Gaussian Process based Subsumption of a Parasitic Control Component

Allan M. Axelrod, Hassan A. Kingravi, Girish V. Chowdhary

Abstract—Many existing control architectures assume that the main control system being designed is the only controller that governs a system’s actuators. However, with the increasing availability of off-the-shelf controls packages, the number of internal unadjustable control systems is increasing. Some of these control systems may behave in parasitic way by enforcing a rigid set of behaviors that could disrupt a desired system behavior. We present a control architecture that can subsume parasitic control behavior through iteratively shaping the main control command with an intelligent feed-forward term. Our architecture requires very little prior knowledge about the subsystem whose behavior is to be subsumed, rather it relies on online learned sparsified predictive Gaussian Process (GP) models. We provide rigorous quantifiable bounds relating the sparsification of the GP to the accuracy in estimating and subsuming the parasitic subsystem. The presented subsumption architecture is realized using a variant of D-Type iterative learning control (ILC) and is validated through a series of flight tests on a Parrot AR Drone 2.0 quadrotor where the quadrotor’s sonar based altitude control loop’s behavior of maintaining a fixed altitude over ground surfaces is subsumed through a main controller via a feed-forward term.

I. INTRODUCTION

An inherent assumption in designing controllers for many systems is that the control system has full authority over the actuators. Indeed, many adaptive, reconfigurable, disturbance rejecting, probabilistic, iterative and optimal robot control architectures assume that the control system being designed is the only system that can adjust the robot’s actuators [4, 6, 7, 12, 17, 18, 20, 22, 25–27, 32, 41]. However, this assumption does not always hold in practice. For example, several aerial robots today come integrated with Commercial Off The Shelf (COTS) stability augmentation systems that the command the same actuators that the designer is wishing to control. Very little information is available about how these COTS components work, and they often do not output sufficient information to enable the designer to guess their internal state. The prevalent practice has often been to simply ignore the existence of such internal control loops, and build the control architecture around them. This approach can fail however if the internal system is preventing the robot from achieving some desired or nominal behavior. In this case, undesirable system responses may result even when no external disturbance or damage is imparted on the system, but the system responds as if there is some perturbation.

A. M. Axelrod, and A. Prof. G. V. Chowdhary are with Mechanical and Aerospace Engineering at Oklahoma State University, Stillwater OK, \{allanma, girish. chowdhary\}@okstate.edu. H.A. Kingravi was a postdoctoral associate at Oklahoma State University and is now with Pindrop security, \{hkingravi@yahoo.com\}. This work was sponsored in part by Department of Energy Award Number DE-FE0012173 and the Air Force Office of Scientific Research Award Number FA9550-14-1-0399.

Another situation where this may happen is when a Cyber attack is stealthily changing the control response. We term such perturbations imparted by parasitic unadjustable control elements as disruptions. Figure 1 illustrates that while an external disturbance affects the forces acting on the system, an internal disruption changes the control command itself.

This paper presents a method to address the relatively unaddressed problem of designing reliable control systems for robots that have parasitic control loops. One approach to tackle this problem can be to attempt to establish control authority by rewriting the actuator drivers online. Yet, establishing control authority by direct hardware level manipulation is not always necessary, nor practical in an online setting. An alternate approach that is pursued here comes from adjusting the control commands of the robot’s controller to subsume the parasitic control input. The main challenge in realizing this approach is that the behavior of the internal control loop is unknown and can be stochastic [1].

The key contribution of this paper is an online control architecture for subsuming parasitic, uncertain, unadjustable, internal control components through control command shaping. Our approach relies on building a predictive model of the internal parasitic control component. Therefore, unlike purely reactive control architectures – such as model free disturbance rejection [40] or adaptive control [37] – that would rely on modifying the control commands only after the parasitic control causes a disruption of normal operation, our approach is able to proactively anticipate the change in advance to shape the control command. To accommodate the stochasticity in the response of the parasitic component, we model its behavior as a distribution over functions using the Gaussian Process (GP) Bayesian Nonparametric (BNP) Prior [59]. Our choice of the GP-BNP is motivated from the fact that GP-BNPs are flexible enough to adapt to an unknown operational domain and can inherently handle measurement and process noise. To ensure that the architecture is implementable on resource constrained robotic hardware, we employ Csato and Opper’s sparse online GP regression.
algorithm which can automatically adjust the kernel set to meet a computational budget [10]. Another key contribution over past work on control using sparse GPs [9] in our paper is that we provide rigorous bounds on the accuracy of the online sparsified GP estimate of the underlying stochastic process. These bounds provide a quantifiable way to select the computational budget for the presented GP based subsumption architecture, and can be directly generalized to other GP based control architectures [2, 9, 13, 28, 29]. The presented architecture is validated through a series of flight tests on the Parrot AR Drone 2.0 quadrotor Unmanned Aerial Vehicle (UAV). The AR Drone is a popular and low cost aerial robot that many researchers are using for indoor flight experimentation. The AR Drone’s internal sonar based altitude control loop is designed to keep the vehicle 1 meter above any underlying surface. However, the user has no way to directly switch off or control the sonar behavior. This internal control loop can be parasitic if the intended behavior requires the UAV to fly at altitudes below 1 meter. We demonstrate the effectiveness of the presented architecture to subsume the quadrotor’s internal, unadjustable, sonar based altitude control loop.

II. RELATED WORK

Subsumption architectures deal with creating complex behaviors by combining simpler behaviors. Existing work in subsumption hopes to decouple the subsystems from one another [33, 38], however this is not always possible. The coupling problem from subsumption literature motivates our approach to negate the commands from the parasitic layer(s) of a control architecture.

There is very little direct work in designing control systems to subsume the behavior of parasitic, unadjustable, and unobservable control subsystems. Cascaded control systems have been studied [5, 6, 14, 15, 21, 23, 56], however, the underlying assumption in those works is that the designer controls the behavior of each subsystem in the cascade. Another relevant body of literature is that of stochastic disturbance rejection [4, 15, 40]. However, disturbance rejection assumes sufficient control authority to reject external forces, while we are concerned with subsumption of parasitic control competing for the same control authority. Reconfigurable control and fault detection is concerned with control strategies that are robust to sensor and actuator failure as well as structure modification. In most reconfigurable control systems, the control loops are assumed to be known and adjustable [17, 41]. However, existing reconfigurable control strategies may be ill-suited to detecting and rejecting parasitic control components or a cyber attack, such as Stuxnet, since the control code introduced by the cyber attack is unknown and it may not trigger component failure detection [24]. Our architecture can lead to reconfiguration strategies in presence of such unknown control components, provided uncorrupted feedback signals.

In Iterative Learning Control (ILC) literature, it is assumed that the control signal from a previous iteration is known and reproducible [4, 12, 16, 18, 25, 32]. Yet in the presence of a parasitic control component, only the part of the control signal is known and reproducible. However, the unknown control signal may be modeled as a distinct class of disturbance and rejected using a variant of D-type ILC [35].

III. SUBSUMPTION OF PARASITIC CONTROL

In this section, we formulate a feed-forward term for subsuming an uncertain, stochastic parasitic subsystem. The dynamics of the robot are defined as

$$\dot{x}(t) = f(t, x(t), u(t)),$$

where the state is $x(t) \in \mathbb{R}^n$, and the signal sent to the system actuator is $u(t) \in \mathbb{R}^l$. The nominal performance error of the system, $e_{nom}(t)$, is defined in terms of a reference state, $x_{ref}(t)$, and the state of the nominal system, $x(t)$, as

$$e_{nom}(t) = x_{ref}(t) - x(t).$$

For brevity, the time argument $t$ is omitted henceforth. Suppose that the system [1], receives a control input composed of a known control component $u_{act}(\cdot)$, and an unknown uncontrollable component $u_{dsr}(\cdot)$. This unknown uncontrollable component is referred to as a parasitic control component. It is assumed that the behavior of the parasitic component is stochastic and not directly measurable, that is, it is not possible to get samples of $u_{dsr}(\cdot)$. Furthermore, let $\theta_{nom}$ indicate that the parasitic control component is active, but does not cause the system to deviate from the desired nominal performance. The nominal system response $u(x)$, is

$$u(x) = u_{act}(e_{nom}) + u_{dsr}(\theta_{nom}, x).$$

Subsequently, the parasitic control changes its behavior to $\theta_{dsr}$ which causes the system to deviate from the nominal performance as

$$e_{dsr}(t) = x_{ref} - x(t).$$

and

$$u(x) = u_{act}(e_{dsr}) + u_{dsr}(\theta_{dsr}, x).$$

Iterative learning control is applied to subsume the unknown parasitic control signal at each iteration $\tau$ as

$$u(x_{\tau+1}) = u_{act}(e_{dsr_{\tau+1}}) + u_{dsr}(\theta_{dsr_{\tau}}) + \sum_{j=1}^{\tau} u_{shp_j},$$

where $u_{shp_j}$ is a feed-forward subsuming signal

$$u_{shp_j} = u_{act}(\bar{e}_{dsr_j}) + u_{act}(\bar{e}_{dsr_j} - e_{nom}),$$

and where $u_{act}(\bar{e}_{dsr_j} - e_{nom})$ is a variant of the D-type update law. The error terms for $u_{shp_j}$ are obtained using Gaussian Process regressions as in [13] and [14]. Bound on the predicted errors and a bound on the predicted residual error $\bar{e}_{dsr_j} - e_{nom}$ are provided by Theorems 1 and 3, respectively. The Bayesian feed-forward subsuming signal is then

$$u_{shp_j} = (I - \Sigma_1^2)(u_{act}(\bar{e}_{dsr_j}) + u_{act}(\bar{e}_{dsr_j} - e_{nom})).$$
where $\Sigma_1$, is calculated using [1]. The predictive uncertainty, $\Sigma_2$, ranges from the zero matrix to $I$ and is used to scale the feedforward control term. A predictive uncertainty of $I$ indicates an uncertain GP prediction, so then the shaping term is cancelled out. A predictive uncertainty of the zero matrix indicates a highly confident GP prediction, so then the shaping term is completely utilized.

IV. GP-BASED SUBSUMPTION

The form of $u_{d_{sr}}(\theta_{nom}(t), x(t))$ is potentially problematic; its response is stochastic and the range of values that it may assume is unknown. An effective strategy for modeling such stochastic responses is by utilizing a data-driven Bayesian nonparametric method; in particular, Gaussian Processes (GPs) suit this need since they have been theoretically and pragmatically demonstrated to be an effective nonparametric method for model reference adaptive control (MRAC) in the presence of stochasticity. A model of the nominal system is built by training an online GP on $\bar{e}_{nom}(t)$ and a model of the disrupted system is built by training an online GP on $e_{d_{sr}}(t)$. In what follows, the data on which the GPs are trained on is denoted by $s(t) = [y(t), e_y(t)]$, where $y(t)$ is the position in the $y$ component, and $e_y(t)$ is the error for the position in the $y$ component, and $s(t) = e_z(t)$, where $e_z(t)$ is the position error in the $z$ component. For brevity, we will sometimes drop the time dependence in the equations.

A. GP Regression

A GP is defined as a collection of random variables such that every finite subset is jointly Gaussian. The joint Gaussian form of $\Sigma$ is calculated using [1]. The predictive uncertainty, $\Sigma$, ranges from the zero matrix to $I$ and is used to scale the feedforward control term. A predictive uncertainty of $I$ indicates an uncertain GP prediction, so then the shaping term is cancelled out. A predictive uncertainty of the zero matrix indicates a highly confident GP prediction, so then the shaping term is completely utilized.

and posterior variance in GP regression can be computed as

$$p(s_{\tau+1}|R_{\tau}, s_{\tau}, r_{\tau+1}) \sim \mathcal{N}(\hat{m}_{\tau+1}, \hat{\Sigma}_{\tau+1}),$$

where

$$\hat{m}_{\tau+1} = \beta_{\tau+1}^T k_{r_{\tau+1}}, \quad \hat{\Sigma}_{\tau+1} = k_{\tau+1}^T - k_{\tau+1}^T C_{\tau} k_{r_{\tau+1}},$$

are the updated mean and covariance estimates, respectively, and where $C_{\tau} := (K(R_{\tau}, R_{\tau}) + \omega^2 I)^{-1}$, $\beta_{\tau+1} := C_{\tau} s_{\tau}$, $k_{r_{\tau+1}} = K(r_{\tau+1}, R_{\tau})$ and $k_{\tau+1} = k(r_{\tau+1}, r_{\tau+1})$.

Since both $R_{\tau}$ and $s_{\tau}$ grow with data, computing the inverse becomes computationally intractable over time. This is less of a problem for traditional GP regression applications, which often involve finite learning samples and offline learning. However, in an online setting, the linear growth in the sample set cardinality degrades computational performance. Therefore, the extension of GP regression for control requires an online method to restrict the number of data points stored for inference. Since the set $R$ generates a family of functions $F_R \subset \mathcal{H}$ whose richness characterizes the quality of the posterior inference, a natural and simple way to determine whether to add a new point to the subspace is to check how well it is approximated by the elements in $R$, using the kernel linear independence test [39]. This restricted set of selected elements, called the basis vector set, is denoted by $\mathcal{BV}$. When incorporating a new data point into the GP model, the inverse kernel matrix can be recomputed with a rank-1 update. When the budget is exceeded, a basis vector element must be removed prior to adding another element [31]. There are many schemes to remove the basis vector; in our experiments, we rely on a method that efficiently approximates the KL divergence between the current GP and the $(t + 1)$ alternative GPs missing one data point each, then deletes removes the data point with the largest KL divergence. See [39] for more details.

B. Approximation Error

In this section, review bounds quantifying the effectiveness of the approximate mean $m$ computed by the online GP algorithm. Later, these bounds will be used to provide stronger bounds for the subsampling feed forward term. Note that when a basis vector is either replaced or added to $\mathcal{BV}$, it induces a modified GP, whose mean and covariance we denote by $e(r) \sim \mathcal{GP}(\hat{m}^\sigma(r), k(r,r'))$.

Let $\tau^{-2} K_{ij} := \tau^{-2} k_{ij}(r_{i}, r_{j})$ be the kernel matrix associated to $m$ (assumed normalized), and let $|\mathcal{BV}| = p_{max}$. Then the approximate mean is associated with a kernel matrix induced by a quantization operator $\theta : \{1, \ldots, \tau\} \rightarrow \{1, \ldots, p_{max}\}$, such that $r_{i} \mapsto c_{\theta_{i}}$, where $c_{\theta_{i}} \in D_{x}$ are the set of chosen centers $\mathcal{BV}$. Let $\hat{m}^\sigma(r) = \sum_{i=1}^{\tau} \alpha_{i} k(c_{\theta_{i}}, r)$, where $\alpha = (\tilde{K} + \omega^2 I)^{-1} y$ and where $\tilde{K}_{ij} := k(c_{\theta_{i}}, c_{\theta_{j}})$.

**Theorem 1** [31](Global Approximation Theorem) Let $m^\sigma$ and $\hat{m}^\sigma$ be defined as above and let $|s|_{\infty} \leq M_{\sigma}$. Then

$$||m(r) - \hat{m}^\sigma(r)|| \leq \frac{2\sqrt{M_{\sigma}^2} \sqrt{k_{\max}}}{\omega^4} + \frac{8k_{\max} M_{\sigma}^2}{\omega^2},$$

(12)
where $k_{\text{max}} := \max_i \| \psi(r_i) - \psi(c_{\theta(i)}) \|_{\mathcal{H}}$ is the greatest kernel approximation error.

C. Application to GP-Based Subsumption

We can use the global approximation theorem to quantify the approximation error of the GPs we train on. Specifically, it’s assumed that

$$e_{\text{nom}}(t) \sim \mathcal{GP}(m_0(r), k_0(r, r'))$$
$$e_{\text{dstr}}(t) \sim \mathcal{GP}(m_1(r), k_1(r, r'))$$

(13) and (14) allow us to bound the difference between the mean outputs of the GPs, thus proving an upper bound on the residual error (i.e. $e_{\text{dstr}} - e_{\text{nom}}$) in mean square error. First, we need the following corollary.

**Corollary 2** Let $\bar{e}_{\text{nom}}(t)$ and $\bar{e}_{\text{dstr}}(t)$ be defined as in (13) and (14), and let $\bar{m}_0(t)$ and $\bar{m}_1(t)$ be the means of the GP models associated to $\bar{e}_{\text{nom}}(t)$ and $\bar{e}_{\text{dstr}}(t)$ trained using the online GP algorithm. Define

$$e_{\text{nom}}^\sigma(r) := m_0(r) - \bar{m}_0^\sigma(r)$$
$$e_{\text{dstr}}^\sigma(r) := m_1(r) - \bar{m}_1^\sigma(r).$$

Then

$$\|e_{\text{nom}}^\sigma(r)\| + \|e_{\text{dstr}}^\sigma(r)\| \leq \frac{1}{\sigma^2} \| m(r) - m(r) \|,$$

(17)

where $M^\sigma := M_0^\sigma + M_1^\sigma$, and these maximum values are associated to the corresponding GPs.

**Proof:** Apply Theorem 1 to the GP models (13) and (14) separately to prove the statement. ■

This corollary confirms the intuition that a large basis vector set results in a smaller error for the approximation, and that the difference in means lie within the bounds of the observations so far. Thus, we can now prove a theorem bounding the difference in the actual (ideal) means of the GPs (13) and (14), in terms of the data we have seen so far.

**Theorem 3** Let $\bar{e}_{\text{nom}}(t)$ and $\bar{e}_{\text{dstr}}(t)$ be defined as in (13) and (14), and let $\bar{m}_0(t)$ and $\bar{m}_1(t)$ be the means of the GP models associated to $\bar{e}_{\text{nom}}(t)$ and $\bar{e}_{\text{dstr}}(t)$ trained using the online GP algorithm. Then

$$\|m_0(r) - m_1(r)\| \leq C' + \frac{M^\sigma \kappa}{\omega^2}$$

(18)

where $C'$ is defined as the error (17) in Corollary 2 and $M^\sigma$ is also defined as in (14).

**Proof:** Using the triangle inequality,

$$\|m_0(r) - m_1(r)\| \leq \|m_0(r) - \bar{m}_0^\sigma(r)\| + \|\bar{m}_0^\sigma(r) - m_1(r)\|$$

$$\leq \|e_{\text{nom}}^\sigma(r)\| + \|e_{\text{dstr}}^\sigma(r)\| + \|m_0^\sigma(r) - \bar{m}_1^\sigma(r)\|$$

$$\leq C' + \|m_0^\sigma(r) - \bar{m}_1^\sigma(r)\|.$$

To bound the last term, note that both approximate means share the same kernel matrix $\tilde{K}$. We have

$$\bar{m}_0^\sigma(r) = \sum_{i=1}^T \tilde{a}_0 ik(c_{\theta(i)}, r)$$
$$\bar{m}_1^\sigma(r) = \sum_{i=1}^T \tilde{a}_1 ik(c_{\theta(i)}, r),$$

where

$$\tilde{a}_0 := \left( \tau^{-2} \tilde{K} + \omega^2 I \right)^{-1} s_0^\sigma$$
$$\tilde{a}_1 := \left( \tau^{-2} \tilde{K} + \omega^2 I \right)^{-1} s_1^\sigma.$$

Using ideas from the proof of Theorem 1, we have

$$\|\bar{a}_0 - \bar{a}_1\| \leq \frac{\|s_0^\sigma - s_1^\sigma\|}{\omega^2}.$$

Then

$$\|m(r) - \bar{m}^\sigma(r)\| \leq \|\tilde{a}_0^T - \tilde{a}_1^T\| \|\tilde{K}r\|$$

$$\leq \|\tilde{a}_0^T - \tilde{a}_1^T\| \|\tilde{K}\|$$

$$\leq \|\tilde{K}\| \|\tilde{a}_1\|$$

$$\leq \frac{M^\sigma \kappa}{\omega^2}.$$

The above theorem allows us to compute a bound between the means of the GPs modeling the nominal and disruption signals, as a function of the observations so far. Thus, we can predict not only the expected error for the next learning iteration, but we also can predict the expected residual error in the next learning iteration as well.

V. RESULTS

A. Description of the Experiment

The Parrot AR Drone 2.0 quadrotor [3] is designed to be teleoperated and it is compatible with the Robotics Operating System (ROS) [30]. A ROS python (ROSPY) infrastructure was developed to enable autonomous flight within a motion-capture testbed. The quadrotor uses a downward-facing sonar array to enforce an operational altitude of 1 meter off the ground, and the code to measure or adjust the sonar-based control component is not accessible through ROSPY. A nominal proportional altitude controller was implemented on ROSPY. The quadrotor’s internal control loop however competes with this nominal controller when tracking trajectories below 1 meter altitude. This behavior of the quadrotor’s sonar based altitude control loop therefore can be considered parasitic when tracking trajectories below 1 meter over the ground or solid objects, resulting in unreliable and oscillatory quadrotor performance.

The presented architecture was verified using waypoint flights from waypoint A to B. The single-dimensional A-to-B trajectory flight tests were selected since a disruption in the z-direction also affects the translational y-directions of the quadrotor due to couple dynamics. Each of the final data sets are representative of 20 consecutive A-to-B trajectory flight tests. For the nominal data set, the quadrotor executed 20 flights over flat ground as an online Gaussian Process learned z-position-error based on the y-position and y-position-error. For the disruption data set, the quadrotor executed the 20
A-to-B trajectory flight tests as an online Gaussian Process trained on the y-position and y-position-error to learn the z-position-error. However, a box was placed with one edge at -18.4 cm and the other edge at 30.6 cm in terms of the testbed y-coordinate grid. The box dimensions 72cm x 51cm x 21cm where the 72cm dimension was centered about the y-axis of the testbed and the 51cm dimension was aligned parallel to the y-axis. So as the quadrotor passed over the box, the sonar output changed and the quadrotor altitude was adversely affected. However, the effect was not immediate as some linear data fusion process in the quadrotor mitigated the immediate sonar response of flying over the box [3].

Since Csato and Opper’s sparsified online GP regression was not yet executable in python, the quadrotor executed 20 A-to-B flight tests using (7) for the subsumed data set, where the trajectory was generated offline using Csato and Opper’s sparsified GP regression in MATLAB.

B. Discussion of Results

The quadrotor was commanded to follow a path between the y-position -1 meter and the y-position 1 meter. The 3-dimensional Gaussian Process regressions used are shown from a 2-dimensional perspective in figures 3a-3f. The large variance in the data was a result of the selected platform. There have been numerous attempts to demonstrate autonomous control of the Parrot AR Drone 2.0 quadrotor and performance issues were common [11, 19, 34]. It is worth noting that the difficulty involved in developing a quality ROS or ROSPY controller for the Parrot AR Drone 2.0 motivated the initial platform selection. Thus, a GP was trained on the batch of 20 nominal runs and another GP was trained on the batch of 20 disrupted runs. The resulting regressions were used to demonstrate the impact of a single learning iteration of the subsuming feed-forward term. The subsumed response was recorded across 20 flights.

The performance of the nominal, disrupted and subsumed responses are compared in figures 2c and 2d. The subsumed response shows that the disruption caused by the sonar has been reduced across the batched ILC. More interestingly, the changes in the nominal response due to the sonar are also mitigated by the subsumed response; i.e., the subsumed response outperforms the nominal and disruption responses before the major sonar disruption is encountered. Had the subsumed response been separated into nominal and disruption test regimes, this finding would be more evident. However, the gathered experimental data is sufficient to demonstrate the aforementioned result.

VI. CONCLUSION

A novel control architecture for subsuming parasitic control components using a variant of D-type ILC was presented. The architecture leveraged past error terms and knowledge of the actionable control component to reduce the impact of the parasitic control component on the overall system performance. Sparsified Gaussian Processes (GP) were leveraged to learn a predictive model based on the past error terms to subsume the parasitic control component for online implementation. Rigorous quantifiable bounds related the sparsification of the GP to the accuracy in estimating and subsuming the parasitic subsystem.

REFERENCES


