2022 IEEE 61st Conference on Decision and Control (CDC)
December 6-9, 2022. Cancún, Mexico

dSLAP: Distributed Safe Learning and Planning for Multi-robot Systems

Zhenyuan Yuan[1] Minghui Zhu[1]

2022 IEEE 61st Conference on Decision and Control (CDC) | 978-1-6654-6761-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CDC51059.2022.9992938

***Abstract*— This paper considers the problem where a group of mobile robots subject to unknown external disturbances aim to safely reach goal regions. We develop a distributed safe learning and planning algorithm that allows the robots to learn about the external unknown disturbances and safely navigate through the environment via their single trajectories. We use Gaussian process regression for online learning where variance is adopted to quantify the learning uncertainty. By leveraging set-valued analysis, the developed algorithm enables fast adaptation to newly learned models while avoiding collision against the learning uncertainty. Active learning is then applied to return a control policy such that the robots are able to actively explore the unknown disturbances and reach their goal regions in time. Sufficient conditions are established to guarantee the safety of the robots. A set of simulations are conducted for evaluation.**

I. Introduction

Intelligent robots are becoming ubiquitous in our life, such as autonomous driving, precision agriculture and emergency response. Artificial intelligence is a key component to achieve the vision of long-term autonomy. The decisions of the robots may have profound effects on surrounding objects and unwanted outcomes could cause damages physically or monetarily [1]. Hence, safety becomes a vital issue that must be examined before widely deploying intelligent robots.

Safe machine learning aims to solve certain learning tasks and meanwhile ensure robots' safety. It can be categorized into offline learning and online learning. For offline learning, models are trained using a fixed set of data, and the training is usually formulated as optimization problems where objective functions and constraints reflect safety considerations. Interested readers are referred to Section 3 of paper [2].

When robots encounter changes of environments, online learning is desired to ensure mission completion. Notice that the learning errors could be large during the initial learning phase when training data is limited. It is therefore important and challenging to keep the robots safe during the entire learning process. By learning tasks, related literature can be categorized into four classes: (1) exploration [3], where the objective is to learn about the uncertainties of a dynamic model or an environment; (2) optimization [4], where decision variables are selected to optimize an unknown objective function; (3) reinforcement learning (RL) [2], where the objective is to find an optimal control policy to maximize aggregate return; (4) control [5], where the objective is to derive a control policy to achieve certain specifications, such as stabilization and goal reaching. These works usually define safety as hard constraints. Safety is achieved if the constraints are satisfied throughout the learning process.

This paper studies safe control, which aims to synthesize a control policy by online learning uncertainties and steer a system to a goal region. Paper [5] considers general nonlinear dynamics and derives the backup policy against system uncertainties by solving a two-player zero-sum differential game. Paper [6] determines the switching law based on the estimates of the region of attraction of a given backup policy for a robot with control-affine dynamics. Robust model predictive control (MPC) is applied over a linear time-invariant system in [7] to determine the switch between a learning-based controller and a backup controller. Backup policies are appended as the terminal solution to a robust MPC in [8] such that a robot can never go beyond the safe region during exploration. Paper [9] combines a model-free RL-based controller with a model-based controller utilizing control barrier functions to guarantee safety and improve system performances online. The aforementioned papers only consider single robots and static state constraints (e.g., static obstacles), whereas in multi-robot systems, from the perspective of each single robot, the state constraints are dynamic due to the motion of the other robots, analogous to moving obstacles.

Contribution statement. We consider the problem where a group of mobile robots with general nonlinear dynamics subject to unknown external disturbances aim to safely reach goal regions. We propose dSLAP, the **d**istributed **S**afe **L**earning **A**nd **P**lanning framework, which enables the robots to online learn about the disturbances, update their safe actions that avoid collisions against the learning uncertainty, and actively collect data to better reach the goals. Our contribution is summarized as follows:

- Our motion planner has two stages. First, built on set-valued analysis, a distributed safe motion planner allows for fast adaptation to the sequence of dynamic models resulted from online Gaussian process regression. The planner constructs a directed graph through connecting a robot's one-step forward sets, and then truncates the graph by removing the control inputs leading to collisions. Second, a distributed model predictive controller selects safe control inputs balancing moving towards the goals and actively learning the disturbances.
- Our two-stage motion planning is in contrast to the classic formulation [10] [11] of optimal multi-robot mo-

[1]Zhenyuan Yuan and Minghui Zhu are with School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802, USA ({zqy5086,muz16}@psu.edu). This work was partially supported by NSF grants ECCS-1710859, CNS-1830390 and ECCS-1846706 and the Penn State College of Engineering Multidisciplinary Research Seed Grant Program.

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.

tion planning, whose solutions solve collision avoidance and optimal arrival simultaneously and are known to be computationally challenging (PSPACE-hard [12]). Instead, the computational complexity of dSLAP is independent of the number of the robots.

- We derive sufficient conditions to guarantee the safety of the robots in the absence of backup policy.

Monte Carlo simulation is conducted for evaluations.

Notations. We use superscript $(\cdot)^{[i]}$ to distinguish the local values of robot i. Define the distance metric $\rho(x,x') \triangleq \|x-x'\|_\infty$, the point-to-set distance as $\rho(x,\mathcal{S}) \triangleq \inf_{x'\in\mathcal{S}} \rho(x,x')$ for a set $\mathcal{S}$, the closed ball centered at $x \in \mathbb{R}^{n_x}$ with radius r as $\mathcal{B}(x,r) \triangleq \{x' \in \mathbb{R}^{n_x} | \rho(x,x') \leqslant r\}$, and shorthand $\mathcal{B}$ the closed unit ball centered at 0 with radius 1.

Below are the implementations of common procedures. *Element removal*: Given a set $\mathcal{S}$ and an element s, procedure Remove removes element s from $\mathcal{S}$; i.e., $\mathsf{Remove}(\mathcal{S},s) \triangleq \mathcal{S}\setminus\{s\}$. *Element addition*: Given a set $\mathcal{S}$ and an element s, procedure Add appends s to $\mathcal{S}$, i.e., $\mathsf{Add}(\mathcal{S},s) \triangleq \mathcal{S}\cup\{s\}$. *Nearest neighbor*: Given a state s and a finite set $\mathcal{S}$, Nearest chooses a state in $\mathcal{S}$ that is closest to s; i.e., $\mathsf{Nearest}(s,\mathcal{S})$ picks $y \in \mathcal{S}$, where $\rho(s,y) = \rho(s,\mathcal{S})$.

II. Problem Formulation

In this section, we introduce the model of the multi-robot system, describe the formulation of the motion planning problem, and state the objective of this paper.

Mobile multi-robot system. Consider a network of robots $\mathcal{V} \triangleq \{1,\cdots,n\}$. The dynamic system of each robot i is given by the following differential equation:

$$\dot{x}^{[i]}(t) = f^{[i]}(x^{[i]}(t),u^{[i]}(t)) + g^{[i]}(x^{[i]}(t),u^{[i]}(t)), \quad (1)$$

where $x^{[i]}(t) \triangleq [x_q^{[i]}(t)^T, x_r^{[i]}(t)^T]^T \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$ is the state of robot i at time t, $x_q^{[i]} \in \mathcal{X}_q \subseteq \mathbb{R}^{n_q}$ is the location of the robot, $x_r^{[i]} \in \mathcal{X}_r \subseteq \mathbb{R}^{n_r}$, $n_r = n_x - n_q$, is the rest part of the state (e.g., heading angle and velocity), $u^{[i]}(t) \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is its control input, $f^{[i]}$ denotes the system dynamics of robot i, and $g^{[i]}$ represents the external unknown disturbance. We impose the following assumption:

Assumption II.1. **(A1)** (*Lipschitz continuity).* The system dynamics $f^{[i]}$ and the unknown disturbance $g^{[i]}$ are Lipschitz continuous.

(A2) (*Compactness).* Spaces $\mathcal{X}$ and $\mathcal{U}$ are compact. ■

Assumption **(A1)** implies that $f^{[i]} + g^{[i]}$ is Lipschitz continuous. Choose constant $\ell^{[i]}$, which is larger than the Lipschitz constant of $f^{[i]} + g^{[i]}$ and constant $m^{[i]}$, which is larger than the supremum of $f^{[i]} + g^{[i]}$ over $\mathcal{X}$ and $\mathcal{U}$.

Motion planning. We denote closed obstacle region by $\mathcal{X}_O \subseteq \mathcal{X}_q$, goal region by $\mathcal{X}_G^{[i]} \subseteq \mathcal{X}_q \setminus \mathcal{X}_O$, and free region at time t by $\mathcal{X}_F^{[i]}(x_q^{[\neg i]}(t)) \triangleq \mathcal{X}_q \setminus \big(\mathcal{X}_O \bigcup \cup_{j\neq i} \mathcal{B}(x_q^{[j]}(t), 2\zeta)\big)$, where $\neg i \triangleq \mathcal{V}\setminus\{i\}$ and $\zeta > 0$ is the size of the robots. Each robot i aims to synthesize a feedback policy $\pi^{[i]} : \mathcal{X}^n \to \mathcal{U}$ such that the solution to system (1) under $\pi^{[i]}$ satisfies $x_q^{[i]}(t_*^{[i]}) \in \mathcal{X}_G^{[i]}$, $x_q^{[i]}(\tau) \in \mathcal{X}_F^{[i]}(x_q^{[\neg i]}(\tau))$, $0 \leqslant \tau \leqslant t_*^{[i]} < \infty$, where $t_*^{[i]}$ is the first time when robot i reaches $\mathcal{X}_G^{[i]}$. That is, each robot i needs to reach the goal region within finite time and be free of collision.

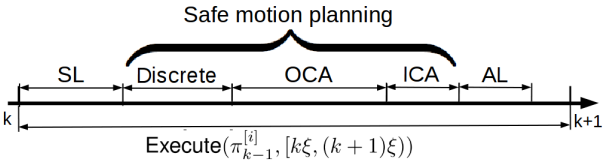


Fig. 1: Implementation of dSLAP over one iteration

Problem statement. This paper aims to solve the above multi-robot motion planning problem despite unknown function $g^{[i]}$. Since the unknown function $g^{[i]}$ is learned online, each robot i should quickly adapt its motion planner in response to the newly learned models. Since function $f^{[i]}$ and the estimates of function $g^{[i]}$ are nonlinear in general, the motion planner will be adapted by a numerical algorithm.

III. Distributed Safe Learning and Planning

In this section, we propose the dSLAP framework (Algorithm 1). Figure 1 shows one iteration of the algorithm in robot i. In each iteration k, the robot executes two modules *in parallel*. One is the computation module where robot i sequentially performs system learning (SL), safe motion planning that includes dynamics discretization (Discrete), obstacle collision avoidance (OCA) and inter-robot collision avoidance (ICA), and active learning (AL) that synthesizes control policy $\pi_k^{[i]}$. The other is the control module where the control policy $\pi_{k-1}^{[i]}$, computed in iteration $k-1$, is applied for all $t \in [k\xi,(k+1)\xi)$, where ξ is the discrete time unit.

Algorithm 1: The dSLAP framework

Input: $\mathcal{X}$, $\mathcal{U}$, $\mathcal{X}_O$, $\mathcal{X}_G^{[i]}$, κ, p_{init}, $\bar{p}$, $\tilde{k}$, $\bar{\tau}$, ξ, φ, ψ, δ, $r_k^{[i]}$, $\ell^{[i]}$, $m^{[i]}$;
Init: $p_1 \leftarrow p_{init}$; $\pi_0^{[i]}, \forall i \in \mathcal{V}$;
for $k = 1,2,\cdots,\tilde{k}$ **do**
  for $i \in \mathcal{V}$ *(Computation module)* **do**
    $\mathcal{D}_k^{[i]} \leftarrow$ CollectData;
    $\mu_k^{[i]}, \sigma_k^{[i]} \leftarrow \mathsf{SL}(\mathcal{D}_k^{[i]})$;
    $\mathcal{X}_{p_k}, \mathcal{U}_{p_k}, h_{p_k} \leftarrow \mathsf{Discrete}(p_k)$;
    $\mathcal{X}_{safe,k}^{[i]} \leftarrow$ OCA;
    $\mathcal{X}_{safe,k}^{[i]} \leftarrow$ ICA;
    $\pi_k^{[i]} \leftarrow \mathsf{AL}(\mathcal{X}_{safe,k}^{[i]})$;
    $p_{k+1} \leftarrow \min\{p_k+1, \bar{p}\}$;
  for $i \in \mathcal{V}$ *(Control module)* **do**
    Execute$(\pi_{k-1}^{[i]}, [k\xi,(k+1)\xi))$;

A. System learning

In this section, we introduce the SL procedure for learning the external disturbance $g^{[i]}$. In each iteration k, each robot i first collects a new dataset $\mathcal{D}_k^{[i]}$ through the CollectData procedure, which returns $\mathcal{D}_k^{[i]} \triangleq \{g^{[i]}(x^{[i]}(\tau),u^{[i]}(\tau)) + e^{[i]}(\tau), x^{[i]}(\tau), u^{[i]}(\tau)\}_{\tau=(k-1)\xi}^{(k-1)\xi+\delta\bar{\tau}}$, where $e^{[i]}(\tau) \sim \mathcal{N}(0,(\sigma_e^{[i]})^2 I_{n_x})$

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.

is robot i's local observation error, δ is the sampling period, and $\bar{\tau}$ is the number of samples to be obtained. Then robot i independently estimates $g^{[i]}$ using Gaussian process regression (GPR) [13]. By specifying prior mean function $\mu_0 : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^{n_x}$, and prior covariance function $\kappa : [\mathcal{X} \times \mathcal{U}] \times [\mathcal{X} \times \mathcal{U}] \to \mathbb{R}_{>0}$, GPR models $g^{[i]}$ as a sample from a Gaussian process prior $\mathcal{GP}(\mu_0, \kappa)$ and predicts $g^{[i]}(x^{[i]}, u^{[i]}) \sim \mathcal{N}(\mu_k^{[i]}(x^{[i]}, u^{[i]}), (\sigma_k^{[i]}(x^{[i]}, u^{[i]}))^2)$.

B. Safe motion planning

Safe motion planning is a multi-grid algorithm utilizing set-valued analysis. Inspired by [10] [11], we propose a new set-valued dynamics to discretize robot dynamics (1). We use the set-valued dynamics to approximate the one-step forward set and construct a directed graph. We then identify safe states and remove control inputs which lead to collision.

Dynamics discretization. As in [10] [11], treating the estimation errors of $g^{[i]}$ as adversarial inputs, we approximate system (1) using the following discretized set-valued dynamic system: $\Gamma_k^{[i]}(x^{[i]}, u^{[i]}) \triangleq [x^{[i]} + \epsilon_{p_k}^{[i]}\big((f^{[i]} + \mu_k^{[i]})(x^{[i]}, u^{[i]}) + \gamma\sigma_k^{[i]}(x^{[i]}, u^{[i]})\mathcal{B}\big) + \alpha_{p_k}^{[i]}\mathcal{B}] \cap \mathcal{X}_{p_k}$, where $\epsilon_{p_k}^{[i]}$ is the temporal resolution, $\alpha_{p_k}^{[i]} \triangleq 2h_{p_k} + 2\epsilon_{p_k}^{[i]}h_{p_k}\ell^{[i]} + (\epsilon_{p_k}^{[i]})^2\ell^{[i]}m^{[i]}$ is the dilation term, h_{p_k} is the spatial resolution, $\mathcal{X}_{p_k}$ is the discrete state space, $\mathcal{U}_{p_k}$ is the discrete control space such that $u^{[i]} \in \mathcal{U}_{p_k}$, and p_k is the discretization parameter at iteration k. Through Discrete, robot i sets

$$h_{p_k} \triangleq 2^{-p_k}, \mathcal{X}_{p_k} \triangleq h_{p_k}\mathbb{Z}^{n_x} \cap \mathcal{X}, \mathcal{U}_{p_k} \triangleq h_{p_k}\mathbb{Z}^{n_u} \cap \mathcal{U}. \quad (2)$$

Set temporal resolution as $\epsilon_{p_k}^{[i]} \triangleq \epsilon^{[i]} = \lambda^{[i]}\xi$, where $\lambda^{[i]}$ is the constant that ensures each iteration k with duration ξ can be partitioned into an integer number of small intervals with duration $\epsilon^{[i]}$. Further augmenting the dilation term $\alpha_{p_k}^{[i]}$ with h_{p_k}, we define the one-step forward set of duration $\epsilon^{[i]}$ by

$$\mathsf{FR}_k^{[i]}\big(x^{[i]}, u^{[i]}\big) \triangleq \Big[x^{[i]} + \epsilon^{[i]}(f^{[i]}(x^{[i]}, u^{[i]}) + \mu_k^{[i]}(x^{[i]}, u^{[i]})) + (\epsilon^{[i]}\gamma\bar{\sigma}_k^{[i]} + \alpha_{p_k}^{[i]} + h_{p_k})\mathcal{B}\Big] \cap \mathcal{X}_{p_k},$$

where $\bar{\sigma}_k^{[i]} \triangleq \sup_{x^{[i]} \in \mathcal{X}, u^{[i]} \in \mathcal{U}} \sigma_k^{[i]}(x^{[i]}, u^{[i]})$. Finer discretization, corresponding to a larger p_k, provides better approximation of the dynamic model, whereas coarser discretization returns solutions faster. Hence, we increment the discretization parameter at each iteration to refine the discretization while feasible solutions are synthesized at lower resolution. We set a maximum $\bar{p}$ to prevent prolonged computation due to unnecessarily fine discretization.

Obstacle collision avoidance (Algorithm 2). Procedure OCA aims to identify the safe states of the set-valued dynamic system and remove the control inputs that lead to collision with the obstacles. Informally, a state is safe if there is a controller that can keep the robot from colliding with the obstacles when the robot starts from the state. Otherwise, the state is unsafe. Then procedure OCA consists of two steps. First, it identifies the unsafe states where their one-step forward sets are too close to the obstacles. Second, UnsafeUpdate (Algorithm 3) searches the remaining unsafe states backward and removes the control inputs leading to the unsafe states and the obstacles.

Algorithm 2: Procedure OCA

$\mathcal{X}_{unsafe,k,0}^{[i]} \leftarrow \emptyset$;
for $x^{[i]} \in \mathcal{X}_{p_k}$ **do**
  $\mathcal{U}_{p_k}^{[i]}(x^{[i]}) \leftarrow \mathcal{U}_{p_k}$;
  if $\rho(x_q^{[i]}, \mathcal{X}_O) > m^{[i]}\epsilon^{[i]} + h_{p_k}$ **then**
    for $u^{[i]} \in \mathcal{U}_{p_k}^{[i]}(x^{[i]})$ **do**
      if *Collision*$(\mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]}), k) == 1$ **then**
        Remove$(\mathcal{U}_{p_k}^{[i]}(x^{[i]}), u^{[i]})$
      else
        for $y^{[i]} \in \mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]})$ **do**
          Add$(\mathsf{BR}_k^{[i]}(y^{[i]}, u^{[i]}), x^{[i]})$
  if $\rho(x_q^{[i]}, \mathcal{X}_O) \leqslant m^{[i]}\epsilon^{[i]} + h_{p_k}$ *or* $\mathcal{U}_{p_k}^{[i]}(x^{[i]}) = \emptyset$ **then**
    Add$(\mathcal{X}_{unsafe,k,0}^{[i]}, x^{[i]})$;
$\bar{\mathcal{X}}_{unsafe,k,0}^{[i]} \leftarrow$ UnsafeUpdate$(\mathcal{X}_{unsafe,k,0}^{[i]}, k)$;
$\mathcal{X}_{safe,k}^{[i]} \leftarrow \mathcal{X}_{p_k} \setminus \bar{\mathcal{X}}_{unsafe,k,0}^{[i]}$;
Return $\mathcal{X}_{safe,k}^{[i]}$;

The first step is accomplished by the outer for-loop in OCA. In particular, for each state $x^{[i]} \in \mathcal{X}_{p_k}$ that is sufficiently distant from the obstacles, collisions are checked over all the one-step forward sets $\mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]})$ with inputs $u^{[i]} \in \mathcal{U}_{p_k}$. This is done by procedure Collision, where

Collision$(\mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]}), k)$ returns 1
if $\exists y^{[i]} \in \mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]})$ such that
$\rho(x_q^{[i]}, \mathcal{X}_O) \leqslant m^{[i]}\epsilon^{[i]} + h_{p_k}$; otherwise, returns 0.

The control input is removed if the corresponding one-step forward set collides with the obstacles. Otherwise, the set $\mathsf{BR}_k^{[i]}$, the one-step $\epsilon^{[i]}$-duration backward set of $y^{[i]}$ applied $u^{[i]}$, is constructed as follow

$$\forall x^{[i]} \in \mathsf{BR}_k^{[i]}(y^{[i]}, u^{[i]}), y^{[i]} \in \mathsf{FR}_k^{[i]}(x^{[i]}, u^{[i]}).$$

If all the control inputs in $\mathcal{U}_{p_k}^{[i]}(x^{[i]})$ are removed, state $x^{[i]}$ is identified as unsafe and included in the set $\mathcal{X}_{unsafe,k,0}^{[i]}$, together with the states that are within $m^{[i]}\epsilon^{[i]} + h_{p_k}$ of the obstacles. Notice that the distance $m^{[i]}\epsilon^{[i]} + h_{p_k}$ represents an over-approximation of the distance the robot can reach within one time step with size $\epsilon^{[i]}$ on $\mathcal{X}_{p_k}$. This distance prevents the robot from "cutting the corner" of the obstacles due to the discrete approximation.

In the second step, robot i runs procedure UnsafeUpdate to iteratively remove all the control inputs that lead to identified unsafe states. Robot i's set of unsafe states $\bar{\mathcal{X}}_{unsafe,k,0}^{[i]}$ is then completed. For each state in $\mathcal{X}_{p_k} \setminus \bar{\mathcal{X}}_{unsafe,k,0}^{[i]}$, we have $\mathcal{U}_{p_k}^{[i]}(x^{[i]}) \neq \emptyset$ and any control $u^{[i]} \in \mathcal{U}_{p_k}^{[i]}(x^{[i]})$ can ensure collision avoidance with the obstacles for one iteration.

Inter-robot collision avoidance (Algorithm 4). Procedure ICA aims to remove the control inputs that lead to collision

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3: UnsafeUpdate($\mathcal{X}^{[i]}_{unsafe,k,j}, k$)

$\bar{\mathcal{X}}^{[i]}_{unsafe,k,j} \leftarrow \mathcal{X}^{[i]}_{unsafe,k,j}$;
$Flag \leftarrow 1$;
while $Flag == 1$ **do**
  $\mathcal{X}^{[i]}_{new} \leftarrow \mathcal{X}^{[i]}_{unsafe,new}$;
  $Flag \leftarrow 0$;
  for $y^{[i]} \in \mathcal{X}^{[i]}_{new}$ **do**
    for $u^{[i]} \in \mathcal{U}_{p_k}$ **do**
      for $x^{[i]} \in BR^{[i]}_k(y^{[i]}, u^{[i]})$ **do**
        Remove($\mathcal{U}^{[i]}_{p_k}(x^{[i]}), u^{[i]}$);
        if $\mathcal{U}^{[i]}_{p_k}(x^{[i]}) == \emptyset$ *and* $x^{[i]} \notin \bar{\mathcal{X}}^{[i]}_{unsafe,k,j}$ **then**
          Add($\bar{\mathcal{X}}^{[i]}_{unsafe,k,j}, x^{[i]}$);
          $Flag \leftarrow 1$;
Return $\bar{\mathcal{X}}^{[i]}_{unsafe,k,j}$;

Algorithm 4: Procedure ICA

$\mathcal{X}^{[i]}_k \leftarrow$
 $x^{[i]}_q(k\xi) + (2\xi m^{[i]} + 2\zeta + m^{[i]}\epsilon^{[i]} + \frac{1}{2}\alpha^{[i]}_{p_k} + 3h_{p_k})\mathcal{B}$;
Broadcast($\mathcal{X}^{[i]}_k$);
for $j \in \mathcal{V}$, $j \neq i$ **do**
  if $j < i$ **then**
    $\mathcal{X}^{[i]}_{unsafe,k,j} \leftarrow \big[[\mathcal{X}^{[j]}_k + \epsilon^{[i]}\gamma\bar{\sigma}^{[i]}_k\mathcal{B}] \cup \mathcal{X}_r\big] \cap \mathcal{X}_{p_k}$;
    $\bar{\mathcal{X}}^{[i]}_{unsafe,k,j} \leftarrow$
     UnsafeUpdate($\mathcal{X}^{[i]}_{unsafe,k,j}, k$);
    $\mathcal{X}^{[i]}_{safe,k} \leftarrow \mathcal{X}^{[i]}_{safe,k} \setminus \bar{\mathcal{X}}^{[i]}_{unsafe,k,j}$;
Return $\mathcal{X}^{[i]}_{safe,k}$;

with the robots with higher priority. It treats the robots with higher priority as moving obstacles and removes the control inputs that lead to these obstacles. First, each robot i broadcasts its reachability sets $\mathcal{X}^{[i]}_k$ within an iteration at the beginning of each iteration k. Upon receiving the messages from robot j with higher priority, i.e., $j < i$, robot i identifies a new set of unsafe states $\mathcal{X}^{[i]}_{unsafe,k,j}$ induced by $\mathcal{X}^{[i]}_k$ in the discrete state space $\mathcal{X}_{p_k}$. Second, robot i invokes procedure UnsafeUpdate to remove all the control inputs leading to the newly identified unsafe states. Robot i then updates the set of the safe states $\mathcal{X}^{[i]}_{safe,k}$ by removing the new unsafe states $\bar{\mathcal{X}}^{[i]}_{unsafe,k,j}$. For each state $x^{[i]} \in \mathcal{X}^{[i]}_{safe,k}$, $\mathcal{U}^{[i]}_k(x^{[i]}) \neq \emptyset$ and any control $u^{[i]} \in \mathcal{U}^{[i]}_k(x^{[i]})$ ensures collision avoidance with the obstacles and the robots with higher priority. In the worst case, a robot removes all the states in its own state space, where the computation complexity is independent of the number of robots.

C. Active learning and real-time control (Algorithm 5)

In this section, we utilize the safe control inputs obtained above and synthesize a model predictive controller (MPC) to actively learn the disturbance $g^{[i]}$ and approach the goal.

Algorithm 5: AL

Procedure $\pi^{[i]}_k(x^{[i]}(t))$;
  $w[k] \leftarrow e^{-\psi k}$;
  $\hat{x}^{[i]}(t) \leftarrow$ Nearest($x^{[i]}(t), \mathcal{X}^{[i]}_{safe,k}$);
  $(u^{[i]}_*(t), \cdots, u^{[i]}_*(t + \varphi\epsilon^{[i]})) \leftarrow$ solve MPC in (3);
Return $u^{[i]}_*(t)$;

First, state $x^{[i]}(t)$ is projected onto $\mathcal{X}^{[i]}_{safe,k}$; the projection is $\hat{x}^{[i]}(t) \triangleq$ Nearest($x^{[i]}(t), \mathcal{X}^{[i]}_{safe,k}$). Second, we capture the objective of goal reaching using distance $\rho(\hat{x}^{[i]}(t + \varphi\epsilon^{[i]}), \mathcal{X}^{[i]}_G)$, where $\varphi \in \mathbb{N}$ is the discrete horizon of the MPC formulated below. Then the objective of exploration is described by a utility function $r^{[i]}_k(\hat{x}^{[i]}(t), u^{[i]}(t))$; candidate utility functions, e.g., $r^{[i]}_k(\hat{x}^{[i]}(t), u^{[i]}(t)) = \sigma^{[i]}_k(\hat{x}^{[i]}(t), u^{[i]}(t))$, are available in [14]. Next, the safety constraint is honored by choosing control inputs from the safe control set $\mathcal{U}^{[i]}_k(\hat{x}^{[i]}(t))$. Lastly, the dynamic constraint is approximated by the one-step forward set $\mathsf{FR}^{[i]}_k$. Formally, the controller $\pi^{[i]}_k : \mathcal{X} \to \mathcal{U}$ is synthesized by solving the finite-horizon optimal control:

$$\begin{aligned} \min \quad & (1 - w[k])\rho(\hat{x}^{[i]}(t + \varphi\epsilon^{[i]}), \mathcal{X}^{[i]}_G) \\ & + w[k] \sum_{\tau=t}^{t+\varphi\epsilon^{[i]}} r^{[i]}_k(\hat{x}^{[i]}(\tau), u^{[i]}(\tau)), \end{aligned} \tag{3}$$

where the decision variables are $u^{[i]}(t) \in \mathcal{U}^{[i]}_k(\hat{x}^{[i]}(t)), \cdots, u^{[i]}(t + \varphi\epsilon^{[i]}) \in \mathcal{U}^{[i]}_k(\hat{x}^{[i]}(t + \varphi\epsilon^{[i]}))$, subject to $\hat{x}^{[i]}(\tau + \epsilon^{[i]}) \in \mathsf{FR}^{[i]}_k(\hat{x}^{[i]}(\tau), u^{[i]}(\tau))$ and $\tau \in \{t, t + \epsilon^{[i]}, \cdots, t + (\varphi - 1)\epsilon^{[i]}\}$. To ensure the robot eventually reaches the goal, we select the weight $w[k] \triangleq e^{-\psi k}$ for some $\psi > 0$ such that $w[k]$ diminishes.

The above finite-horizon optimal control problem is solved once for every time duration $\epsilon^{[i]}$, and the returned control input is fixed for a duration $\epsilon^{[i]}$. Specifically, consider a sequence $\{t^{[i]}_{k+1,n}\}_{n=0}^{\bar{n}^{[i]}_{k+1}} \subset [(k+1)\xi, (k+2)\xi]$, where $t^{[i]}_{k+1,0} = (k+1)\xi$, $t^{[i]}_{k+1,n} = t^{[i]}_{k+1,n-1} + \epsilon^{[i]}$ and $\bar{n}^{[i]}_{k+1} \triangleq \xi/\epsilon^{[i]}$. Procedure $\pi^{[i]}_k(x^{[i]}(t^{[i]}_{k+1,n}))$ solves the above finite-horizon optimal control problem at $n = 0, 1, \cdots, \bar{n}^{[i]}_{k+1} - 1$. The solution has the form $(u^{[i]}_*(t^{[i]}_{k+1,n}), \cdots, u^{[i]}_*(t^{[i]}_{k+1,n} + \varphi\epsilon^{[i]}))$, and $\pi^{[i]}_k(x^{[i]}(t^{[i]}_{k+1,n})) = u^{[i]}_*(t^{[i]}_{k+1,n})$ is returned. For all $t \in [t^{[i]}_{k+1,n}, t^{[i]}_{k+1,n+1})$, we have $u^{[i]}(t) = u^{[i]}_*(t^{[i]}_{k+1,n})$. The controller execution is performed in Algorithm 1.

D. Performance guarantees

In this section, we provide the performance guarantees for dSLAP. To obtain theoretic guarantees, we assume that $g^{[i]}$ is correctly specified by a known Gaussian process. For notational simplicity, we assume $g^{[i]} \in \mathbb{R}$. Generalizing to multi-dimensional can be done by applying the union bound.

Assumption III.1. *(Specified process).* It satisfies that $g^{[i]} \in \mathbb{R}$ and $g^{[i]} \sim \mathcal{GP}(\mu_0, \kappa)$. ■

That is, function $g^{[i]}$ is completely specified by a Gaussian process with prior mean μ_0 and kernel κ. This assumption is

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.

common in the analysis of GPR (Theorem 1, [15]). Theorem III.2 below shows that the safety of the robots lasts until the end of an iteration with high probability if they are around the set of safe states at the beginning of the iteration.

Theorem III.2. Suppose Assumptions II.1 and III.1 hold. If $\mathcal{B}(x^{[i]}(k\xi), h_{p_{k-1}}) \cap \mathcal{X}^{[i]}_{safe,k-1} \neq \emptyset$, $k \geqslant 1$, for all $i \in \mathcal{V}$, then dSLAP renders $x^{[i]}_q(t) \in \mathcal{X}^{[i]}_F(x^{[\neg i]}_q(t))\ \forall t \in [k\xi, k\xi+\xi)$ with probability at least $1 - |\mathcal{V}||\mathcal{X}_p||\mathcal{U}_p|e^{-\gamma^2/2}$. ■

IV. Simulation

In this section, we conduct Monte Carlo simulations to evaluate the dSLAP algorithm. The simulations are run in Python, Linux Ubuntu 18.04 on an Intel Xeon(R) Silver 4112 CPU, 2.60 GHz with 32 GB of RAM.

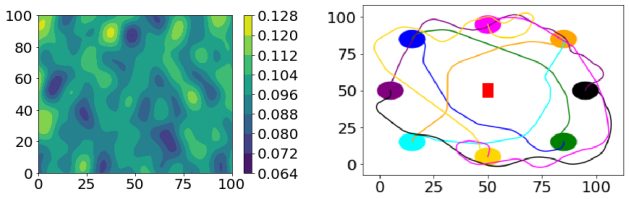
Simulation scenarios. We evaluate the dSLAP algorithm using Zermelo's navigation problem [16] in a 2D space under the following scenario: A group of robots are initially placed evenly on the plane and switch their positions at the destinations. The robots are immediately retrieved once they reach the goals. This example is also used in [17] [18] to demonstrate complicated multi-robot coordination scenarios.

Dynamic models. Consider constant-speed boat robots with length $L = 1.5$ meters (m) moving at speed $v = 0.5$ meters/seconds (m/s). For each robot i, let $x^{[i]}_{q,1}$ and $x^{[i]}_{q,2}$ be the coordinates on a 2D plane, $x^{[i]}_r$ be the angle between the heading and the x-axis, and $u^{[i]}$ be the steering angle. The state space is given by $\mathcal{X} = [0, 100] \times [0, 100] \times [-\pi, \pi]$. External wind disturbance ν is applied at $x^{[i]}_{q,1}$ such that the system dynamics has the following form: $\dot{x}^{[i]}_{q,1}(t) = 2\cos x^{[i]}_r(t)$, $\dot{x}^{[i]}_{q,2}(t) = 2\sin x^{[i]}_r(t)$, $\dot{x}^{[i]}_r(t) = \frac{0.5}{1.5}\tan u^{[i]}(t)$. The control $u^{[i]}$ takes discrete values and the control space is $\mathcal{U} = \{\pm 0.3\pi, \pm 0.15\pi, 0\}$.

Parameters. The kernel of GPR is configured as $\kappa(z, z') = 0.0025\exp(-\frac{\|z-z'\|_2^2}{2})$, which is 0.0025 times the RBF kernel in the sklearn library. The factor 0.0025 is selected such that the supremum of the predictive standard deviation is 0.05, or 10% of the robots' speed. This can be selected based on the prior knowledge of the variability of the disturbance. Other parameters are selected as $\gamma = 1$, $p_{init} = 4$, $\bar{p} = 5$, $\bar{k} = 200$, $\bar{\tau} = 20$, $\xi = 8$, $q = 2$, $\psi = 1$, $\delta = 0.1$, and $r^{[i]}_k = -\sigma^{[i]}_k$, which are determined according to the desired confidence level and the computation capability of the robots.

A. Multi-robot maneuver.

Wind fields and initial configurations. We evaluate dSLAP using 30,000 scenarios generated as follows. We randomly generate 2D spatial wind fields, with average speed ν in different ratios of the robots' speed, i.e., $\nu = r_w v$, $r_w > 0$, and standard deviation 2% of the robots' speed, using the Von Karman power spectral density function as described in [19]. This wind model is used to test multi-robot navigation in [19] [20]. A sample with $r_w = 0.2$ is shown in Figure 2a. We randomly generate 60 different wind fields for each $r_w \in \{0.1, 0.2, \cdots, 1\}$. We deploy n robots with 10 different initial configurations in the simulation, where $n \in \{1, 2, 4, 6, 8\}$. Figure 2b shows one configuration of 8 robots' initial states and goal regions, and the corresponding trajectories under dSLAP in the wind field in Figure 2a. The circular disks are the goal regions of the robots and the red rectangle is the static obstacle. Other configurations are generated by different permutations and removals of robots from that in Figure 2b.



(a) A sample of wind field experienced by the robots

(b) Trajectories of the robots

Fig. 2: A sample of wind fields and robot trajectories

Ablation study. To the best of our knowledge, this paper is the first to consider extreme scenarios where backup policies are absent, in contrast to [5]–[9]. Hence, we compare dSLAP with its three variants, Vanilla, Robust and Known, that do not learn the wind disturbances. Vanilla assumes $g^{[i]} = 0$, $\forall i \in \mathcal{V}$, whereas Robust assumes $\sup_{x^{[i]} \in \mathcal{X}, u^{[i]} \in \mathcal{U}} |g^{[i]}(x^{[i]}, u^{[i]})| \leqslant \hat{r}_w v$ and thus $\dot{x}^{[i]} \in f^{[i]}(x^{[i]}, u^{[i]}) + \hat{r}_w v\mathcal{B}$, where $\hat{r}_w > 0$. We adopt $\hat{r}_w = 0.1$ such that Robust has the same level of conservativeness as dSLAP before collecting any data. The benchmark Known is obtained by running the dSLAP framework with the disturbances exactly known, which is equivalent to dSLAP with an infinite amount of data.

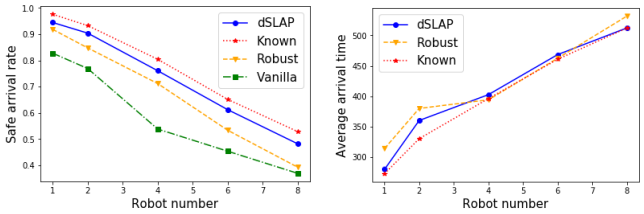
Results. The average safe arrival rates of dSLAP, Robust, Vanilla and Known among the 30,000 cases are shown in Figure 3a. From Figure 3a, we can see that dSLAP's performance is superior to those of Robust and Vanilla. This is due to the fact that dSLAP online learns about the unknown disturbances and adjusts the policies accordingly. On the other hand, Robust (or Vanilla) only captures part of (or none of) the disturbances through the prior estimates, which can be unsafe when the disturbances exceed the estimates. Furthermore, we can observe that the safe arrival rate for dSLAP decreases linearly with respect to the number of robots. This corresponds to the probability $1 - |\mathcal{V}||\mathcal{X}_p||\mathcal{U}_p|e^{-\gamma^2/2}$ in Theorem III.2. Notice that the gap between Known and dSLAP is small. This indicates that dSLAP enables safe arrival in most feasible cases. Figure 3b compares the average safe arrival times among dSLAP, Robust and Known. We exclude the comparison with Vanilla since its safe arrival rate is far lower than the other three while safety is this paper's top priority. The arrival times of the three algorithms are comparable. This indicates dSLAP improves safe arrival rate without sacrificing arrival time, i.e., being more conservative.

Run-time computation. This section shows the wall computation time of dSLAP when the robots are deployed in the wind field in Figure 2a with configuration in Figure 2b, as an

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.

ID	Total time	SL		Discrete+OCA		ICA		AL	
		time	Percentage	time	Percentage	time	Percentage	time	Percentage
1	5.71±0.45	0.84±7.87e^{-3}	14.73±1.08	4.26±0.12	74.91±3.95	6.03e^{-3}±1.85e^{-3}	0.11±9.44e^{-3}	0.61±0.32	10.26±5.07
2	5.93±0.47	0.81±0.01	13.70±1.03	4.21±0.06	71.32±5.10	0.06±0.03	1.08±0.59	0.85±0.41	13.90±6.38
3	5.69±0.34	0.82±1.43e^{-3}	14.48±0.88	4.14±0.04	73.10±3.90	0.14±0.01	2.56±0.32	0.58±0.32	9.87±5.08
4	5.18±0.14	0.82±2.03e^{-3}	15.85±0.43	4.01±0.04	77.37±1.98	0.16±0.09	3.07±1.76	0.20±0.15	3.70±2.81
5	5.90±0.35	0.82±0.01	13.96±0.65	4.30±0.08	73.10±3.41	0.23±0.03	3.96±0.51	0.54±0.27	8.98±4.10
6	5.66±1.29	0.88±0.16	15.7±0.65	4.55±1.08	80.26±1.58	0.13±0.11	2.57±2.23	0.10±0.13	1.46±1.64
7	5.94±0.99	0.88±0.10	14.96±0.82	4.49±0.65	75.85±2.69	0.19±0.10	3.39±2.03	0.38±0.38	5.79±5.19
8	5.94±1.14	0.88±0.13	14.90±0.55	4.65±0.83	78.53±1.30	0.29±0.07	5.02±1.62	0.12±0.22	1.56±2.45

TABLE I: Computation time (seconds) for each robot in one iteration



(a) Percentage of safe arrivals (b) Average time of safe arrivals

Fig. 3: Ablation study of dSLAP

# of robots	1	2	4	6	8
Wall time	5.837± 0.085	5.843± 0.118	5.830± 0.102	5.832± 0.129	5.839± 0.119

TABLE II: Wall clock time (seconds) per iteration

example. Table I presents the average plus/minus one standard deviation of computation time for one iteration for each component of dSLAP and the corresponding percentages (%) of the total computation time. Discrete+OCA consumes most of the computation resources because a discrete set-valued approximation of the continuous dynamics over the entire state-action space is constructed through these two procedures, especially in OCA. Table I shows that the computation costs of the other procedures are mostly sub-second. Table II shows that the average wall time plus/minus one standard deviation per iteration versus the number of robots deployed. This implies that the computation time within each robot is independent of the number of the robots.

V. Conclusion

We study the problem where a group of mobile robots subject to unknown external disturbances aim to safely reach goal regions. We propose dSLAP that enables the robots to quickly adapt to a sequence of learned models resulted from online GPR, and safely reach the goal regions. Sufficient conditions to ensure the safety of the system is derived. The developed algorithm is evaluated by Monte Carlo simulation.

References

[1] K. R. Varshney, "Engineering safety in machine learning," in *Information Theory and Applications Workshop (ITA)*, 2016, pp. 1–5.

[2] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[3] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *Proc. IEEE Conf. Decision and Control (CDC)*, 2016, pp. 4661–4666.

[4] I. Usmanova, A. Krause, and M. Kamgarpour, "Safe convex learning under uncertain constraints," in *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2019, pp. 2106–2114.

[5] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Trans. Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[6] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss, "A general framework to increase safety of learning algorithms for dynamical systems based on region of attraction estimation," *IEEE Trans. Robotics*, 2020.

[7] K. P. Wabersich and M. N. Zeilinger, "Linear model predictive safety certification for learning-based control," in *Proc. IEEE Conf. Decision and Control (CDC)*, 2018, pp. 7130–7135.

[8] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *Proc. IEEE Conf. Decision and Control (CDC)*, 2018, pp. 6059–6066.

[9] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, vol. 33, no. 01, 2019, pp. 3387–3395.

[10] G. Zhao and M. Zhu, "Pareto optimal multi-robot motion planning," *IEEE Trans. Automatic Control*, vol. 66, no. 9, pp. 3984–3999, 2021.

[11] P. Cardaliaguet, M. Quincampoix, and P. Saint-Pierre, "Set-valued numerical analysis for optimal control and differential games," in *Stochastic and differential games*. Springer, 1999, pp. 177–247.

[12] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Annual Symposium on Foundations of Computer Science*, 1979, pp. 421–427.

[13] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[14] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.

[15] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Trans. Information Theory*, vol. 58, no. 5, pp. 3250–3265, Jan. 2012.

[16] S. Zlobec, *Zermelo's Navigation Problems*. Boston, MA: Springer US, 2001.

[17] H. G. Tanner and A. Boddu, "Multiagent navigation functions revisited," *IEEE Trans. Robotics*, vol. 28, no. 6, pp. 1346–1359, 2012.

[18] O. Arslan, D. P. Guralnik, and D. E. Koditschek, "Coordinated robot navigation via hierarchical clustering," *IEEE Trans. Robotics*, vol. 32, no. 2, pp. 352–371, 2016.

[19] K. Cole and A. Wickenheiser, "Impact of wind disturbances on vehicle station keeping and trajectory following," in *AIAA Guidance, Navigation, and Control Conference*, 2013, p. 4865.

[20] K. Cole and A. M. Wickenheiser, "Reactive trajectory generation for multiple vehicles in unknown environments with wind disturbances," *IEEE Trans. Robotics*, vol. 34, no. 5, pp. 1333–1348, 2018.

Authorized licensed use limited to: Penn State University. Downloaded on February 02,2023 at 01:11:11 UTC from IEEE Xplore. Restrictions apply.