2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Configuration Space Decomposition for Learning-based Collision Checking in High-DOF Robots

Yiheng Han[1], Wang Zhao[1], Jia Pan[2] and Yong-Jin Liu[1†]

***Abstract*— Motion planning for robots of high degrees-of-freedom (DOFs) is an important problem in robotics with sampling-based methods in configuration space $\mathcal{C}$ as one popular solution. Recently, machine learning methods have been introduced into sampling-based motion planning methods, which train a classifier to distinguish collision free subspace from in-collision subspace in $\mathcal{C}$. In this paper, we propose a novel configuration space decomposition method and show two nice properties resulted from this decomposition. Using these two properties, we build a composite classifier that works compatibly with previous machine learning methods by using them as the elementary classifiers. Experimental results are presented, showing that our composite classifier outperforms state-of-the-art single-classifier methods by a large margin. A real application of motion planning in a multi-robot system in plant phenotyping using three UR5 robotic arms is also presented.**

I. INTRODUCTION

Motion planning plays an important role in robotics, which finds a collision-free path to move a robot from a source to a target position. Configuration space $\mathcal{C}$ [1] is widely used in robot motion planning, whose spatial dimensions characterize the degrees-of-freedom (DOFs) of the robot and each point in $\mathcal{C}$ represents a configuration of the robot. By decomposing the space $\mathcal{C}$ into a free subspace $\mathcal{C}_{free}$ (i.e., the set of robot configurations without self-collision or collision with obstacles) and an in-collision subspace $\mathcal{C}_{clsn} = \mathcal{C} \setminus \mathcal{C}_{free}$, motion planning is equivalent to finding a path completely within $\mathcal{C}_{free}$.

For robots of high DOFs (such as 6-DOF robotic arms UR5) in complex or dynamic environments, the boundary of $\mathcal{C}_{clsn}$ is very complicated and usually cannot be represented analytically [1], [2], [3]. Sampling-based motion planning (SBMP) methods (e.g., [4], [5]) were then proposed to use sample points for characterizing $\mathcal{C}_{free}$ and avoid explicitly establishing the boundary of $\mathcal{C}_{clsn}$. Some representative researches include the classic probabilistic roadmaps (PRM) [6], rapidly-exploring random trees (RRT) [7], [8], the variants of PRM and RRT (e.g., [5]), and some state of the arts [9], [10].

*This work was supported by the Natural Science Foundation of China (61725204, 61521002) and Grant from Science and Technology Department of Jiangsu Province, China.

[1]Y. Han, W. Zhao and Y-J Liu are with BNRist, MOE-Key Laboratory of Pervasive Computing, Department of Computer Science and Technology, Tsinghua University, Beijing, China. hyh18@mails., zhao-w19@mails. and liuyongjin@}tsinghua.edu.cn

[2]J. Pan is with the Department of Computer Science, The University of Hong Kong. jpan@cs.hku.hk

[†]Corresponding author

To implicitly define $\mathcal{C}_{free}$, a large number of samples are required in SBMP methods and each sample is guaranteed to be collision-free by passing an exact collision detector such as GJK [11] or FCL [12], which is very time-consuming. To speed up the collision checking procedure, machine learning methods have been introduced into this area. As summarized in Section II, these methods first use a small subset of samples to train a classifier F. Then given an arbitrary sample s in unknown regions in $\mathcal{C}$, the classifier can output a prediction $F(s)$ that serves as a filter to quickly identify obviously in-collision or collision-free samples, and only a small set of samples with ambiguity need to be finally checked by the exact collision detector. We call these predictions by machine learning methods as *approximate* collision checking. To ensure the success of these machine learning methods, the trained classifier must have a high accuracy.

SBMP methods can be used for both single query and multiple queries. For single query, the RRT methods (e.g., [7], [8]) start at a source point in $\mathcal{C}$ and iteratively grow a search tree. At each iteration, a randomly selected point is used to drive the system with a small time step and this leads to a new vertex that is added to the tree by an edge linking it to the nearest vertex in the existing tree. The iteration terminates when the target point is reached. For multiple queries, the PRM method [6] and its variants (e.g., [5]) spread out sample points uniformly covering the whole space $\mathcal{C}_{free}$. Then given any arbitrary source and target points, a collision-free path is obtained by making use of these uniform samples

The trained classifiers in learning-based methods can improve the efficiency of both single and multiple queries in two ways. The first way is to use the prediction of the classifier to quickly identify the new collision-free samples when adding new vertices into the existing search tree. Pan and Manocha [13] propose a fast probabilistic collision checking method and prove that the collision query predicted by their classifier converges to the exact collision detection when the size of sampling points increases. Therefore, online learning is important, since the classifier needs to be online updated to improve the classification accuracy when more and more collision-free points are added into the data set. The second way is that instead of using a large number of samples to cover $\mathcal{C}_{free}$, a small set of samples can be used to train the classifier and a large number of samples that are predicted by a classifier as collision-free can be quickly generated by the classifier. After a path is planned using these samples, a final exact collision checking is applied for every

(a) Rendered system design (b) A snapshot of real system

Fig. 1. An updated multi-robot plant phenotyping system [15]. By applying our proposed composite classifier, 30% motion planning time can be saved.

sample in the path. For those samples that are in collision, a local repairing operation [14] is performed to update the path. In both way, the accuracy and query time of the classifiers are critical for the performance of motion planning.

In most previous machine learning methods, only one binary classifier was trained to predict whether a query sample point $s \in \mathcal{C}$ is in $\mathcal{C}_{free}$ or $\mathcal{C}_{clsn}$. In particular, the subspace $\mathcal{C}_{free}$ or $\mathcal{C}_{clsn}$ is learned as a single entity. In this paper, we propose a novel decomposition of $\mathcal{C}_{free}$ or $\mathcal{C}_{clsn}$, based on the DOFs of the robot. By decomposing $\mathcal{C}_{free}$ or $\mathcal{C}_{clsn}$ into a set of smaller subspaces related to the DOFs of the robot, we construct a composite classifier that consists of a set of simple classifiers and each classifier corresponds to a decomposed subspace. The advantage of this composite classifier is that the set of simple classifiers can be performed as a set of hierarchical filters that quickly filter in-collision samples from easy to hard levels.

Our composite classifier with the configuration-space decomposition scheme is general and can work compatibly with any previous machine learning methods using a single classifier (e.g., [16], [13], [17], [14]). We show that our composite classifier has a much higher accuracy and averagely 2 times faster than previous single classifier methods in most cases. We also apply our composite classifier in the motion planning of an updated multi-robot plant phenotyping system [15]. This updated system (Figure 1) consists of three UR5 robotic arms and each arm is equipped with an Intel RealSense SR-300 depth camera. To achieve fast, precise and noninvasive measurements for high-throughput plant phenotyping, all of three arms move simultaneously in each round of phenotyping data acquisition. Our results show that using the proposed composite classifier, 30% motion planning time can be saved.

II. RELATED WORK

Collision detection is frequently executed in sampling-based motion planning. In this section, we briefly review the recent machine learning methods that are introduced to speed up the time-consuming collision detection process.

An early work using machine learning is the neural network approach [18] that can only handle the collision detection for box-shaped objects. Pan and Manocha [3] design efficient GPU-based parallel k-nearest neighbors (KNN) and parallel collision detection algorithm, and propose an approximation representation for the configuration space based on machine learning. Pan and Manocha [13] further make use of KNN in online learning configuration space. To achieve fast probabilistic collision checking, they use locality-sensitive hashing techniques that only have a sub-linear time complexity. Their probabilistic collision checking can effectively improve the performance (up to 2x speedup) of various motion planners such as RRT, RRT*, PRM and lazyPRM. Das and Yip [14] propose another proxy collision detector that can achieve efficient active learning by utilizing lazy Gram matrix evaluation and a new cheaper kernel to reduce the training and query time. Heo et al. [19] develop a deep learning method that uses monitoring signals (i.e., external torque at every robotic joint) to estimate collision detection, which is applicable for industrial collaborative robots working with humans.

Most recent researches for speeding up the motion planning in SBMP methods use the idea to train a binary classifier using a small set of samples in $\mathcal{C}$ with correct labels (i.e., in-collision or collision-free) and then approximate collision checking can be quickly obtained using the prediction of the trained classifier. Various classifiers have been applied, including support vector machine (SVM) [16], k-nearest neighbor (KNN) [13], Gaussian mixture models (GMM) [17], [20] and the Gaussian kernel functions [14]. In this paper, we propose a configuration-space decomposition method that leads to a novel composite classifier. This composite classifier consists of a set of binary classifiers and each of them can be any of the above mentioned classifiers, i.e., our model orthogonally complements these previous machine learning methods [16], [13], [17], [14]. We show that our composite classifier can efficiently reduce the query time and improve the accuracy of single-classifier methods.

III. DECOMPOSITION OF CONFIGURATION SPACE

Motion planning in the configuration space $\mathcal{C}$ of high DOFs is a great challenge in robotics. In this paper, we propose a novel decomposition scheme of $\mathcal{C}$ and establish a composite classifier based on the decomposed subspaces, which performs significantly better than a single classifier working directly on $\mathcal{C}$.

Nowadays, robots of high DOFs become ubiquitous, such as robotic arms and humanoid robots. Our work is based on the following important observation. Every robot[1] $\mathcal{R}$ of high DOFs can be separated into disjointed components, satisfying

$$\mathcal{R} = \cup_{k=1}^{n_{\mathcal{R}}} \mathcal{R}_k \text{ and } \mathcal{R}_i \cap \mathcal{R}_j = \emptyset,\ \forall i \neq j,\ i,j \in \{1,2,\cdots,n_{\mathcal{R}}\} \tag{1}$$

where $n_{\mathcal{R}}$ is the number of components. Let $\mathcal{D} = \{d_1, d_2, \cdots, d_n\}$ be all the DOFs of $\mathcal{R}$, where n is the number of DOFs. For each component $\mathcal{R}_i$, one or more DOFs can be assigned to it, denoted as $\mathcal{D}_i$, such that

$$\mathcal{D} = \cup_{k=1}^{n_{\mathcal{R}}} \mathcal{D}_k \text{ and } \mathcal{D}_i \cap \mathcal{D}_j = \emptyset,\ \forall i \neq j,\ i,j \in \{1,2,\cdots,n_{\mathcal{R}}\} \tag{2}$$

All the components can be ordered in such a way that the position and orientation of each component $\mathcal{R}_i$ can be

[1]In our study, we only consider the moving part of the robot; e.g., the base of the UR5 in Figure 3 does not move and is not included.

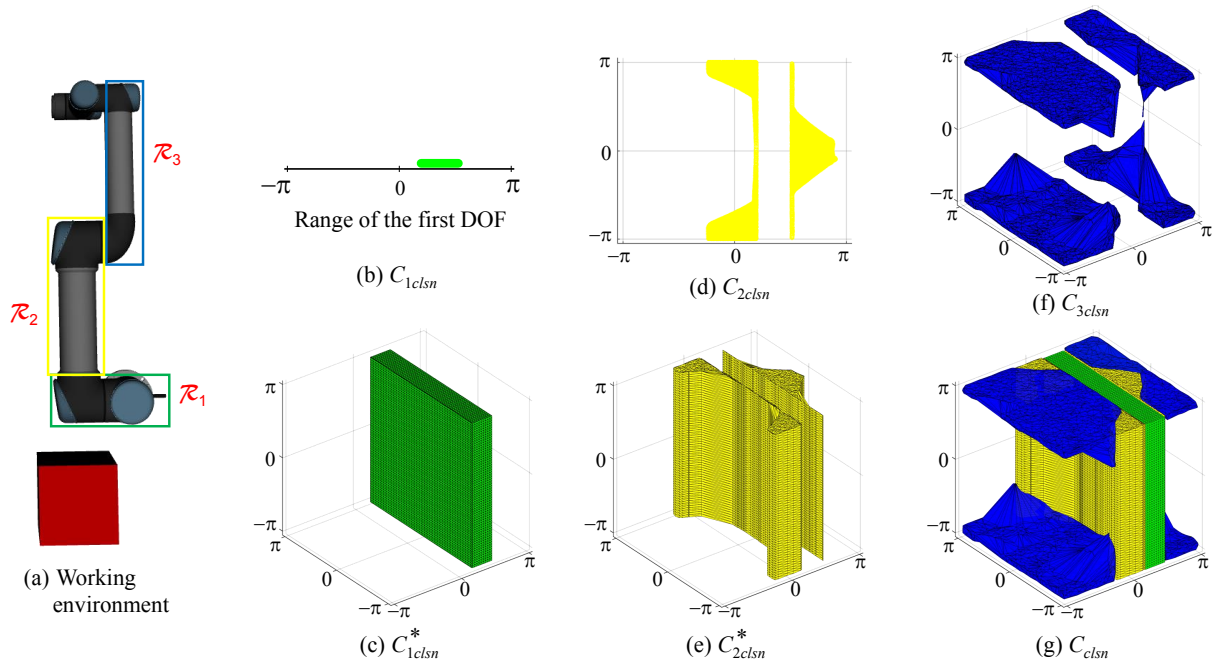

Fig. 2. (a) For easy illustration, we consider a degenerated 3-DOF UR5 robot $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$, with one cubic obstacle (shown in red). Each of three DOFs $\{d_1, d_2, d_3\}$ is ranged from $-\pi$ to π. Each $\mathcal{R}_i$, $i = 1, 2, 3$, is associated with a DOF $\mathcal{D}_i = \{d_i\}$, such that the position and orientation of $\mathcal{R}_i$ are uniquely determined by the DOFs $\{\mathcal{D}_1, \cdots, \mathcal{D}_i\}$. (b) The in-collision subspace $\mathcal{C}_{1clsn}$ (shown as the green bold line segment) in $\mathcal{C}_1$ spanned by the DOF d_1. (c) The expanded in-collision subspace $\mathcal{C}^*_{1clsn}$ (shown in green box) in $\mathcal{C}$ as defined in Eq. (4). (d) The in-collision subspace $\mathcal{C}_{2clsn}$ (shown as the yellow area) in $\mathcal{C}_2$ spanned by the DOFs $\{d_1, d_2\}$. (e) The expanded in-collision subspace $\mathcal{C}^*_{2clsn}$ (shown as the yellow volume) in $\mathcal{C}$ as defined in Eq. (6). (f) The in-collision subspace $\mathcal{C}_{3clsn}$ (shown as the blue volume) in $\mathcal{C}$ spanned by the DOFs $\{d_1, d_2, d_3\}$. (g) The final in-collision space $\mathcal{C}_{clsn} = \cup_{i=1}^{3} \mathcal{C}^*_{iclsn}$ (shown as the color volume) in $\mathcal{C}$ as defined in Eq. (7).

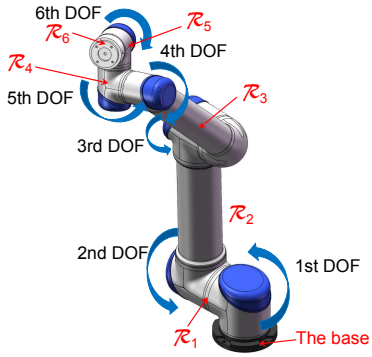

Fig. 3. The decomposition of a 6-DOF UR5 robot $\mathcal{R}$. We denote these 6 DOFs as $\mathcal{D} = \{d_1, d_2, \cdots, d_6\}$. The base of UR5 does not move and is not included into $\mathcal{R}$. $\mathcal{R}$ can be decomposed into six components $\mathcal{R} = \cup_{i=1}^{6} \mathcal{R}_i$. Each $\mathcal{R}_i$ is associated with a DOF $\mathcal{D}_i = \{d_i\}$, such that the position and orientation of each component $\mathcal{R}_i$ can be uniquely determined by the DOFs $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_i\}$.

uniquely determined by the DOFs $\{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_i\}$. One example of UR5 collaborative robot arm by Universal Robots Corp is shown in Figure 3.

The dimension of the configuration space $\mathcal{C}$ of the robot $\mathcal{R}$ is exactly n, i.e., the number of DOFs in $\mathcal{R}$. Most previous works partition $\mathcal{C}$ into a free subspace $\mathcal{C}_{free}$ and an in-collision subspace $\mathcal{C}_{clsn} = \mathcal{C} \setminus \mathcal{C}_{free}$, satisfying that the robot configuration specified by any point in $\mathcal{C}_{free}$ does not have self-collision with $\mathcal{R}$ or collision with obstacles. Based on the characteristics summarized in Eqs. (1-2), we further decompose $\mathcal{C}_{free}$ and $\mathcal{C}_{clsn}$ accordingly to each DOF of $\mathcal{R}$ as follows.

Throughout this paper, we consider static environment with arbitrary complex obstacles. First, we consider the DOFs in $\mathcal{D}_1 = \{d_1, \cdots, d_{n_1}\}$, where n_1 is the number of DOFs in $\mathcal{D}_1$. Let $\mathcal{C}_1$ be the subspace of $\mathcal{C}$ spanned by the DOFs in $\mathcal{D}_1$. We define the subspace $\mathcal{C}_{1free}$ of $\mathcal{C}_1$ as the collection of all configurations of the component $\mathcal{R}_1$ that do not have self-collision or collision with obstacles. In this way, we partition $\mathcal{C}_1$ into the free subspace $\mathcal{C}_{1free}$ and the in-collision subspace $\mathcal{C}_{1clsn} = \mathcal{C}_1 \setminus \mathcal{C}_{1free}$. We further define an expansion operation $*$ that expands the dimension of $\mathcal{C}_{1free}$ from n_1 to n:

$$\mathcal{C}^*_{1free} = \{\mathcal{C} : \text{coordinates in the first } n_1 \text{ dimensions are restricted in the range of } \mathcal{C}_{1free}\} \quad (3)$$

Similarly, we define

$$\mathcal{C}^*_{1clsn} = \{\mathcal{C} : \text{coordinates in the first } n_1 \text{ dimensions are restricted in the range of } \mathcal{C}_{1clsn}\} \quad (4)$$

Obviously, $\mathcal{C}^*_{1clsn} \subseteq \mathcal{C}_{clsn}$.

The above definition can be easily extended to $\mathcal{C}_i$, $1 < i \leq n_{\mathcal{R}}$, which is the subspace spanned by the DOFs in $\mathcal{D}^i = \cup_{j=1}^{i} \mathcal{D}_j$. We denote the number of DOFs in $\mathcal{D}^i$ as n_i. Any element in $\mathcal{D}^i$ specifies a configuration of the component $\mathcal{R}_i$. We define the subspace $\mathcal{C}_{ifree}$ of $\mathcal{C}_i$ as the collection of all configurations of the component $\mathcal{R}_i$ that do not have self-collision with $\cup_{j=1}^{i} \mathcal{R}_i$ or collision with obstacles. Then we can partition $\mathcal{C}_i$ into the free subspace $\mathcal{C}_{ifree}$ and the in-collision subspace $\mathcal{C}_{iclsn} = \mathcal{C}_i \setminus \mathcal{C}_{ifree}$ as

$$\mathcal{C}^*_{ifree} = \{\mathcal{C} : \text{coordinates in the first } n_i \text{ dimensions are restricted in the range of } \mathcal{C}_{ifree}\} \quad (5)$$

$$\mathcal{C}^*_{iclsn} = \{\mathcal{C} : \text{coordinates in the first } n_i \text{ dimensions are restricted in the range of } \mathcal{C}_{iclsn}\} \quad (6)$$

The decomposed subspaces $\{\mathcal{C}^*_{1free}, \mathcal{C}^*_{2free}, \cdots, \mathcal{C}^*_{n_\mathcal{R} free}\}$ and $\{\mathcal{C}^*_{1clsn}, \mathcal{C}^*_{2clsn}, \cdots, \mathcal{C}^*_{n_\mathcal{R} clsn}\}$ have the following two important properties:

$$\cup_{i=1}^{n_\mathcal{R}} \mathcal{C}^*_{iclsn} = \mathcal{C}_{clsn} \quad (7)$$

$$\cap_{i=1}^{n_\mathcal{R}} \mathcal{C}^*_{ifree} = \mathcal{C}_{free} \quad (8)$$

See Figure 2 for an example.

IV. COMPOSITE CLASSIFIER

In this section, we show that the properties in Eqs. (7-8) can lead to an efficient composite classifier. Our proposed composite classifier is orthogonal to previous machine learning methods that train a single binary classifier to distinguish $\mathcal{C}_{free}$ from $\mathcal{C}_{clsn}$ (e.g., [16], [13], [17], [14]). Let f be an elementary classifier that can be any one in these previous works.

Given a set of samples with ground-truth labels (in-collision or collision-free) in $\mathcal{C}$, we train an elementary classifier f_i for each robotic component $\mathcal{R}_i$ in the subspace $\mathcal{C}_i$, $i = 1, 2, \cdots, n_\mathcal{R}$, i.e.,

$$f_i(x) = \begin{cases} 1 & \text{if } x \in \mathcal{C}_{ifree} \\ 0 & \text{otherwise} \end{cases}, \; x \in \mathcal{C}_i \quad (9)$$

Then our composite classifier F can be defined as

$$F = f_1 \wedge f_2 \wedge \cdots \wedge f_{n_\mathcal{R}} \quad (10)$$

where $\wedge$ is the logistic AND operation, meaning that $F(x) = 1$ if and only if all of its operands are true, i.e., $f_i(x) = 1$ for $i = 1, 2, \cdots, n_\mathcal{R}$, $x \in \mathcal{C}$.

The advantage of the composite classifier F is of three-fold:

- Each elementary classifier in $\{f_1, f_2, \cdots, f_{n_\mathcal{R}}\}$ works in a subspace $\mathcal{C}_i$ of $\mathcal{C}$, in which the boundary between $\mathcal{C}_{ifree}$ and $\mathcal{C}_{iclsn}$ is much simpler than the boundary between $\mathcal{C}_{free}$ and $\mathcal{C}_{clsn}$ in $\mathcal{C}$ (see Figures 2b, 2d, 2f, 2g for an example), and thus the classification accuracy of each elementary classifier (as well as the composite classifier) is much higher than that of a single classifier directly working on $\mathcal{C}$;
- Except for $f_{n_\mathcal{R}}$, all other elementary classifiers work in lower-dimensional subspaces $\mathcal{C}_i$ with simpler class boundaries than that of the full configuration space $\mathcal{C}$ and thus the classification speed is faster than that of a single classifier;
- The elementary classifiers $\{f_1, f_2, \cdots, f_{n_\mathcal{R}}\}$ act as a hierarchical set of filters that filter in-collision samples from the lowest to the highest dimensions, i.e., not all the in-collision samples need to be checked by all the filters. Therefore, the overall classification speed is still faster than a single classifier directly working on $\mathcal{C}$.

Our experimental results in Section V show that averagely the classification accuracy of our composite classifier is improved 10%-20% and 2 times faster than a single classifier directly working on $\mathcal{C}$ in most cases.

To take the full advantage of the composite classifier F, it is worth noting the following implementation details.

Effective DOFs in $\mathcal{R}$. In different application scenarios, not every DOF of the robot is of the same importance. For example, for the UR5 robot shown in Figure 3, if a dexterous hand is attached to the end of component $\mathcal{R}_6$, the rotation of $\mathcal{R}_6$ caused by the 6-th DOF may make the dexterous hand collide with obstacles, and thus the 6-th DOF should be seriously considered. However, if a cylindric platform (like the one for the 3D printing in [21]) is attached to the end of component $\mathcal{R}_6$, the rotation of $\mathcal{R}_6$ caused by the 6-th DOF only changes the orientation of the cylindric platform but not its collision status. Therefore, the 6-th DOF is not effective. In most cases, it is easy to determine the effectiveness of each DOF in $\mathcal{R}$ using a simple input from the user. For non-effective DOFs D_j, we don't need to build the subspace C_j or train the classifier f_j.

Online learning. As aforementioned in Section I, to apply the proposed composite classifier F in a single query of the RRT methods [7], [8], it must have the ability of online learning, i.e., efficiently updating F when more and more samples are obtained during the sampling process. Online updating F equals to online updating its elementary classifiers $\{f_1, f_2, \cdots, f_{n_\mathcal{R}}\}$. For commonly used elementary classifiers, their online learning schemes are available, e.g., online learning of SVM [22], KNN [13], GMM [17], [20] and the Gaussian kernel functions [14].

V. EXPERIMENTS

Using the 6-DOF UR5 robot as the experiment platform, we implement the proposed configuration space decomposition method and the composite classifier F in MATLAB. To train F, we randomly sample the parametric domain of the configuration space with a uniform distribution in a C++ ROS environment, and the label for each sample is specified by an exact collision detector called Flexible Collision Library (FCL) [12]. All the running time reported in this section is recorded in a PC with an Intel i7-8700 CPU (3.20GHz) and 16GB RAM.

TABLE I

THE COMPARISON OF SINGLE-CLASSIFIER (SC) METHODS (SVM [16] AND KNN [13]) AND OUR COMPOSITE-CLASSIFIER METHOD USING SVM AND KNN AS ELEMENTARY CLASSIFIER RESPECTIVELY IN TEST ENVIRONMENTS WITH 1K, 10K AND 100K RANDOM SAMPLES IN THE CONFIGURATION SPACE WHOSE WORKING ENVIRONMENT CONSISTS OF A 6-DOF UR5 ROBOT AND CUBIC OBSTACLES. WE RANDOMLY GENERATE 3 WORKSPACES, EACH WITH 4 OBSTACLES, AND THE QUERY TIME AND ACCURACY ARE AVERAGED RESULTS.

Elementary classifier	Number of samples	Accuracy		Query time (μs)	
		SC	Ours	SC	Ours
SVM	1K	0.736	0.878	23.7	15.2
	10K	0.820	0.946	114.9	45.7
	100K	0.923	0.980	998.9	263.4
KNN	1K	0.707	0.846	3.8	4.5
	10K	0.775	0.921	16.1	16.2
	100K	0.852	0.961	107.2	55.4

A. Classification Accuracy and Time

The existing approximate collision detection methods can be categorized into the kernel and clustering approaches. To prove the universality of our approach, we choose two representative single-classifier methods – SVM [16] (kernel) and KNN [13] – as the baselines to compare with our composite classifier. We denote the composite classifier F that uses SVM or KNN as the elementary classifier as F_{SVM} or F_{KNN}, respectively. We generate 3 workspaces by randomly placing four cubic obstacles around the position of 6-DOF UR5 robot. For each workspace, we randomly sample 1K, 10K and 100K points in the configuration space $\mathcal{C}$ as the training set, respectively. And another 10K points are randomly sampled in $\mathcal{C}$ for test. The averaged classification accuracy and query time among 3 workspaces are summarized in Table I. The following four phenomena are observed from these results.

First, the accuracy of the composite classifier is much higher (improving 10%-20%) than that of the single classifier. This is because our composite classifier F decomposes the configuration space $\mathcal{C}$ into a set of subspaces $\{\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_{n_\mathcal{R}}\}$, in which each subspace $\mathcal{C}_i$ has a simple boundary between $\mathcal{C}_{ifree}$ and $\mathcal{C}_{iclsn}$, and can be approximated with a much lower error by an elementary classifier f_i, when compared to the accuracy of using a single classifier to directly classify $\mathcal{C}_{free}$ and $\mathcal{C}_{clsn}$ in $\mathcal{C}$.

TABLE II

THE IN-COLLISION AND COLLISION-FREE PERCENTAGES (%) OF 10K TESTING SAMPLES ARE REPORTED FOR ALL 3 WORKSPACES. FOR IN-COLLISION SAMPLES, THE DETAILED PERCENTAGES (%) FOR $\mathcal{C}_{1clsn}, \mathcal{C}_{2clsn}, \cdots, \mathcal{C}_{6clsn}$ ARE ALSO REPORTED.

Workspace id	In-collision						Collision free
	$\mathcal{C}_{1clsn}$	$\mathcal{C}_{2clsn}$	$\mathcal{C}_{3clsn}$	$\mathcal{C}_{4clsn}$	$\mathcal{C}_{5clsn}$	$\mathcal{C}_{6clsn}$	
1	16.7	19.4	9.5	2.1	5.8	0.2	46.3
2	40.7	23.5	6.3	0.9	3	0	25.6
3	0	24.3	15.0	2.5	8.2	0.1	49.9

Second, the query time of the composite classifier is faster than that of the single classifier in most cases. This is because although the composite classifier may need to check multiple elementary classifiers, the query time of each elementary classifier is faster and only a few samples need to pass all these elementary classifiers. In addition, the dimension and number of features of each elementary classifiers are also less than that of a single classifier. To further reveal the latter property, we report the in-collision and collision-free percentages of 10K testing random samples in Table II, showing that 40%-60% samples are filtered by the first three elementary classifiers in the subspaces $\{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$ in complex environment. In Table I, the composite KNN classifier is not as fast as a single classifier under a small number of sampling points or in a simple environment. However, as the number of training points increases or in more complex environments, the composite classifier still outperforms the single classifier.

Third, the more sampling points, the higher the accuracy of both the composite and single classifiers would be. However, even with 100K samples, the accuracy of the composite classifier is still 10% higher than that of the single classifier.

TABLE III

ABLATION STUDY ON THE NUMBER OF DOFS. WE ONLY TEST ON THE WORKSPACE 1 FOR SIMPLICITY.

Elementary classifier	Number of DOFs	Accuracy		Query time (μs)	
		SC	Ours	SC	Ours
SVM	4-DOF	0.915	0.972	43.0	22.3
	3-DOF	0.964	0.976	21.8	15.5
	2-DOF	0.993	0.993	19.05	9.51
KNN	4-DOF	0.852	0.955	14.45	16.6
	3-DOF	0.919	0.961	12.1	11.8
	2-DOF	0.975	0.990	7.51	7.89

Fourth, the performance gains increase with the number of DOFs, showing that our method could better handle complex robots compared to a single classifier. Note that even for the 2-DOF robot, our classifier is still more advantageous in accuracy and query time.

TABLE IV

EXPERIMENT ON COMPLEX ENVIRONMENTS WITH A LARGE NUMBER OF SMALL CUBIC OBSTACLES. WE USE 10K SAMPLES FOR TRAINING. THE COMPOSITE CLASSIFIER CONSISTENTLY OUTPERFORMS THE SINGLE CLASSIFIER BY A LARGE MARGIN. WHEN THE NUMBER OF OBSTACLES IS LARGE, THE SINGLE CLASSIFIER CANNOT DISTINGUISH COLLISION OR FREE SAMPLES AND LABELS ALMOST ALL THE TESTING SAMPLES AS IN-COLLISION, WHILE THE COMPOSITE CLASSIFIER WORKS WELL.

Elementary classifier	Num of obstacles	Collision free	Accuracy		Query time (μs)	
			SC	Ours	SC	Ours
SVM	50	38.94	0.789	0.908	74.6	34.4
	100	20.62	0.796	0.905	75.4	26.6
	200	7.43	0.926	0.952	36.1	14.8
KNN	50	38.94	0.744	0.897	15.2	11.9
	100	20.62	0.785	0.884	10.8	8.4
	200	7.43	0.918	0.947	11.9	7.8

TABLE V

THE IN-COLLISION AND COLLISION-FREE PERCENTAGES (%) OF 10K TESTING SAMPLES FOR THE GENERATED COMPLEX ENVIRONMENT. FOR IN-COLLISION SAMPLES, THE DETAILED PERCENTAGES (%) FOR $\mathcal{C}_{1clsn}, \mathcal{C}_{2clsn}, \cdots, \mathcal{C}_{6clsn}$ ARE ALSO REPORTED.

Num of obstacles	In-collision						Collision free
	$\mathcal{C}_{1clsn}$	$\mathcal{C}_{2clsn}$	$\mathcal{C}_{3clsn}$	$\mathcal{C}_{4clsn}$	$\mathcal{C}_{5clsn}$	$\mathcal{C}_{6clsn}$	
50	9.05	27.84	15.7	2.55	5.88	0.04	38.94
100	9.6	44.94	18.74	2.43	3.64	0.03	20.62
200	18.1	56.36	14.56	1.84	1.68	0.03	7.43

To further prove the effectiveness of our method in more complex environments, we generate a new workspace by randomly placing 50, 100 and 200 small cubic obstacles around the position of 6-DOF UR5 robot. And we randomly sample 10K points in the configuration space $\mathcal{C}$ as the training set and randomly sample another 10K points for testing. Same as Table II, we report the in-collision and collision-free percentages of the 10K testing samples of this new generated workspace in Table V. Note that these new-generated cubic obstacles are smaller than those used in Table II, but the much larger number of these small obstacles makes the workspace more complex for collision checking.

The classification accuracy and query time are summarized in Table IV, from which the same conclusions can be drawn as those from Table I. Moreover, when there are a very

TABLE VI

THE SUCCESS RATE (THE FIRST NUMBER IN THE BRACKET) AND AVERAGE GENERATING TIME (THE SECOND NUMBER IN THE BRACKET, IN MILLISECOND) FOR MOTION PLANNING IN 100 RANDOMLY GENERATED PAIRS. EACH PAIR CONSISTS OF TWO COLLISION-FREE POINTS (ONE SOURCE AND ONE TARGET) IN THE CONFIGURATION SPACE. WE TEST IN THE WORKSPACE CONSISTING OF A 6-DOF UR5 ROBOT AND 2, 4, 8 CUBIC OBSTACLES, RESPECTIVELY. WE COMPARE THE BASIC RRT METHOD WITH FCL, AND THE VARIANTS OF RRT METHODS WITH LEARNING-BASED COLLISION CHECKING, INCLUDING FASTRON [14], SVM [16], KNN [13] AND OUR COMPOSITE CLASSIFIERS F_{SVM} AND F_{KNN}.

Obstacle number	Basic RRT with FCL	Variants of RRT with learning-based collision checking				
		Fastron	SVM	F_{SVM}	KNN	F_{KNN}
2	(56%, 850.5)	(49%, 524.4)	(38%, 1329.6)	(73%, 1058.1)	(37%, 1330.0)	(51%, 1130.7)
4	(51%, 880.9)	(36%, 562.7)	(35%, 1592.9)	(69%, 1101.9)	(39%, 1866.3)	(46%, 1216.8)
8	(5%, 1000.1)	(0%, not available)	(2%, 2806.7)	(27%, 2398.8)	(6%, 2090.1)	(14%, 1529.0)

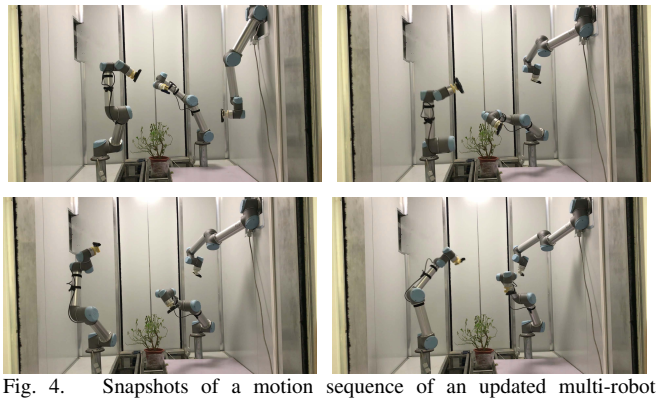

Fig. 4. Snapshots of a motion sequence of an updated multi-robot plant phenotyping system [15] using the learning-based method with our composite classifier. See accompanying demo video for more details.

large number of obstacles, e.g., 200 obstacles in the working environment, the single classifier will lose the ability of classifying the collision-free samples from in-collision ones and predict almost all the samples as in-collision. See the 200 obstacles case in Table IV for example. Note that the single SVM classifier can achieve 92.6% accuracy for 200 obstacles because the single SVM classifier labels all testing samples as in-collision and there are exactly 92.6% in-collision samples in this workspace environment. Even under this setting, the composite classifier still works well, demonstrating its strong capability for complex environments.

B. Motion Planning Efficiency

As aforementioned in Section I, the learning-based collision checking can help improve the efficiency of both single-query and multiple-query sampling-based motion planning. In this section, we use the single-query RRT method as the baseline for comparison.

We use the open motion planning library (OMPL[2]) [23], which provides an optimized implementation of the RRT method. The RRT algorithm uses a biased search to quickly explore the large unsearched space. Using enough time, RRT will eventually build a random space-filling tree and thus find a path between source and target points. For practical usage, we set the parameter *MaxTime* (i.e., maximum planning time used by RRT) to be 3 seconds. Therefore, the faster collision detection, the higher the success rate of path planning.

We set up the basic RRT method implemented by OMPL for the single-query motion planning by using FCL [12] for exact collision detection, which is very time-consuming. For comparison, we replace FCL by four learning-based collision detectors, i.e., Fastron [14], SVM [16], KNN [13] and our composite classifiers F_{SVM} (using SVM as the elementary classifier) and F_{KNN} (using KNN as the elementary classifier). Note that the prediction by trained classifiers only provides approximate collision checking. Once a path is planned by RRT with approximate collision checking, a final exact collision checking by FCL is needed for every sample in the path. For those samples that are in-collision, a local repairing operation proposed in the implementation of [14] is evoked. Therefore, the more accurate the classifier, the fewer samples need to be repaired and the more efficient the motion planning with the learning-based method would be.

As indicated in Section V-A, our proposed composite classifier has much better accuracy and less query time than each single classifier, and then will lead to a more efficient motion planning process. This conclusion is demonstrated by the motion planning results summarized in Table VI. These results are generated in the same working environments consisting of a 6-DOF UR5 robot and 2, 4, 8 cubic obstacles as in Section V-A. Because the query time of FCL method is about 60μs, we use 5000 samples for training classifers. We randomly generate 100 pairs of collision-free source and target points in the configuration space. The motion planning is applied to plan a path between each pair of source and target points. In the limited maximum planning time (3 seconds), not every pair can have a successful motion planning and we define the *success rate* as the ratio $\frac{s}{100}$, where s is the number of pairs that have a successful motion planning and 100 is the number of total pairs. We also define the *average generating time* as the time of generating successful paths averaged on s successful paths. The results in Table VI show that (1) the larger the number of obstacles is, the lower the success rate is, (2) our composite classifier can significantly improve the success rate of single classifiers, (3) RRT with composite classifier F_{SVM} significantly improves the success rate of the basic RRT with FCL, and (4) the average generating time of RRT with composite classifier is much shorter than that of RRT with each individual classifier.

C. Application

We apply the proposed composite classifier in an updated multi-robot plant phenotyping system [15] for improving the efficiency of motion planning. This system was designed to provide fast, precise and noninvasive measurements for robot-assisted high-throughput plant phenotyping. To achieve this goal, this system was equipped with three UR5 robotic arms that can be moved simultaneously in each round of

[2]http://ompl.kavrakilab.org/

phenotyping data acquisition, using the depth camera (Intel RealSense SR-300) mounted at each robotic arm. The configuration space of this system has a very complicated boundary between $\mathcal{C}_{free}$ and $\mathcal{C}_{clsn}$ due to the high possibilities of collision (with plant leaves) and self-collision (among robotic arms). See Figure 1 for a motion planning example.

To satisfy the requirement of high-throughput plant phenotyping, the motion planning has to be fast. Using the state-of-the-art RRT implementation in OMPL on a PC with Intel(R)Xeon(R)Gold 6146 CPU @3.20 GHz, 128 GB RAM and NVIDIA Titan V, the average motion planning time for each round is about 1 second, while using the learning-based method with our composite classifier, the average motion planning time is shorten to 0.692 seconds (only successful cases are counted). Here our method is deployed on this multi-robot system by training separate composite classifier for each robot, i.e. using three composite classifiers to filter the configuration samples during motion planning. We will discuss more sophisticated composite classifier implementation for the multi-robot setup in future work.

D. Discussion

Our system works well under most scenarios and significantly improves the performance of single classifiers. However, it may have a performance degrade when the environment is too simple or the manipulators have overly simple geometry structure. Under these cases, the collision barely happens or happens only for the last few DOFs, thus to distinguish whether a configuration sample is collision or not, our method needs to pass through almost all the decomposed classifiers. For extreme cases, e.g., with no obstacles, the query time of our composite classifier may exceed that of a single classifier by up to 1.5 times.

VI. CONCLUSIONS

In this paper, we propose a simple yet effective configuration space decomposition method based on the characteristics inherent in the DOFs of robot, which leads to a novel composite classifier with a much better classification accuracy than single classifiers. Our method work compatibly with previous method by using them as elementary classifiers. Experimental results on both artificial environments with increasing complexity and a real environment of a multi-robot plant phenotyping system [15] demonstrate that our method can significantly shorten the time of motion planning. Introducing sparse learning or more powerful elementary classifier could further improves the performance, which is an interesting direction for future work. Application in dynamic environment will also be discussed in future work.

REFERENCES

[1] J.-C. Latombe, *Robot Motion Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[2] K. Tang and Y.-J. Liu, "A geometric method for determining intersection relations between a movable convex object and a set of planar polygons," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 636–650, 2004.

[3] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning," *Engineering*, vol. 1, no. 1, pp. 46–57, 2015.

[4] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 2010.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[6] L. E. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[7] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[8] J. J. Kuffner and S. M. Lavalle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 995–1001.

[9] D. Kim, Y. Kwon, and S. Yoon, "Dancing PRM*: Simultaneous planning of sampling and optimization with configuration free space approximation," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 7071–7078.

[10] S. Dai, S. Schaffert, A. Jasour, A. G. Hofmann, and B. C. Williams, "Chance constrained motion planning for high-dimensional robots," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8805–8811.

[11] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[12] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 3859–3866.

[13] J. Pan and D. Manocha, "Fast probabilistic collision checking for sampling-based motion planning using locality-sensitive hashing," *International Journal of Robotics Research*, vol. 35, no. 12, pp. 1477–1496, 2016.

[14] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *CoRR*, vol. abs/1902.08164, 2019.

[15] C. Wu, R. Zeng, J. Pan, C. C. Wang, and Y.-J. Liu, "Plant phenotyping by deep-learning based planner for multi-robots," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3113–3120, 2019.

[16] J. Pan and D. Manocha, "Fast and robust motion planning with noisy data using machine learning," in *Workshop on Robot Learning, in Proc. Int. Conf. on Machine Learning*, 2012.

[17] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 63–69.

[18] I. Garcia, J. D. Martin-Guerrero, E. Soria-Olivas, R. J. Martinez, S. Rueda, and R. Magdalena, "A neural network approach for real-time collision detection," in *IEEE International Conference on Systems, Man and Cybernetics*, 2013.

[19] Y. J. Heo, D. Kim, W. Lee, H. Kim, J. Park, and W. K. Chung, "Collision detection for industrial collaborative robots: A deep learning approach," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 740–746, 2019.

[20] J. Huh, B. Lee, and D. D. Lee, "Adaptive motion planning with high-dimensional mixture models," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 3740–3747.

[21] C. Wu, C. Dai, G. Fang, Y. Liu, and C. C. L. Wang, "Robofdm: A robotic system for support-free fabrication using FDM," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1175–1180.

[22] T. Glasmachers and C. Igel, "Second-order SMO improves SVM online and active learning," *Neural Computation*, vol. 20, no. 2, pp. 374–382, 2008.

[23] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.