2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Safe Control Algorithms Using Energy Functions: A Unified Framework, Benchmark, and New Directions

Tianhao Wei and Changliu Liu

***Abstract*— Safe autonomy is important in many application domains, especially for applications involving interactions with humans. Existing safe control algorithms are similar to one another in the sense that: they all provide control inputs to maintain a low value of an energy function that measures safety. In different methods, the energy function is called a potential function, a safety index, or a barrier function. The connections and relative advantages among these methods remain unclear. This paper introduces a unified framework to derive safe control laws using energy functions. We demonstrate how to integrate existing controllers based on potential field method, safe set algorithm, barrier function method, and sliding mode algorithm into this unified framework. In addition to theoretical comparison, this paper also introduces a benchmark which implements and compares existing methods on a variety of problems with different system dynamics and interaction modes. Based on the comparison results, a new method, called the sublevel safe set algorithm, is derived under the unified framework by optimizing the hyperparameters. The proposed algorithm achieves the best performance in terms of safety and efficiency on the vast majority of benchmark tests.**

I. Introduction

Safe autonomy has become increasingly critical in many application domains. We should ensure not only the safety of the ego robot, but also the safety of other agents (humans or robots) that directly interact with the autonomy. For example, robots should be safe to human workers in human-robot collaborative assembly; autonomous vehicles should be safe to other road participants. For complex autonomous systems with many degrees of freedom, safe operation depends on the correct functioning of all system components, *i.e.*, accurate perception, optimal decision making, and safe control. This paper focuses on safe control which is the last defense to ensure the safety of a system.

A safe control law needs to guarantee that the unsafe region of the system's state space is not *reachable*. Additionally, it requires forward *invariance* of the safe region in the sense that once entered, the state of the system will stay in the safe part of the state space. To achieve forward invariance or set-invariant control [1], a scalar energy function is usually designed such that the control objective (*e.g.*, safety) is with low energy. Then the desired control law should drive the energy function in the negative direction whenever the system state is outside of the desired set (*e.g.*, the safe region). An energy function has many other variations, *e.g.*, a potential function, a barrier function, or a safety index. Representative methods include potential field method (PFM) [2], sliding mode algorithm (SMA) [3], barrier function method (BFM) [4], and safe set algorithm (SSA) [5]. Though the aforementioned methods all have similar structures in the sense that they provide control inputs to decrease the value of the energy function, the connections and relative advantages among them remain unclear. This paper introduces a unified framework for safe controllers to demonstrate how to interpret different safe algorithms as variants of a common energy-function-based control.

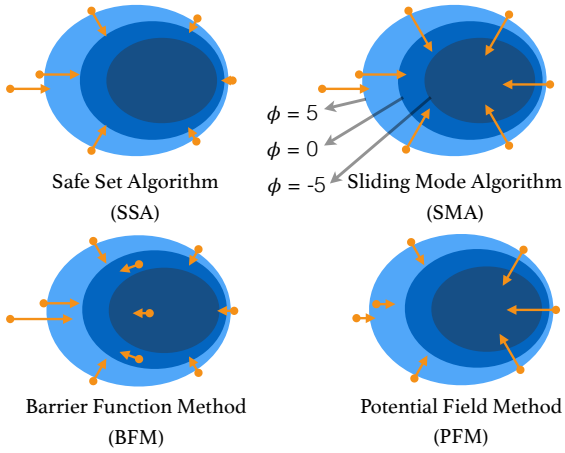

Fig. 1: Illustration of different safe control algorithms on phase graphs. The plane represents the state space. The contours represent level sets of the energy function ϕ. The system is safe when $\phi \leq 0$. An arrow indicates the direction and magnitude of the safe control input at a given state (dot).

Moreover, verification and validation of safe control algorithms are important. Lyapunov analysis [6] is usually adopted to prove invariance of the safe region using safe control laws. In addition to safety performance, we are also interested in understanding how conservative a control law is, and how much optimality or efficiency is sacrificed for the sake of safety. In general, it is difficult to mathematically analyze the trade-off between safety and efficiency in complex tasks. Empirical studies are needed to compare the performance of different algorithms in diverse complex situations. This paper introduces a benchmark system to test safe control algorithms on different dynamic systems (ball, unicycle, SCARA, 4 DoF robot arm) and different interaction modes (passive human model, interactive human model). We focus on three major metrics: safety, efficiency, and a hybrid score that incorporates both safety and efficiency. These metrics reflect the three most concerned aspects: human safety,

This work was supported in part by the National Science Foundation under Grant #1734019, and in part by Holomatic.

T. Wei and C. Liu are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA (e-mail: twei2, cliu6@andrew.cmu.edu).

robot efficiency, and robot safety. To the best knowledge of the authors, this is the first comprehensive benchmark on safe control. It can be used to evaluate not only energy-function-based safe control algorithms, but also controllers based on non-analytical methods such as those based on reinforcement learning [7] and imitation learning [8].

Based on theoretical analysis and comparison results of existing algorithms, it is observed that SSA and BFM have the best performance. Both SSA and BFM have relative advantages over the other in certain circumstances. SSA is triggered less frequently but reacts more radically, hence good for scenarios that are less safe. BFM reacts more gently but is triggered more often, hence good for scenarios that are safer. Based on the observation, we propose a new method, the sublevel safe set algorithm (SSS), combining the strengths of SSA and BFM. This method achieves the best performance on the vast majority of our benchmark tests.

The contributions of the paper are listed below.

1) This paper introduces a unified framework to derive safe control algorithms and shows that existing methods fit into the framework.[1]
2) This paper develops a benchmark for existing safe control methods on a variety of problems with different system dynamic and different interaction modes.[2]
3) This paper proposes a new safe control algorithm SSS that outperforms exiting algorithms on most benchmark tests.

II. Problem and Framework

A. Notation and Problem Definition

Consider the following affine dynamical system with n_x states and n_u inputs

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}, \tag{1}$$

where $\mathbf{x} \in X \subset \mathbb{R}^{n_x}$ is the state vector defined in configuration space, $\mathbf{u} \in U \subset \mathbb{R}^{n_u}$ is the control input vector assumed to be unconstrained, $\mathbf{f} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $\mathbf{g} : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$ defines the system dynamics.

The objective of the system can either be to track a trajectory or to regulate around a settle point. A reference control $\mathbf{u}_0$ is provided to fulfill the system objective. In a safe environment, the robot can just execute $\mathbf{u}_0$. Otherwise, the reference control $\mathbf{u}_0$ may need to be modified by a safe control algorithm to prevent collisions with obstacles. The resulting safe control input $\mathbf{u}$ depends on $\mathbf{u}_0$.

The robot is occupying a certain region of the Cartesian space denoted as $C_r \subset \mathbb{R}^3$. Similarly, the space occupied by the obstacle is denoted $C_o \subset \mathbb{R}^3$. We denote $\mathbf{c}_r$ as the closest point on the robot to the obstacle, $\mathbf{c}_o$ as the closest point on the obstacle to the robot. Mathematically,

$$\mathbf{c}_r, \mathbf{c}_o = \underset{\mathbf{c}_r^* \in C_r, \mathbf{c}_o^* \in C_o}{\arg\min} \|\mathbf{c}_r^* - \mathbf{c}_o^*\|_2. \tag{2}$$

[1]An extended version with proof and additional information can be found at https://arxiv.org/abs/1908.01883.

[2]The benchmark is available at https://github.com/intelligent-control-lab/BIS.

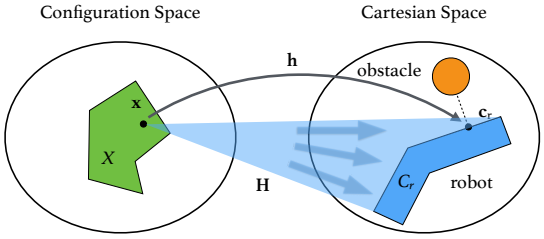

Fig. 2: Illustration of the configuration space X, the state vector $\mathbf{x}$, the occupied area C_r in the Cartesian space, and the closest point $\mathbf{c}_r$ on the robot to the obstacle.

Let $\mathbf{H}\colon \mathbf{x} \mapsto C_r$ be a mapping from the robot state $\mathbf{x}$ to its occupied region C_r. Let $\mathbf{h}\colon \mathbf{x} \mapsto \mathbf{c}_r$ be a mapping from the robot state $\mathbf{x}$ to the closest point $\mathbf{c}_r$. The mappings depend on the robot model. The notations are shown in Fig. 2.

The relation between time derivative of $\mathbf{c}_r$ and time derivative of $\mathbf{x}$ is

$$\dot{\mathbf{c}_r} = \mathbf{h}'(\mathbf{x})\ \dot{\mathbf{x}} = \mathbf{J}_{c_r}\ \dot{\mathbf{x}}, \tag{3}$$

where $\mathbf{J}_{c_r} = \mathbf{h}'(\mathbf{x})$ is the Jacobian matrix.

The relative distance between the obstacle and the robot is denoted by $d := \|\mathbf{c}_r - \mathbf{c}_o\|_2$. The relative velocity is denoted by $\dot{d} = \frac{\mathrm{d}}{\mathrm{d}t}\|\mathbf{c}_r - \mathbf{c}_o\|_2$. We define d_{min} as the minimum required safe distance.

B. Energy Function and Control

Energy-function-based methods use a customized energy function to measure safety. The lower, the safer. The value of ϕ, which is usually called a safety index, increases when the robot is going toward the obstacle. The goal of the algorithm is to provide a control input that draws the robot away from the obstacle by decreasing the safety index.

We denote $\phi : X \to \mathbb{R}$ as an energy function defined on the configuration space and $\tilde{\phi}(\mathbf{c}_r) : C \to \mathbb{R}$ as an equivalent energy function on the Cartesian space, where

$$\phi(\mathbf{x}) = \tilde{\phi}(\mathbf{c}_r) = \tilde{\phi}(\mathbf{h}(\mathbf{x})). \tag{4}$$

The function ϕ should be designed such that the system is safe if $\phi(\mathbf{x}) \leq 0$. By the chain rule,

$$\nabla\phi(\mathbf{x}) = \frac{\partial\phi(\mathbf{x})}{\partial\mathbf{x}} = \mathbf{J}_{\mathbf{c}_r}^{\mathsf{T}}\ \frac{\partial\tilde{\phi}(\mathbf{c}_r)}{\partial\mathbf{c}_r} = \mathbf{J}_{\mathbf{c}_r}^{\mathsf{T}}\ \nabla\tilde{\phi}(\mathbf{c}_r). \tag{5}$$

For safety, $\phi(\mathbf{x})$ should be maintained negative. Once $\phi(\mathbf{x})$ is high, *i.e.*, in danger, it should be made decreasing, *i.e.*, its time derivative $\dot{\phi}(\mathbf{x})$ should be less than 0. Even in the safe situations, $\phi(\mathbf{x})$ should not increase too fast, *i.e.*, its time derivative may be upper bounded. Hence, we have an inequality constraint on $\dot{\phi}(\mathbf{x})$ where

$$\begin{aligned}
\dot{\phi}(\mathbf{x}) &= \nabla\phi^{\mathsf{T}}(\mathbf{x})\ \dot{\mathbf{x}} && \text{(6a)}\\
&= \nabla\phi^{\mathsf{T}}(\mathbf{x})\ (\mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\ \mathbf{u}) && \text{(6b)}\\
&= \underbrace{\nabla\phi^{\mathsf{T}}(\mathbf{x})\ \mathbf{f}(\mathbf{x})}_{\mathrm{L}_{\mathbf{f}}\phi} + \underbrace{\nabla\phi^{\mathsf{T}}(\mathbf{x})\ \mathbf{g}(\mathbf{x})}_{\mathbf{L}_{\mathbf{g}}\phi}\mathbf{u} \leq \xi. && \text{(6c)}
\end{aligned}$$

The slack term $\xi \in \mathbb{R}$ is tunable. If $\xi > 0$, $\phi(\mathbf{x})$ is allowed to increase within a certain rate. If $\xi < 0$, $\phi(\mathbf{x})$ must decrease. The inertia term $\mathrm{L}_{\mathbf{f}}\phi \in \mathbb{R}$ represents how the current state $\mathbf{x}$

affects $\dot{\phi}(\mathbf{x})$. The Lie derivative $\mathbf{L_g}\phi \in \mathbb{R}^{n_u}$ represents how the control input $\mathbf{u}$ affects $\dot{\phi}(\mathbf{x})$.

C. Safe Control Algorithms

Safe control methods that use energy-function-based approaches are reviewed below, in particular, the four methods shown in Fig. 1. These methods have different definitions of the energy functions. Since our framework focuses on the control strategies and can be generalized to any kind of energy function, we only review their control strategies below. Moreover, to focus on the main idea and reduce the number of hyperparameters in the unified framework, these algorithms are presented in their simplest forms. In section II-D, we show how these methods are related.

1) PFM: Instead of deriving a control input $\mathbf{u}$ in configuration space directly, PFM derives a control input $\mathbf{u}_c$ in the Cartesian space first considering the following dynamics:

$$\dot{\mathbf{c}}_r = \mathbf{u}_c := \mathbf{u}_c^0 + \mathbf{u}_c^*, \tag{7}$$

where $\mathbf{u}_c^0$ is the reference control in the Cartesian space transformed from $\mathbf{u}_0$ based on (1) and (3) such that

$$\mathbf{u}_c^0 = \mathbf{J}_{\mathbf{c}_r}\ \mathbf{g}\ \mathbf{u}_0, \tag{8}$$

and $\mathbf{u}_c^*$ is a repulsive "force" added to the reference $\mathbf{u}_c^0$ whenever the safety constraint is violated. In particular,

$$\mathbf{u}_c = \begin{cases} \mathbf{u}_c^0 - c_1\ \nabla\tilde{\phi} & \text{if } \tilde{\phi} \geq 0 \\ \mathbf{u}_c^0 & \text{otherwise} \end{cases}, \tag{9}$$

where $c_1 > 0$ is a tunable constant. Then the equivalent control input $\mathbf{u}$ in the configuration space can be derived from $\mathbf{u}_c$.

2) SMA: It adds a correction term to the reference $\mathbf{u}_0$ along the direction of the Lie derivative whenever the safety constraint is violated.

$$\mathbf{u} = \begin{cases} \mathbf{u}_0 - c_2\ \mathbf{L_g}\phi^{\mathsf{T}} & \text{if } \phi \geq 0 \\ \mathbf{u}_0 & \text{otherwise} \end{cases}, \tag{10}$$

where the constant $c_2 > 0$ should be set large enough such that $\dot{\phi} = \mathrm{L_f}\phi - c_2\ \|\mathbf{L_g}\phi\|^2 + \mathbf{L_g}\phi\ \mathbf{u}_0$ is always negative.

3) SSA: It computes a control input that is closest to the reference $\mathbf{u}_0$ and decreases ϕ when $\phi > 0$.

$$\mathbf{u} = \min_{\mathbf{u}} ||\mathbf{u}_0 - \mathbf{u}||_2,\ \text{s.t. } \dot{\phi} \leq \eta \text{ or } \phi < 0, \tag{11}$$

where $\eta < 0$ corresponds to the slack term in (6). SSA only deviates from the reference $\mathbf{u}_0$ when $\phi \geq 0$.

4) BFM: It computes a control input that is closest to the reference $\mathbf{u}_0$ and satisfies $\dot{\phi} < \lambda\ \phi$ for a constant number $\lambda < 0$.

$$\mathbf{u} = \min_{\mathbf{u}} \|\mathbf{u}_0 - \mathbf{u}\|_2,\ \text{s.t. } \dot{\phi} \leq \lambda\ \phi, \tag{12}$$

where $\lambda\phi$ corresponds to ξ in (6). BFM may always deviate from the reference $\mathbf{u}_0$. When safe, *i.e.*, $\phi < 0$, the control input may lead to the increase of the safety index ϕ.

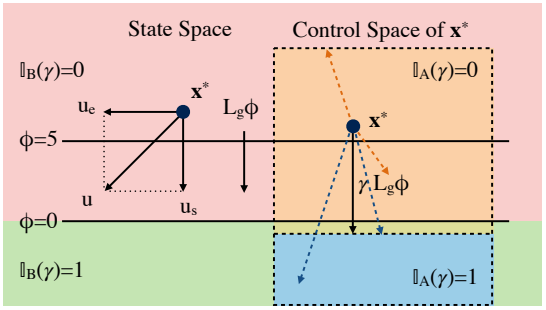

Fig. 3: Illustration of the Lie derivative $\mathbf{L_g}\phi$, perpendicular decomposition of $\mathbf{u}$, and the indicator functions $\mathbb{I}_A(\gamma)$ and $\mathbb{I}_B(\phi)$. Left: Decomposition of $\mathbf{u}$ along and perpendicular to $\mathbf{L_g}\phi$. Right: Constraint on the control space. Blue arrows are examples of $\mathbf{u}_0$ that comply with the constraint; Orange arrows are examples of $\mathbf{u}_0$ that violate the constraint. It is assumed that $\dot{\mathbf{x}} = \mathbf{u}$ in the figure.

D. Unified Framework

Before proposing the unified framework, we first introduce a perpendicular decomposition of the control input $\mathbf{u}$,

$$\mathbf{u} = \mathbf{u}^s + \mathbf{u}^e, \tag{13}$$

where $\mathbf{u}^s$ is parallel to $\mathbf{L_g}\phi$, and $\mathbf{u}^e$ is orthogonal to $\mathbf{L_g}\phi$ as shown in Fig. 3. We call $\mathbf{u}^s$ the *safety component*, since the change of ϕ depends solely on $\mathbf{u}^s$. We call $\mathbf{u}^e$ the *efficiency component*. When safety is ensured, we can add control input orthogonal to $\mathbf{L_g}\phi$ to improve the efficiency of system performance.

Similarly, the reference control input $\mathbf{u}_0$ can be decomposed as $\mathbf{u}_0 = \mathbf{u}_0^s + \mathbf{u}_0^e$ where

$$\mathbf{u}_0^s = \mu\ \mathbf{L_g}\phi^{\mathsf{T}}, \mathbf{u}_0^e = \mathbf{u}_0 - \mathbf{u}_0^s, \tag{14}$$

where

$$\mu := \frac{\mathbf{L_g}\phi\ \mathbf{u}_0}{\|\mathbf{L_g}\phi\|^2}. \tag{15}$$

For convenience, we introduce two indicator functions. Function $\mathbb{I}_B(\phi)$ indicates whether a state is dangerous, *i.e.*,

$$\mathbb{I}_B(\phi) := \begin{cases} 1, & \text{if } \phi \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Function $\mathbb{I}_A(\gamma)$ is for optimization-based methods, *e.g.*, SSA and BFM. It indicates whether $\mathbf{u}_0$ violates the optimization constraint $\dot{\phi} \leq \xi$. According to (6), the optimization constraint defines a half space in the control space as shown in Fig. 3, whose normal direction is along $\mathbf{L_g}\phi$. Define

$$\gamma := \frac{\xi - \mathrm{L_f}\phi}{\|\mathbf{L_g}\phi\|^2}. \tag{17}$$

Then an input $\mathbf{u}$ satisfies the optimization constraint (6c) if and only if $\mathbf{L_g}\phi\ \mathbf{u} \leq \gamma\|\mathbf{L_g}\phi\|^2$. Hence, $\mathbf{u}_0$ is feasible with respect to the constraint if and only if $\mathbf{L_g}\phi\ \mathbf{u}_0 \leq \gamma\|\mathbf{L_g}\phi\|^2$ or equivalently $\mu \leq \gamma$. Then we define

$$\mathbb{I}_A(\gamma) := \begin{cases} 1, & \text{if } \mu > \gamma \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Definition 1 (Energy-Function-Based Safe *control method is called an energy-functio trol method if:*

1) *it uses a scalar energy function ϕ to*
2) *it provides safe control inputs in the*

$$\begin{cases} \mathbf{u}^s = \alpha\ \mathbf{L_g}\phi^{\mathsf{T}} \\ \mathbf{u}^e = \mathbf{u}_0^e + \beta\ \mathbf{u}_i^e \end{cases}$$

where $\alpha \in \mathbb{R}$ is a tunable para response, $\beta \in \mathbb{R}$ is a tunable parame level, and $\mathbf{u}_i^e \in \mathbb{R}^{n_u}$ is some vector or

Theorem 1. *PFM in* (9)*, SMA in* (10)*, SSA in* (12) *are all energy-function-based safe They all satisfy* (19) *with difference choices of parameters. In all methods, $\beta = 0$.*

1) *for PFM:*

$$\alpha = \mu - \mathbb{I}_B(\phi)\ c_1. \tag{20}$$

2) *for SMA:*

$$\alpha = \mu - \mathbb{I}_B(\phi)\ c_2. \tag{21}$$

3) *for SSA:*

$$\alpha = (1 - \mathbb{I}_A(\gamma)\ \mathbb{I}_B(\phi))\ \mu + \mathbb{I}_A(\gamma)\ \mathbb{I}_B(\phi)\ \gamma, \tag{22}$$

where γ follows from (17) *and $\xi = \eta$.*

4) *for BFM:*

$$\alpha = (1 - \mathbb{I}_A(\gamma))\ \mu + \mathbb{I}_A(\gamma)\ \gamma \tag{23}$$

where γ follows from (17) *and $\xi = \lambda\phi$.*

E. Sublevel Safe Set Algorithm

Looking into the control strategies for different methods, we notice that

1) BFM's slack term in (12) is a dynamic term that is related to energy function value, while SSA's slack term in (11) is not.
2) SSA may only provide control correction when $\phi > 0$. The corresponding hyperparameter is (16). Yet BFM may provide control correction regardless of the value of ϕ.

When the parameters (*e.g.*, the energy function ϕ) are designed less conservative, SSA only starts to provide control correction at a close distance, while BFM may deviates from the optimal reference all the time. In this situation, SSA will be more efficient. When the parameters are designed more conservative, though BFM's control correction goes into effect from a far distance, BFM only makes a minor correction for most of the time. Meanwhile, SSA still provides radical corrections. In this way, BFM will be more efficient. These analyses will be supported by experimental results in section IV.

We propose a new method, Sublevel Safe Set (SSS), to combine the strengths of SSA and BFM. SSA only provides control correction when $\phi \geq 0$, while the control correction relies on the value of the energy function ϕ. The phase graph for SSS is shown in Fig. 4. Thus, no matter how

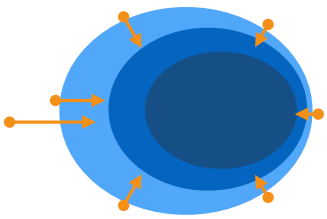

Fig. 4: Phase graph for SSS.

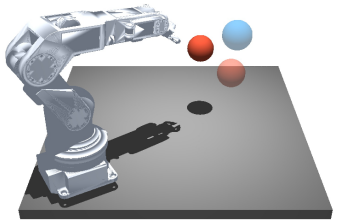

Fig. 5: Screenshot of BIS.

the parameters are designed, SSS should be more efficient than both SSA and BFM, which will also be verified in the experiment results in section IV.

By combining the slack term ξ in BFM and the parameter α (22) in SSA, hyperparameters for SSS are designed to be

$$\begin{cases} \alpha = (1 - \mathbb{I}_A(\gamma)\ \mathbb{I}_B(\phi))\ \mu + \mathbb{I}_A(\gamma)\ \mathbb{I}_B(\phi)\ \gamma \\ \beta = 0 \\ \gamma = \frac{\xi - \mathrm{L_f}\phi}{\|\mathbf{L_g}\phi\|^2} \\ \xi = \lambda\phi \end{cases}. \tag{24}$$

The corresponding control strategy in its optimization form is

$$\mathbf{u} = \min_{\mathbf{u}} ||\mathbf{u}_0 - \mathbf{u}||,\ \text{s.t. } \dot{\phi} < \lambda\ \phi \text{ or } \phi < 0. \tag{25}$$

III. Benchmark of Interactive Safety

A. Overview of the Benchmark

The unified framework provides a powerful tool to analyze connections and differences among algorithms. However, it is insufficient to solely rely on mathematical analysis to derive the performance of different algorithms in stochastic environments. In this paper, we are interested in the trade-off between safety and efficiency when these algorithms are applied on interactive tasks. Empirical studies are needed to compare different algorithms and understand their relative advantages in diverse complex situations.

We introduce the benchmark of interactive safety (BIS) to perform empirical studies on safe control methods. BIS consists of a collection of robot models which can be used to benchmark and compare the performance of different safe control algorithms. A set of ready-made robot models can be easily extended by users. By changing robot models, we can test an algorithm with different system dynamics. By replacing the robot controller, we can test different control algorithms on a same dynamic system. By instantiating two robot models and letting a human subject (or a human-like controller) control one of them, we can test human-robot interactions. This paper is focused on the two agent case with one human and one robot, while multi-agent cases will be studied in the future.

In the simulation, the human agent is represented as an orange ball. The task of both the human and the robot is to reach a series of goals, which is represented by transparent balls. Screenshot of the scenario is shown in Fig. 5. BIS includes a data generator to generate test scenarios and an evaluator which evaluates all algorithms under the same condition.

B. Robot Models under Comparison

All robot models contains two modules: control module and execution module. In the control module, the robot first uses a Kalman filter to update the state of itself and the environment based on the measured data. Then the robot performs a collision check and computes the minimum distance and the closest point to the obstacle. Finally, the robot calls a control algorithm to compute the desired control input. In the execution module, the control input is applied to the robot simulator. The configuration of the robot is updated according to the dynamic model specified in the robot simulator. The robot simulator can also be replaced by a robot hardware to achieve hardware-in-the-loop evaluation.

BIS currently include four different robot dynamic models: ball robot model, unicycle robot model, SCARA robot model, and 4 DoF robot arm robot model.

The robot model library in BIS can be easily extended. Adding a robot model into the robot model library requires two functions $\mathbf{h}$ and $\mathbf{h}'$. Function $\mathbf{h}$ maps the robot state $\mathbf{x}$ into Cartesian critical point $\mathbf{c}_r$ as shown in Fig. 2. Function $\mathbf{h}'$ is the Jacobian in (3).

C. Controllers under Comparison

Controllers are called by robot models in the control module. BIS has included five energy-function-based control methods, *i.e.*, PFM, SMA, BFM, SSA, SSS, and a human-like controller. The human-like controller models human behavior and generates control input that is similar to human. An imitation learning algorithm is designed to learn human behavior models from real human subjects. We asked 3 human subjects to control the human agent to achieve 100 goals one by one. The human model is learned from the demonstration data.

D. Experiment Methods

The experiments are setup with the following steps:

1) Use a data generator to generate random goals and save them as test scenarios.
2) Use the same test scenario and human model to test each algorithm.
3) Compare the test results.

We use 40 pieces of 30 seconds long test scenarios to test different algorithms in the experiments. The frame rate is 20 fps. In the study, it is assumed that 1) the Jacobian matrix does not change within one frame; and 2) all noises follow normal distribution.

The function ϕ is chosen to be [5],

$$\phi = d_{min}^2 - d^2 - k\dot{d}. \tag{26}$$

In the experiments, we test the performance of the algorithms under different values of their parameters. In particular, we tune the following two sets of parameters: 1) the parameters associated with ϕ, *i.e.*, d_{min} and k; and 2) the parameters specific to each algorithms, *i.e.*, c_1 for PFM, c_2 for SMA, η for SSA, λ for BFM, λ for SSS.

E. Evaluation Metrics

Three metrics are used to evaluate the performance of different algorithms: an efficiency score and a safety score for human-robot interactions, and a hybrid score for robot co-working. For all scores, the higher, the better.

1) Efficiency Score: We use the average number of goals achieved in a given period by the robot as the efficiency score.

2) Safety Score: Intuitively, this score is similar to the negation of a safety index. We design the safety score to be a weight sum of the relative velocity, where the weights are decided by the relative distance. The following factors need to be considered.

- The score should decrease when the distance between the robot and the obstacle decreases.
- The score should decrease when the robot is moving faster to the obstacle.
- The weight should change rapidly when the robot is close to the obstacle.
- The score should not accumulate when the relative distance is larger than a threshold d_s.

Based on these considerations, the safety score is defined as

$$\text{safety} = -\sum_{0}^{T} \min(0, \log(d/d_s))\ \dot{d}.$$

The safety score takes both physical safety and psychological safety into consideration. When the robot is near the obstacle and rushing toward it, even if it does not end up with a collision, it receives a penalty because the robot is threatening the obstacle.

3) Hybrid Score: In robot-robot collaboration scenario, physical safety is the only concern. Only collision matters, while psychological safety should be ignored. We define the hybrid score as the maximum efficiency without collision to evaluate the performance of the algorithms in these situations. This score reflects an algorithm's best performance when it can ensure physical safety.

IV. Comparison Results

A. Trade-off between Safety and Efficiency

We first evaluate the trade-off between the efficiency score and the safety score for all algorithms as shown in Fig. 6. The trade-off curve can be obtained by tuning the parameters in the algorithms. For example, in PFM in (9), when the magnitude of c_1 is smaller, the system can be less safe but more efficient due to fewer detours. Different parameters may result in different safety and efficiency scores, which corresponds to different points on the trade-off graph in Fig. 6. If a set of parameters leads to a high safety score, we say the parameters are conservative. The up-right convex hull of the those points is the trade-curve for one algorithm. If an algorithm has a trade-off curve covers all other algorithms, *i.e.*, higher efficiency for the same safety score, we say it outperforms other algorithms.

SSA and SMA both provide control correction only in the boundary of the safe region of states. However, based

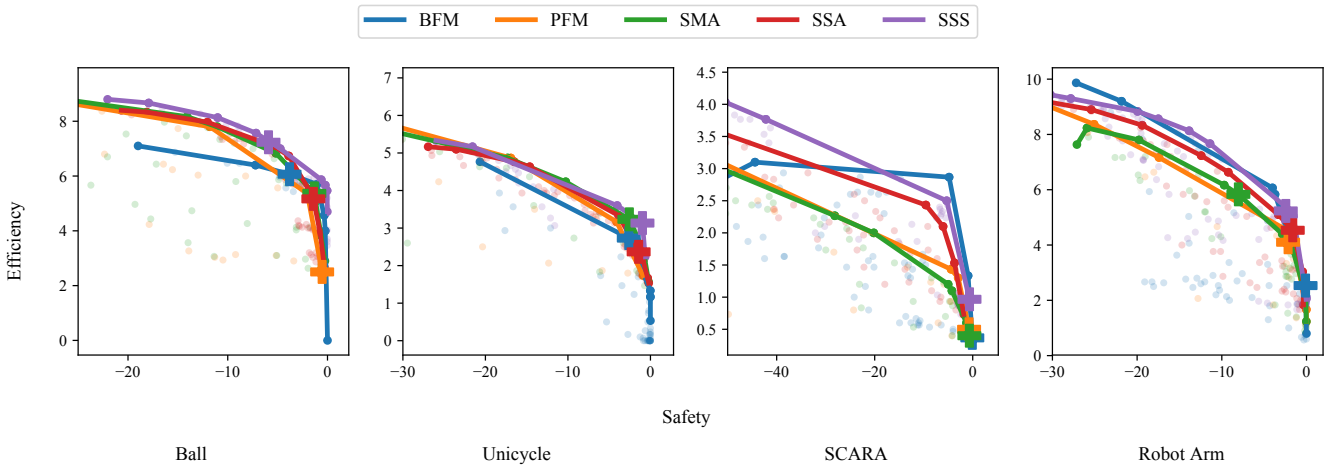

Fig. 6: The trade-off curves between safety and efficiency for four robot models.

on (21)and (22), SSA provides a smoother control input comparing to SMA, which makes it more efficient.

BFM provides control corrections even inside the safe region. This behavior makes it less efficient when the parameters are conservative. However, we noticed that BFM has a better performance than most algorithms with conservative parameters for the following two reasons.

1) When the parameters are conservative, control correction is triggered a lot. Frequent correction eliminates the advantage of only correcting at the boundary of the safe region, *i.e.*, SSA and SMA are not superior at efficiency in this situation.
2) BFM's control correction is a dynamic term that is related to energy function value, while SSA is not. Though control correction is triggered more often, BFM only makes a minor correction for most of the time. However, SSA treats all corrections equally. In other words, SSA is more likely to be overreacting.

Our new algorithm SSS overcomes the drawbacks of SSA and BFM. Thus, it achieves the best performance on the vast majority of benchmark tests.

B. Hybrid Score

To demonstrate the performance of algorithms in a robot-robot collaboration scenario, we record the hybrid scores on different robot models as shown in table I. SSS has the best average performance, which achieves two best scores and two second scores. SMA achieves two best scores, which is out of our expectation.

	Ball	Unicycle	SCARA	RobotArm
SSS	**7.23**	<u>3.13</u>	**0.96**	<u>5.23</u>
BFM	<u>6.07</u>	2.73	0.37	2.53
SSA	5.17	2.37	null	4.53
SMA	5.37	**3.23**	<u>0.39</u>	**5.83**
PFM	2.50	null	0.03	4.10

TABLE I: Comparison of hybrid scores (Maximum Efficiency without Collision). The best two results are shown in **bold** and <u>underline</u> respectively. If collision happens, the method gets a null as the hybrid score.

V. Conclusion

This paper introduced a unified framework to derive safe control laws using energy functions. We proved that a variety of controllers can be derived from the unified framework by applying different hyperparameters. A benchmark system was introduced to evaluate the performance of different algorithms on a variety of scenarios with different system dynamics. The unified framework and the benchmark system helped us understand how hyperparameters of an algorithm affects the performance in terms of safety and efficiency. Based on the unified framework and comparison results, we proposed a new method, sublevel safe set algorithm (SSS). This new method combined the strengths of two best-performed algorithms: the safe set algorithm (SSA) and the barrier function algorithm (BFM), which outperform existing methods on most benchmark tests.

References

[1] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.
[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.
[3] L. Gracia, F. Garelli, and A. Sala, "Reactive sliding-mode algorithm for collision avoidance in robotic systems," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2391–2399, 2013.
[4] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.
[5] C. Liu and M. Tomizuka, "Control in a safe set: Addressing safety in human-robot interactions," in *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2014, pp. V003T42A003–V003T42A003.
[6] M. S. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on automatic control*, vol. 43, no. 4, pp. 475–482, 1998.
[7] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Advances in neural information processing systems*, 2017, pp. 908–918.
[8] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A fast integrated planning and control framework for autonomous driving via imitation learning," in *ASME 2018 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, 2018, pp. V003T37A012–V003T37A012.