2024 IEEE International Conference on Robotics and Automation (ICRA)
May 13-17, 2024. Yokohama, Japan

Safety Verification of Closed-loop Control System with Anytime Perception

Lipsy Gupta[1], Jahid Chowdhury Choton[2] and Pavithra Prabhakar[3]

2024 IEEE International Conference on Robotics and Automation (ICRA) | 979-8-3503-8457-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICRA57147.2024.10611044

***Abstract*— In this paper, we consider the problem of safety analysis of a closed-loop control system with anytime perception sensor. We formalize the framework and present a general procedure for safety analysis using reachable set computation. We instantiate the procedure for two concrete classes, namely, the classical discrete-time linear system with linear state feedback controller and an extension with variable update rates. We present an exact computational method based on polyhedral manipulations for the first class and an over-approximate method for the second class. Our experimental results demonstrate the feasibility of the approach.**

I. Introduction

Robotic systems are being increasingly deployed in human environments, and hence, ensuring their safe functioning is of utmost importance. These systems consist of a controller interacting with the robot dynamics to make decisions while navigating the workspace [2], [23]. This had led to the investigation of rigorous methods to provide guarantees on the desired requirements of these control systems [12], [21].

A typical architecture for a software controlled robotic systems consists of a plant modeling the robot dynamics, sensors measuring the state of the system and a controller providing inputs to the plant based on the current sensed state. With the increase in the complexity of the robotic tasks, computationally expensive sensors such as LIDARs and cameras are being used. There is a vast range of sensors to choose from in terms of precision, however, these come with varying latencies. Specifically, a high precision sensor has large latency in computing the sensed value. Hence, anytime sensing has been proposed as a novel paradigm that allows to exploit the precision-latency trade-off and has been shown to enhance controller performance [25]. The objective of this work is to formalize the closed-loop control framework with anytime sensing and explore safety verification algorithms.

First, we present the closed-loop control system with anytime sensing that extends the classical sensor with anytime sensor and the classical controller with one that provides both the control input and the update rate. More precisely, the standard sensor that outputs an estimate $\hat{x}$ of the state x with error ϵ, is replaced by an anytime sensor that outputs the estimate $\hat{x}$ with an error ϵ that depends on a given latency (upper bounded by the update rate). Next, we present a general procedure for safety analysis of these systems based on reachable set computation. We consider two specific classes of systems, namely, discrete-time linear systems with linear state feedback controllers and fixed precision sensors, and an extension of those to variable update rates and anytime sensors. The first class of systems is relatively simpler to analyze and we provide an instantiation of our general procedure using polyhedral manipulation that computes the exact reachable set. For the second class of systems, we provide an algorithm that over-approximates the reachable set. We ran our experiments on a 2-dimensional toy example as well as a 4-dimensional robotic control example. Our experiments demonstrate the feasibility of the safety analysis approach.

The authors are with the Department of Computer Science at the Kansas State University.

II. Related Work

The notion of anytime algorithm has been studied and refers to an algorithm that provides an output at any given point during its run-time, and is in general, expected to provide better quality answers when given more time [7]. The idea of anytime algorithms has been used widely in several domains such as in graph search [20] and motion planning [24]. It has recently been gaining attention in the area of control theory [10], [11], [28] with the introduction of anytime perception modules, and has been shown to result in improved control performance [25].

With the increasing use of anytime perception in autonomous robot control, safety verification of systems employing them has become important. There is extensive work in the area of safety verification of closed-loop control systems [9], [27], [31] using reachable set computation. Since the problem of computing reachable sets is undecidable for most dynamical systems, various approaches for over-approximation of these reachable sets have been explored including abstraction-refinement based approaches and symbolic approaches based on a variety of data structures [3], [4], [8], [13], [15]–[18], [26], [30]. There has also been some work addressing reachable set computation for dynamical systems with uncertainties [1], [14], [19], [22]. While these systems model uncertainty arising from the sensors, they are not sufficient to capture the anytime sensing aspects. In this work, we present a rigorous modeling and analysis framework for closed-loop system analysis with anytime sensing.

III. Preliminaries

We denote the set of natural numbers by $\mathbb{N}$, the set of real numbers by $\mathbb{R}$, and the set of non-negative reals by $\mathbb{R}_{\geq 0}$. Let $n \in \mathbb{N}$. For $x \in \mathbb{R}^n$, let x_i denote the projection of x onto i^{th} component, that is, $x = (x_1, x_2, ..., x_n)$. For $x \in \mathbb{R}^n$, the infinity norm on x is given by $\|x\| = \max_{1 \leq i \leq n} |x_i|$, and

the Euclidean norm is given by $\|x\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$. Given a set $X \subseteq \mathbb{R}^n$ and $v \in \mathbb{R}^n$ with $v_i \geq 0$, for all $1 \leq i \leq n$, the set $Box(X, v) = \{x + \alpha \,|\, x \in X, \alpha \in \mathbb{R}^n, |\alpha_i| < v_i, 1 \leq i \leq n\}$ represents the box around X with respect to the vector v. For $x \in \mathbb{R}$, the notation $vec_n(x)$ denotes the vector $(x, \cdots, x) \in \mathbb{R}^n$. The Minkowski sum of the sets $X_1, \cdots, X_k \subseteq \mathbb{R}^n$ is given by $X_1 \oplus \cdots \oplus X_k = \{x_1 + \cdots + x_k \,|\, x_i \in X_i, 1 \leq i \leq k\}$. For two sets $A, B \in \mathbb{R}^n$, the minimum distance between them is defined as $Dis_{min}(A, B) = \min\{\|a - b\|_2 \,|\, a \in A, b \in B\}$, and the asymmetric Hausdorff distance from A to B is defined as $Dis_{AH}(A, B) = \sup_{a \in A} \inf_{b \in B} \|a - b\|_2$. Let $G : X \to A \times B$ be a function, then the function $G_1 : X \to A$ is the projection of $G(x)$ on the set A for all $x \in X$, and similarly we have another function G_2. Any subset of the Euclidean space considered in the paper is assumed to be closed and bounded.

IV. Feedback Control System with Anytime Sensing

In this section, we formalize the notion of closed-loop control system with anytime sensing. It is similar to the traditional closed-loop feedback control system with a plant, a sensor and a controller, wherein the standard sensor is replaced by an anytime sensor as shown in Figure 1. In the classical setting, the sensor outputs an approximation of the sensed value with some precision. An anytime sensor has the ability to adjust the precision based on user provided latency, that is, given a time representing the latency (often dictated by the update rate), the anytime sensor outputs the sensed value within that time and a precision that depends on the latency. That is, if given a larger time, the sensor would output a more precise value than if given a smaller time. So, the controller outputs both the input and the update rate to be used by the plant to update the state, wherein the update rate is also input to the sensor as the latency.

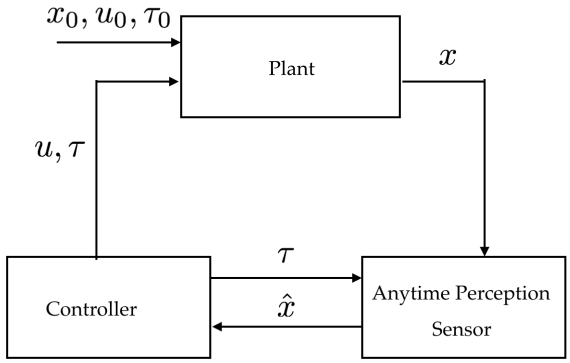

Fig. 1. Feedback Control System with Anytime Sensing

A. Syntax

A feedback control system with anytime sensing is formally defined as follows.

Definition 1: An (n, m)-dimensional feedback control system with anytime sensing is a tuple $S = (F, P, G)$, where

- $F : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}^n$ models the dynamics of the plant, $n \in \mathbb{N}$ and $m \in \mathbb{N}$ represent the dimesnions of the state space and the input vector respectively.
- $P : \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0}$ is a decreasing function modeling the relations between latency and error of the anytime sensor.
- $G : \mathbb{R}^n \longrightarrow \mathbb{R}^m \times \mathbb{R}_{\geq 0}$ represents the controller, wherein G_1 generates the control input and G_2 generates the update rate for the plant.

The function F takes as input a state and an input along with a time period for which the input needs to be applied for, and outputs the state of the plant at the end of that time period. The anytime perception sensor is characterized by its precision function P, which takes latency as its input and outputs a precision on its approximation of the state. In particular, if more time is given to the sensor to sense the value of the system, it will give lesser approximation error, and hence it's a decreasing function. The sensor is called anytime perception sensor as it can be asked to sense the state of the system within a given time. The feedback controller takes the approximated state as its input, and outputs the next input value to the system along with an update rate, that is, a time duration for which it needs to be applied. Note that the update rate and the latency is same so as so as to command the sensor to give the next approximated state space value at the end of that time period.

B. Semantics

Next we define the semantics of a feedback control system with anytime sensing in terms of its executions. Let $S = (F, P, G)$ be an (n, m)-dimensional feedback control system with anytime sensing. The system executes as follows. Starting from a state x with input u and update rate τ, the plant updates the state to $x' = F(x, u, \tau)$. Next, the sensor outputs an approximated value $\hat{x}$ of the updated state x' which is guaranteed to be within an error bound of $P(\tau)$. Then, the feedback controller G takes as input the value $\hat{x}$ and outputs an input value u' to be applied for the time period τ'. The computation then repeats with the new values x', u' and τ'.

Let $X_0 \subseteq \mathbb{R}^n, U_0 \subseteq \mathbb{R}^m$, and $T_0 \subseteq \mathbb{R}$ be the sets of initial states, initial inputs, and initial update time values, respectively. A sequence

$$\eta = x_0, x_1, x_2, \cdots$$

is called an execution of S from X_0, U_0 and T_0 if there exist sequences $u_0, u_1, u_2, ...$, and $\tau_0, \tau_1, \tau_2, ...$ such that the following holds.

1) $x_0 \in X_0$, $u_0 \in U_0$, and $\tau_0 \in T_0$.
2) For each $i \geq 1$,
   - $x_i = F(x_{i-1}, u_{i-1}, \tau_{i-1})$, and
   - $(u_i, \tau_i) = G(\hat{x}_i)$ for some $\hat{x}_i \in Box(\{x_i\}, \alpha)$, where $\alpha = vec_n(P(\tau_{i-1}))$

For each $i > 0$, let $t_i = \sum_{j=0}^{i-1} \tau_j$. Thus for each $i > 0$, x_i is the state of the system at time t_i, $\hat{x}_i$ represents the

approximated state at time t_i, and u_i is the input vector given to the plant to evolve from time t_i to t_{i+1}. For an execution η, we use $\eta[i]$ to denote its i^{th} element, that is, $\eta[i] = x_i$.

C. Safety Verification Problem

We are interested in verifying the safety of the feedback control system with anytime sensing with respect to collision with a given unsafe set. This is accomplished by computing the reachable states of the system and check that it does not intersect with the unsafe set. Hence, let us first define the set of states that a feedback control system with anytime sensing can reach with a certain number of iterations of the loop. Let us fix the following notation for the rest of the paper. Let $S = (F, P, G)$ be an (n, m)-dimensional feedback control system with anytime sensing. Let X_0, U_0 and T_0 be the sets of initial states, initial input vectors, and initial time values respectively.

Definition 2: For each $i \geq 1$, the set

$$Reach_S(X_0, U_0, T_0, i) = \{\eta[i] : \eta \text{ is an execution of } S\},$$

is called the reachable set of the system within i steps starting from the initial conditions (X_0, U_0, T_0).

Problem 1: Safety Problem: Given S, X_0, U_0 and T_0, an unsafe set $W \subseteq \mathbb{R}^n$, and $k \in \mathbb{N}$, the safety problem is to check whether $Reach_S(X_0, U_0, T_0, i) \cap W = \emptyset$ for all $1 \leq i \leq k$.

V. Special cases of feedback control system with anytime sensing

In this section, we consider two concrete classes of the feedback control systems with anytime sensing, namely, the classical discrete-time linear systems with constant update rate and an extension of that to a variable update rate setting.

A. Discrete-time linear system

Recall that the plant in a discrete-time linear system is given by:

$$x(t+1) = Ax(t) + Bu(t) \qquad x(0) \in X_0,\ u(0) \in U_0. \tag{1}$$

Here, the update rate is always a constant unit time, and the matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are time invariant. The corresponding plant function is $F(x, u, \tau) = Ax + Bu$. Since, the update rate is constant, we now have a traditional sensor with a time-independent precision, that is, $P(\tau) = \epsilon$. The linear state feedback controller is given by

$$u(t) = K\hat{x}(t),$$

and corresponds to $G(x) = (Kx, 1)$, where 1 represents the fact that the update rate is always 1.

B. Linear system with variable update rates

Next, we consider a model for a system with variable update rates which can capture approximations of linear continuous-time systems. The plant dynamics is given by:

$$x(t_0 + \tau) = A\tau x(t_0) + B\tau u(t_0) + Cx(t_0) + Du(t_0) + E \tag{2}$$

The matrices $A, C \in \mathbb{R}^{n\times n}, E \in \mathbb{R}^{n\times 1}$, and $B, D \in \mathbb{R}^{n\times m}$ are time invariant. We note that the above equation models an approximation of a continuous time linear system: $\dot{x}(t) = Ax(t)+Bu(t)$, $x(0) \in X_0,\ u(0) \in U_0$, the solution of which is given by:

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-s)}Bu(s)ds$$

Now, if we assume u to be a constant vector from time 0 to t, the above solution can be approximated by $x(t) \approx I.x(0) + Atx(0) + Btu(0) + R$ for some matrix $R \in \mathbb{R}^{n\times 1}$. Thus we consider a more general form of such a dynamics in Equation 2.

Thus we consider an (n, m)-dimensional feedback control system with anytime sensing $S = (F, P, G)$, where $F : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}^n$ is given by $F(x, u, \tau) = A\tau x + B\tau u + Cx + Du + E$. Recall that the function P models the error given the latency, so one such natural way to define P is as $P(\tau) = \frac{c}{\tau}$ for some constant $c > 0$.

The feedback controller outputs the control input and the update rate. We assume that the control input is given by a linear function as before. Note that the update rate needs to be small if the current state of the system is close to the unsafe set W, while it could be large if far from the unsafe set. Hence, we define $G : \mathbb{R}^n \to \mathbb{R}^m \times \mathbb{R}_{\geq 0}$ to be $G(x) = (G_1(x), G_2(x))$ where $G_1(x) = Kx$ and $G_2(x) = \min\{rm_x, b\}$ for some constants $b, r > 0$, and $m_x = \min\{\|x - w\|_2 \,|\, w \in W\}$ is the minimum distance of the state x from the unsafe set $W \subseteq \mathbb{R}^n$. Note that we bound the update rate from above by some constant b.

VI. Safety Verification Algorithms

In this section, we present the safety verification framework for a feedback control system with anytime sensing by computing its reachable set. We begin with a generic algorithm for the verification of any such system, and then we instantiate this algorithm to obtain the specific safety verification algorithms for the two special cases that we considered in the previous section. Given initial sets of states, inputs, and update rates, each algorithm in its i^{th} iteration, computes the reachable set formed using an execution of length i or an over-approximation of this reachable set, and checks if it intersects the unsafe state. In case it does, the algorithm aborts, otherwise it gives the reachable set or its over-approximation as its output.

Algorithm 1: This algorithm takes the following inputs: an (n, m)-dimensional anytime feedback system $S = (F, P, G)$, a set $D_0 \subseteq \mathbb{R}^{n+m+1}$ representing a set of triples of state, input and update rate values (x, u, t), an unsafe set W, and the number of iterations k of the feedback loop. The broad idea is to propagate the set D_0 through the plant, anytime sensor and the controller and repeat these steps k times. Recall from Figure 1 that in any iteration, first, the plant updates its state, that is, outputs $x' = F(x, u, t)$. Next the sensor outputs the sensed state $\hat{x}$, however, this depends on the current update rate in addition to the current state x'. So, just computing the set of states X' corresponding to x' without keep track of the

Algorithm 1 Generic Anytime Perception Algorithm

1: **Input:** $D_0, W, S = (F, P, G), k$
2: **Output:** X_k
3: $D \leftarrow D_0$
4: **for** $i = 1 : k$ **do**
5: $\quad X_{time} = \{(F(x, u, t), t) \mid (x, u, t) \in D\}$
6: $\quad \widehat{X} = \{(x, x + \alpha) \mid (x, t) \in X_{time}, \|\alpha\| \leq P(t)\}$
7: $\quad D = \{(x, G_1(\hat{x}), G_2(\hat{x})) \mid (x, \hat{x}) \in \widehat{X}\}$
8: $\quad X = \{x \mid (x, y, z) \in D\}$
9: $\quad$ **if** $X \cap W \neq \emptyset$ **then** Abort
10: $\quad$ **end if**
11: **end for**
12: $X_k = X$
13: **return** X_k

Algorithm 2 Anytime Perception Algorithm for Discrete-time Linear System (Exact)

1: **Input:** $P_0, W, A, B, K, \epsilon, k$
2: **Output:** X_k
3: $C = \begin{bmatrix} A & B \end{bmatrix}$, $K' = \begin{bmatrix} I & 0 \\ 0 & K \end{bmatrix}$
4: $P \leftarrow P_0$
5: **for** $i = 1 : n$ **do**
6: $\quad X = LT(C, P)$
7: $\quad \widehat{X} = Box(Twice(X), \beta_\epsilon)$
8: $\quad P = LT(K', \widehat{X})$
9: $\quad$ **if** $X \cap W \neq \emptyset$ **then** Abort
10: $\quad$ **end if**
11: **end for**
12: $X_k = X$
13: **return** X_k

τs that lead to x' will not be sufficient to propagate the reach set through the anytime sensor. Hence, in Line 5, we compute the set of (x', t) pairs resulting from propagation through the plant. Note that the sensor then outputs the sensed state $\hat{x}$ corresponding to (x', t), and the controller uses $\hat{x}$ to output the (u', τ'), that is, the control input and update rate for the next iteration. Note that for the propagation through the plant to be exact for the next iteration, it is imperative to know which x' led to (u', τ'). Hence, this information needs to be propagated, first through the sensor by computing the pairs $(x', \hat{x})$ and then through the controller by computing the triples (x', u', τ'). This is reflected in the computations in Lines 6 and 7. The next theorem establishes that the set output by this algorithm is the reach set of the system after k iterations of the loop.

Theorem 1: Algorithm 1 computes the set $Reach_S(D_0, k)$.

Note that the above algorithm defines the sets that need to be computed, however, does not provide explicit procedures to compute them. Next, we instantiate Algorithm 1 to compute reachable sets of the two special cases of the feedback control systems considered in Section V-A and Section V-B. We begin with the discrete-time linear system.

Algorithm 2: Note that for this class of systems the update rate is constant and the sensor is the standard sensor, that is, $P(\tau)$ is constant. Therefore, the input set is $P_0 \subseteq \mathbb{R}^{m+n}$ rather than $\mathbb{R}^{m+n+1}$ and is assumed to be a polyhedron. Further, we don't need to keep track of time factor in Line 5 of Algorithm 1 and the function F is a linear transformation on the set of pairs $(x, u) \in P \subseteq \mathbb{R}^{m+n}$. Hence, we compute the set X, which is obtained by the linear transformation $LT(C, P)$ defined by the multiplication of the matrix C with the polyhedron P, where $C = \begin{bmatrix} A & B \end{bmatrix} \in \mathbb{R}^{n \times (n+m)}$. Since the operation $LT(C, P)$ is a linear transformation on a polyhedron, X is also a polyhedron.

The next step consists of computing (x, x'), where x' is any state within $\epsilon = P(1)$ of x, corresponding to Line 6 of Algorithm 1. Now, this step is accomplished by first stacking a state twice to form the set of pairs (x, x) for $x \in X$, and then expanding the second part of every stacked state by ϵ. Stacking the state vector twice is obtained by the linear transformation $Twice(X) = MX$ where $M = \begin{bmatrix} I \\ I \end{bmatrix} \in \mathbb{R}^{2n \times n}$. Bloating the second half of every stacked state by ϵ, that is, the set $Box(Twice(X), \beta_\epsilon)$, where $(\beta_\epsilon)_i = 0$ for $1 \leq i \leq n$, and $(\beta_\epsilon)_i = \epsilon$ for $n + 1 \leq i \leq 2n$, corresponds to the Minkowski sum of $Twice(X)$ with $Box(0, \beta_\epsilon)$. This whole step is summarized in Line 7 of Algorithm 2.

Finally, we compute the polyhedron P that captures (x', u) pairs by the linear transformation $LT(K', \widehat{X})$, where $K' = \begin{bmatrix} I & 0 \\ 0 & K \end{bmatrix} \in \mathbb{R}^{(n+m) \times 2n}$. Here, the polyhedron P stores all states and all the possible inputs corresponding to each state. This step corresponds to the computation of the set D in Algorithm 1. Since, each step of this algorithm computes the sets outlined in Algorithm 1 exactly, we obtain the following result.

Theorem 2: For an initial polyhedron $P_0 \subseteq \mathbb{R}^{n+m}$, an unsafe polyhedron W, and $k \in \mathbb{N}$, Algorithm 2 computes the exact set $Reach_S(P_0, k)$.

Next we instantiate Algorithm 1 to compute the reachable sets of the feedback system defined in Section V-B. Note that the dynamics has terms that involve product of x and u with τ. Hence, the system is not linear in the traditional sense. While non-linear (polynomial) theory of reals can be use to compute the exact set, it will not be efficient. Hence, we present Algorithm 3 that only uses polyhedral manipulations to efficiently compute an over-approximation of the reachable set. Before studying Algorithm 3, let's see the following proposition which helps in over-approximating the set of states.

Algorithm 3: We represent D_0 and the subsequent sets D in Algorithm 1 through an approximate set that is a Cartesian product of P_0 representing a set of pairs of states and inputs and T_0 representing a set of update rates. We maintain this data structure through the iteration. Note that if $x \in \mathbb{R}^n, u \in \mathbb{R}^m$, $0 \leq l' \leq u'$, and $e(\tau) = A\tau x + B\tau u + Cx + Du + E$. Then $\{e(\tau) : \tau \in [l', u']\} = \text{Convex Hull}\{e(l'), e(u')\}$. Thus, in each iteration, given a set of update rates as an interval $[l, u]$, we compute the state part of X_{time} by

Algorithm 3 Anytime Perception Algorithm for Discrete-time System with update rates (Over-approximate)

1: **Input:** $P_0, T_0 = [l_0, u_0], W, A, B, C, D, E, K, c, r, b, k$
2: **Output:** X_k
3: $K' = \begin{bmatrix} I & 0 \\ 0 & K \end{bmatrix}$
4: $P \leftarrow P_0 \times \{1\}, l \leftarrow l_0, u \leftarrow u_0$
5: **for** $i = 1 : n$ **do**
6: $\quad J_1 = \begin{bmatrix} Al + C & Bl + D & E \end{bmatrix}$
7: $\quad J_2 = \begin{bmatrix} Au + C & Bu + D & E \end{bmatrix}$
8: $\quad X_1 = LT(J_1, P)$,
9: $\quad X_2 = LT(J_2, P)$
10: $\quad X = CH(X_1, X_2)$
11: $\quad \epsilon = \frac{c}{l}$
12: $\quad \widehat{X} = Box(Twice(X), \beta_\epsilon)$
13: $\quad P_1 = LT(K', \widehat{X})$
14: $\quad P = P_1 \times \{1\}$
15: $\quad X_{app} = Box(X, \alpha_\epsilon)$
16: $\quad l = \min\{b, r.Dis_{min}(X_{app}, W)\}$
17: $\quad u = \min\{b, r.Dis_{AH}(X_{app}, W)\}$
18: $\quad$ **if** $X \cap W \neq \emptyset$ **then** Abort
19: $\quad$ **end if**
20: **end for**
21: $X_k = X$
22: **return** X_k

computing the convex hull of two reach sets for Equation 2 corresponding to $\tau = l$ and $\tau = u$. This is accomplished in Lines 6-10 of Algorithm 3.

Next, we need to compute the $\widehat{X}$. This is done similar to Algorithm 2. While the exact algorithm would require computing for every $\tau \in [l, u]$, a corresponding ϵ and bloating the state set appropriately, we again approximate this by taking a conservative value of ϵ corresponding to l. Recall, for this system, we have $P(\tau) = \frac{c}{\tau}$ for some constant $c > 0$. Since $\frac{c}{l} > \frac{c}{u}$, $\epsilon = \frac{c}{l}$ corresponds to the maximum bloating and our results will be sound (over-approximate). This is performed in Lines 11-12 of Algorithm 3.

The control input computation is similar to that of Algorithm 2 and is done in Lines 13-14 of Algorithm 3. The Cartesian Prodcut with $\{1\}$ is a technicality required to handle the constant E in the dynamics by expanding every vector with a constant 1. Finally, we need to compute the set of update rates. Note that the lowest update rate l corresponds to the state x' that is closest to the unsafe set. This distance is computed by $Dis_{min}(X_{app}, W)$, where X_{app} is the set of all approximated state values given by the sensor, i.e., the set of second halves of the vectors in $\widehat{X}$. The highest update rate u corresponds to the x' that is farthest from the unsafe set, and this value is given by $Dis_{AH}(X_{app}, W)$. The computation corresponding to the set of updates rates represented as an interval $[l, u]$ is summarized in Lines 15-17 of Algorithm 3, where minimum with respect to b is taken to ensure that the update rate has a ceiling of b.

Note that when we compute the reachable sets, we don't keep track of which states and inputs correspond to which update rates; hence, the sets computed in each step of Algorithm 3 over-approximate the coresponding sets in Algorithm 1. This is captured in the following theorem.

Theorem 3: For an initial polyhedra $P_0 \subseteq \mathbb{R}^{n+m}$, $T_0 \subseteq \mathbb{R}$, an unsafe polyhedron W, and $k \in \mathbb{N}$, Algorithm 3 computes an over-approximation of the reach set $Reach_S(P_0, k)$, that is, $Reach_S(P_0, k) \subseteq X_k$.

VII. Experiments

In this section, we present the evaluation of Algorithm 2 and Algorithm 3 on two case studies. The first one is a toy example of 2-dimensional closed loop system, and the second one is a 4-dimensional closed loop system which is a balance system of a two wheeled robot with closed-loop control [5]. We implemented our algorithms in a Python toolbox; all the experiments were run on a Macbook Pro 2021 machine (OS: MacOS Ventura 13.5.2, Processor: Apple M1 Pro, RAM: 16GB). We considered a vertex-based representation for the input polyhedral sets, and implemented linear transformation and Minkowski sum on this representation. Finally, we use Parma Polyhedra Library [6] on top of the mathematical software tool Sagemath [29] to report the number of vertices and the volume of the resulting reachable polyhedral sets.

Recall that in Algorithm 3, we need to calculate the minimum distance between two polyhedra. We calculate an approximation of this value by computing the minimum distance among all the vertices of the two polyhedra. Since all our polyhedra are convex and bounded and are in the Euclidean space, we compute the distance $Dis_{AH}(X_{app}, W)$ by computing $\min_{w \in W} \max_{x \in X_{app}} \|w - x\|_2$. The next result says that the $\max$ in the above expression is achieved on at least one of the vertices of X_{app}. Thus for each vertex of W, we calculate its maximum distance from all the vertices of X_{app}, and finally we take minimum of all the obtained values corresponding to each vertex of W.

Proposition 1: Let $x \in \mathbb{R}^n$, and $P \subseteq \mathbb{R}^n$ be a polyhedron such that $x \notin P$. The maximum distance from x to P, that is, $\max_{p \in P} \|x - p\|_2$ is given by $\|p - v\|_2$ for some vertex v of P.

The first example we consider is a toy closed loop system, where the state X belongs to 2-dimensional space and the input U belongs to 1-dimensional space. For stability, we choose the feedback controller K such that the eigenvalues of $A - BK$ are less than 0. We took a polyhedron W in the state-space as the unsafe region. The sensor function P for implementing Algorithm 3 is taken to be $P(\tau) = 500/\tau$.

The second example is a balance system of a two wheeled robot with closed-loop control [5]. A balance system is a mechanical system in which the center of mass is balanced above a pivot point. The system tries to stabilize an object on top of its body such that the object does not fall to the ground. The dynamics of the system is given by:

$$\begin{bmatrix} (M+m) & -ml\cos\theta \\ -ml\cos\theta & (J+ml^2) \end{bmatrix} \begin{bmatrix} \ddot{p} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} c\dot{p} + ml\sin\theta\dot{\theta}^2 \\ \gamma\dot{\theta} - mgl\sin\theta \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}$$

Here, p and $\dot{p}$ are the state variables that represent the position and velocity of the base of the system, θ and $\dot{\theta}$

TABLE I

ALGORITHM 2 ON THE 2D CLOSED-LOOP SYSTEM

Experiment 1 (ϵ = 0)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	18	6.286159277
5	4	18	6.357446909
10	4	18	6.438996077
50	4	18	7.113317251
Experiment 2 (ϵ = 0.2)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	18	0.004370928
3	64	281.68	0.065777779
5	1024	586.32	0.310337067
8	65536	1201.36	23.40886497
Experiment 3 (ϵ = 2)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	18	0.00424099
3	64	4498	0.065255165
5	1024	13074	0.324712038
8	65536	33970	24.02043891

TABLE II

ALGORITHM 2 ON THE 4D BALANCE SYSTEM

Experiment 1 (ϵ = 0)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	112	0.022784948
5	4	112	0.088227034
10	4	112	0.161881685
50	4	112	0.780392885
Experiment 2 (ϵ = 0.2)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	112	0.79925704
2	16	269.22	0.83965683
3	256	628.18	1.201859951
4	4096	2054.44	6.627283812
Experiment 3 (ϵ = 2)			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	4	112	0.02553606
2	16	1380.87	0.065253019
3	256	6428.18	0.429074049
4	4096	30337.77	7.774876118

represent the angle and angular rate of the object above the base, M is the mass of the base, m and J are the mass and moment of inertia of the object to be balanced, l is the distance from the base to the center of mass, c and γ are coefficients of the viscous friction, and g is the acceleration due to gravity. The above equations when represented in the state space form correspond to a system with 4-dimensional state space and 1-dimensional input vector.

Table I summarizes the results of running Algorithm 2 on the 2-dimensional closed loop system for varying values of sensor error ϵ, and varying number of iterations of the closed-loop. The columns Num_Ver(X), Volume(X), and Runtime report the number of vertices, the volume and the computation time of the polyhedron representing the reachable set of states. For $\epsilon = 0$, we do not need to perform Minkowski sum operation with a box, hence, the number of vertices in the reach set remains constant. For all $\epsilon > 0$, the number of vertices of the reachable set increases with increase in the number of iterations due to repeated Minkowski sum operations with a box, however, it is still independent of the specific value of ϵ, and hence, the specific value of ϵ does not affect the runtime drastically. On the other hand, the volume of the reachable set is affected by the specific value of the ϵ. Table II summarizes the results of running Algorithm 2 on 4-dimensional balance system. We observe similar trends as in Table I, however, due to the increased dimension of this system, the computations are more expensive.

Table III summarizes the results of running Algorithm 3 on both the 2-dimensional and 4-dimensional systems. We do not compare with different values of ϵ since the error bound dynamically changes during the execution of the algorithm. Note that the volumes of reachable set increase much more drastically than for Algorithm 2. This reflects the over-approximation in the computation of the reachable sets as captured in Theorem 3. For large number of iterations, all of the above algorithms fail to compute the reach sets within a reasonable time limit, with Algorithm 3 getting stuck for smaller number of iterations as compared to Algorithm 2.

TABLE III

ALGORITHM 3 ON THE TWO SYSTEMS

Experiment 1: 2D Closed-loop System			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	6	2.6745e+03	1.3740808
2	8	3.1948e+10	1.4058749
3	10	3.1854e+14	1.6339948
4	12	3.1693e+18	3.4348258
5	14	3.1533e+22	21.029801
6	18	3.1374e+26	327.53359
Experiment 2: 4D Balance System			
Iteration	Num_Ver(X)	Volume(X)	Runtime (s)
1	2	112	2.820056915
2	16	54377	3.089679003
3	129	2.7646e+08	13.276090

VIII. CONCLUSION

We formalized the safety verification problem for closed-loop control system with anytime perception and presented a generic procedure for safety analysis. We instantiated the procedure for two classes of systems, and presented algorithms based on polyhedral manipulations. Our experiments demonstrate the feasibility of the approach. However, there are several computational challenges with respect to the scalability of the approach. In the future, we would like to experiment on real case studies as well as extend the framework to non-linear systems.

ACKNOWLEDGMENT

This work was partially supported by NSF Grant No. 2008957 and an Amazon Research Award. The authors would like to thank Shawn Keshmiri and Heechul Yun for discussions on the anytime perception framework.

REFERENCES

[1] Matthias Althoff, Colas Le Guernic, and Bruce H. Krogh. Reachable set computation for uncertain time-varying linear systems. In Marco Caccamo, Emilio Frazzoli, and Radu Grosu, editors, *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, pages 93–102. ACM, 2011.

[2] Alexander Amini, Guy Rosman, Sertac Karaman, and Daniela Rus. Variational end-to-end navigation and localization. In *IEEE International Conference on Robotics and Automation*, pages 8958–8964. IEEE, 2019.

[3] Eugene Asarin, Thao Dang, and Antoine Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43(7):451–476, 2007.

[4] Eugene Asarin, Thao Dang, Oded Maler, and Olivier Bournez. Approximate reachability analysis of piecewise-linear dynamical systems. In *Hybrid Systems: Computation and Control (HSCC)*, volume 1790 of *Lecture Notes in Computer Science*, pages 20–31. Springer, 2000.

[5] Karl Johan Astrom and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, USA, 2008.

[6] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. *arXiv e-prints*, page cs/0612085, December 2006.

[7] Mark S. Boddy and Thomas L. Dean. Solving time-dependent planning problems. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence. Detroit, MI, USA, August 1989*, pages 979–984. Morgan Kaufmann, 1989.

[8] Alongkrit Chutinan and Bruce H. Krogh. Computational techniques for hybrid system verification. *IEEE Trans. Autom. Control.*, 48(1):64–75, 2003.

[9] P. Ezudheen, Zahra Rahimi Afzal, Pavithra Prabhakar, Deepak D'Souza, and Meenakshi D'Souza. Verifying band convergence for sampled control systems. In *NASA Formal Methods - 12th International Symposium, NFM 2020, Moffett Field, CA, USA, May 11-15, 2020, Proceedings*, volume 12229 of *Lecture Notes in Computer Science*, pages 329–349. Springer, 2020.

[10] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. PAMPC: perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, pages 1–8. IEEE, 2018.

[11] Daniele Fontanelli, Luca Greco, and Antonio Bicchi. Anytime control algorithms for embedded real-time systems. In *Hybrid Systems: Computation and Control (HSCC)*, volume 4981 of *Lecture Notes in Computer Science*, pages 158–171. Springer, 2008.

[12] Mohammed Foughali and Alexander Zuepke. Formal verification of real-time autonomous robots: An interdisciplinary approach. *Frontiers Robotics AI*, 9:791757, 2022.

[13] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 379–395. Springer, 2011.

[14] Bineet Ghosh and Parasara Sridhar Duggirala. Robust reachable set: Accounting for uncertainties in linear dynamical systems. *ACM Trans. Embed. Comput. Syst.*, 18(5s):97:1–97:22, 2019.

[15] Antoine Girard and Colas Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In Magnus Egerstedt and Bud Mishra, editors, *Hybrid Systems: Computation and Control (HSCC)*, volume 4981 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2008.

[16] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *Hybrid Systems: Computation and Control (HSCC)*, volume 3927 of *Lecture Notes in Computer Science*, pages 257–271. Springer, 2006.

[17] Colas Le Guernic and Antoine Girard. Reachability analysis of hybrid systems using support functions. In *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 540–554. Springer, 2009.

[18] Sumit Kumar Jha, Bruce H. Krogh, James E. Weimer, and Edmund M. Clarke. Reachability for linear hybrid automata using iterative relaxation abstraction. In Alberto Bemporad, Antonio Bicchi, and Giorgio C. Buttazzo, editors, *Hybrid Systems: Computation and Control (HSCC)*, volume 4416 of *Lecture Notes in Computer Science*, pages 287–300. Springer, 2007.

[19] Ratan Lal and Pavithra Prabhakar. Bounded error flowpipe computation of parameterized linear systems. In *2015 International Conference on Embedded Software, EMSOFT 2015, Amsterdam, Netherlands, October 4-9, 2015*, pages 237–246. IEEE, 2015.

[20] Maxim Likhachev, Dave Ferguson, Geoffrey J. Gordon, Anthony Stentz, and Sebastian Thrun. Anytime search in dynamic graphs. *Artif. Intell.*, 172(14):1613–1643, 2008.

[21] Matt Luckcuck, Marie Farrell, Louise A. Dennis, Clare Dixon, and Michael Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Comput. Surv.*, 52(5):100:1–100:41, 2019.

[22] Ertai Luo, Niklas Kochdumper, and Stanley Bak. Reachability analysis for linear systems with uncertain parameters using polynomial zonotopes. In *Hybrid Systems: Computation and Control (HSCC)*, pages 17:1–17:12. ACM, 2023.

[23] Tobias Nägeli, Javier Alonso-Mora, Alexander Domahidi, Daniela Rus, and Otmar Hilliges. Real-time motion planning for aerial videography with real-time with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics Autom. Lett.*, 2(3):1696–1703, 2017.

[24] Venkatraman Narayanan, Mike Phillips, and Maxim Likhachev. Anytime safe interval path planning for dynamic environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 4708–4715. IEEE, 2012.

[25] Yash Vardhan Pant, Houssam Abbas, Kartik Mohta, Rhudii A. Quaye, Truong X. Nghiem, Joseph Devietti, and Rahul Mangharam. Anytime computation and control for autonomous systems. *IEEE Trans. Control. Syst. Technol.*, 29(2):768–779, 2021.

[26] Pavithra Prabhakar and Mahesh Viswanathan. A dynamic algorithm for approximate flow computations. In *Hybrid Systems: Computation and Control (HSCC)*, pages 133–142. ACM, 2011.

[27] Sebastian Preuße, Hans-Christian Lapp, and Hans-Michael Hanisch. Closed-loop system modeling, validation, and verification. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation, ETFA 2012, Krakow, Poland, September 17-21, 2012*, pages 1–8. IEEE, 2012.

[28] Daniel E. Quevedo and Vijay Gupta. Sequence-based anytime control. *IEEE Trans. Autom. Control.*, 58(2):377–390, 2013.

[29] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.2.1)*, YYYY. `https://www.sagemath.org`.

[30] Mark Wetzlinger, Niklas Kochdumper, Stanley Bak, and Matthias Althoff. Fully-automated verification of linear systems using reachability analysis with support functions. In *Hybrid Systems: Computation and Control (HSCC)*, pages 5:1–5:12. ACM, 2023.

[31] Aditya Zutshi, Sriram Sankaranarayanan, and Ashish Tiwari. Timed relational abstractions for sampled data control systems. In P. Madhusudan and Sanjit A. Seshia, editors, *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*, volume 7358 of *Lecture Notes in Computer Science*, pages 343–361. Springer, 2012.