polyhedron type constraints. The constraint set X defines a safe set of positions and velocities in the flying area. The constraint set U of inputs keeps desired pitch and roll magnitudes less than 30 degrees and desired thrust within limits of the hex-rotor abilities.