Data-Driven Damage Detection and Control Adaptation for an Autonomous Underwater Vehicle

Özer Özkahraman*, Pouria Tajvar*, Dimos V. Dimarogonas, Petter Ögren
KTH Royal Institute of Technology
{ozero, tajvar, dimos, petter}@kth.se

***Abstract*— Underwater robotic exploration missions typically involve traveling long distances without any human contact. The robots that go on such missions risk getting damaged by the unknown environment, accruing great costs and missed opportunities. Thus it is important for the robot to be able to accommodate unknown changes to its dynamics as much as possible and attempt to finish the given mission, or at the very least move itself to a retrievable position.**

In this paper, we show how we can detect physical changes to the robot reliably (79% on real robot data) and then incorporate these changes through adapting the model to the data followed by automated control redesign. We adopt a piecewise-affine (PWA) modelling of the dynamics that is well suited for low data regime learning of the dynamics and provides a structure for computationally efficient control synthesis. We demonstrate the effectiveness of the proposed method on a combination of real robot data and simulated scenarios.

I. Introduction

A vast majority of the Earth is covered with water, mostly unexplored, and some covered with ice. In order to understand our planet better and care for it, we must learn the behavior of its waters. The ice cover further complicates the exploration problem, since surface vehicles are rendered ineffective. Remote underwater operation in these areas is also not possible due to the lack of bandwidth and range of underwater communication methods. Such environments are where Autonomous Underwater Vehicles(AUVs) can be utilized to great effect.

During long-range under-ice missions, a vehicle might go through physical changes, such as bio-fouling, which is when organisms grow on the AUV and similar natural effects [1], and parts being damaged. Such changes are hard to foresee and account for, and fall into the unknown-unknowns category of knowledge about the mission. This difficulty mostly comes from the fact that these effects are dependent on the chemical and physical properties of the waters the vehicle will travel through, which are unknown before the mission. The cost of cancelling an under-ice long-range mission prematurely due to a damaged part can include losing the vehicle, the data collected so far, and possibly years of work time and preparation. Such a high cost of failure necessitates a method in which the vehicle will attempt to detect the presence of and handle physical changes and continue the mission, or at the very least return to a position where it can be retrieved. On the other hand, limited

*equal contribution

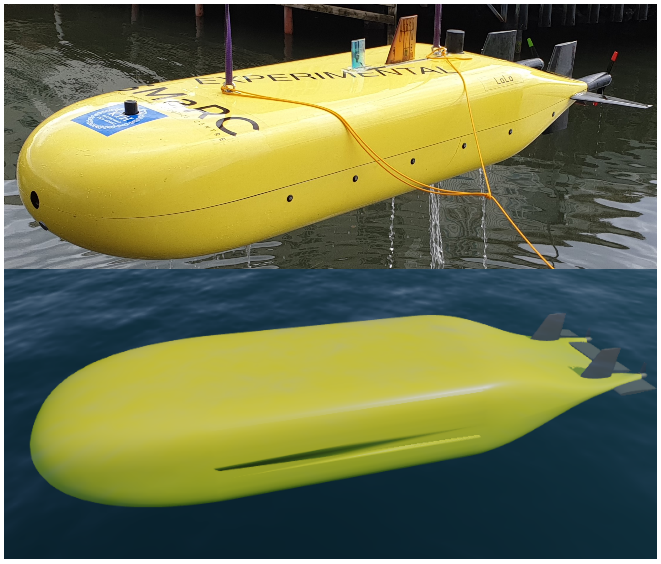

Fig. 1: The AUV LoLo [3] that is used in this paper. Top: The real vehicle. Bottom: Simulation in Stonefish

computational power and scarce data availability prohibits expensive global model learning approaches. Therefore it is desirable to be able to adjust the dynamics model locally in the region of the robot's state-input space where new data is collected and adapt the control policy accordingly.

We identify the following problems to reach reliable autonomous adaptation to dynamics changes:

Problem 1: The vehicle must be able to detect physical changes. The vehicle may or may not undergo physical changes during a long mission. The exact nature of these changes is unknown before the mission starts, thus the vehicle must detect changes from its nominal behavior. This problem is categorized as an open-set recognition problem [2] where the classifier is tasked with recognizing which class a given input is, in addition to recognizing out of distribution (OOD) samples.

Problem 2: The vehicle should attempt to accommodate physical changes. Given that a physical change has been detected, the vehicle must autonomously be able to attempt to handle these changes by incorporating them into its motion plan. If the changes are too extreme to complete the mission, the vehicle should attempt to reach a state where it can be retrieved.

In order to solve the above problems, we propose an extension over CBF-BTs [4] that checks if the nominal model used is still valid, and if it is not, switches to a data-driven controller that is synthesised during the mission. The detection needed for Problem 1 is done through Scalable Variational Gaussian Processes (SVGPs) [5] trained on augmented simulation data. The data from the simulated vehicle is augmented using the relatively small amount of nominal-condition data from the real vehicle. To evaluate the SVGP method, we compare it to a nearest-neighbor based method.

To handle Problem 2, we propose switching to a piecewise-affine (PWA) modelling of the dynamics for control design as long as the nominal model is invalid. This enables us to tune the model locally when non-conforming (abnormal) data is detected in part of the state-input space. We use a sampling based approach to adapt the control policy to the change in the dynamics, and possibly still complete the mission. It also allows for infeasibility checks, i.e. when the cost becomes infinite, allowing for the application of more drastic emergency measures when the mission is impossible to complete.

The contribution of this paper is twofold; we first incorporate vehicle control inputs into OOD detection for vehicle dynamics change identification and we then develop a sampling-based control for PWA systems to efficiently adapt to the altered vehicle dynamics. The implementation of our proposed method can be found online[1].

II. Related work

Open set recognition (OSR): The open set recognition problem has been studied using a wide range of methods [6]–[8] where both traditional machine learning methods and deep neural network based methods are modified to recognize OOD samples. Particularly, our work in this paper is most similar to [9] where a nearest-non-outlier method is employed to classify a given sample. However, the aforementioned methods consider the samples without their semantics and most of them are based on the image classification domain. In contrast, we incorporate the known intentions of the vehicle, in the form of control inputs, during classification. In the robotics domain, an adjacent problem called novelty detection has been studied [10], [11]. However, these works consider the detection of new structures in the data that is extrinsic to the perceiving vehicle, usually in the context of infrastructure monitoring.

Sampling based control synthesis: Early works in sampling based control synthesis such as [12]–[14] present an alternative to hierarchical motion-planning and control for problems where the system dynamics is highly constrained, e.g. non-holonomic constraints, sensing limitations or restrictive input bounds. In such problems the constraints should be considered already during motion planning; in the case of RRT algorithm the constraints are imposed typically as boundary value problems in the *steer* and *near* functions. Recent works such as [15], [16] have provided methods to relax the reliance on solving boundary value problems in earlier works as well as improving the scalability. Our work is closest to [12], [17] in the sense that we also utilize reachable set computation to guide the sampling, we however adopt a Zonotope based reachable set computation, to enable fast adaptation of *near* and *steer* functions with changing dynamics while explicitly incorporating dynamic uncertainty in the planned timed trajectory. The Zonotope based reachable set computation and control approach was initially proposed in [18] albeit for symbolic control rather than sampling based control.

[1]https://github.com/KKalem/Data-Driven-Damage-Detection-and-Control-Adaptation-for-an-Autonomous-Underwater-Vehicle

III. Background and Preliminaries

In this section, we give a brief working overview of the methods used in Section V.

Scalable Variational Gaussian Processes (SVGPs): A Gaussian Process (GP) is a distribution over functions within a continuous domain, such as time or space [19]. For the purposes of this paper, we can view GPs as a black-box regression model that given a list of training points X and a testing point x', produces a Normal distribution $N(\mu, \sigma^2)$ as its prediction. Given N training points, the GP will have time complexity $O(N^3)$, which disallows its use with large training sets.

SVGPs are an extension on the GP formulation that introduces a set of *inducing points*. The number of inducing points is a hyper-parameter and is usually orders of magnitude smaller than the training set itself. The usage of these inducing points reduces the training time considerably, and allows SVGPs to be used with large datasets, such as velocity data from an AUV. See [5] for a more detailed description of SVGPs.

Behavior Trees: A BT is a switching control structure where a so-called *tick* is sent from the root to the leaves at some frequency. The control flow is determined by the inner nodes and the return values of their children. The inner nodes are one of $\{Sequence, Fallback\}$, arrows and question marks in Figure 5 respectively. The leaf nodes are one of $\{Action, Condition\}$, rectangles and rounded-rectangles in Figure 5 respectively. Each node can return one of $\{Success, Failure, Running\}$, shown as green, red and blue respectively in Figure 5. Success indicates that the node has completed its action successfully or that the condition holds and vice-versa for Failure. Running indicates that the node will take more than one tick's time to complete. The Sequence node ticks its children in order (left to right in Figure 5)as long as they keep returning Success, if all of them return Success, then the Sequence return Success, if one returns Failure then Sequence also returns Failure. The Fallback node does the same, for the opposite cases. Both inner nodes will return Running if a child returns Running. At every tick, the return state of every node is reset, keeping the BT reactive to changes. For a more detailed view of BTs, see [20].

The CBF-BTs [4] is a method that combines Control Barrier Functions(CBFs) and BTs into a model that can handle concurrent and sequential goals. This structure allows for graceful degradation of performance. However, the CBFs require a nominal model of the vehicle to be valid.

k-Dimensional Trees(KD-trees): KD-trees are a space-partitioning structure [21], that allows for fast (average $O(logn)$ time for n points in the tree) nearest-neighbor lookups. In this paper, we use KD-trees to implement a nearest-non-outlier OOD detector as a baseline method.

Zonotopes: A *Zonotope* is a convex, centrally symmetric set; Zonotope representation of sets has been used as an efficient tool in reachability analysis and control particularly because of their compact representation and the fact that the reachable set from a zonotope in the state space of a linear system, remains a Zonotope [22], [23]. We use $Z(\mu, G) \subset \mathbb{R}^n$, where $\mu \in \mathbb{R}^n$ and $G \in \mathbb{R}^{m\times n}$, to represent the following Zonotope set:

$$Z(\mu, G) = \{x \in \mathcal{X} \mid x = \mu + G\omega,\ \omega \in [-1, 1]^n\}, \qquad (1)$$

where $[-1, 1]^n \subset \mathbb{R}^n$ is the Cartesian product of n closed $[-1, 1]$ intervals. We refer to μ as the center of the zonotope and G as the set of generators. $Z.\mu$ and $Z.G$ are used to refer to the corresponding parameters of the zonotope Z.

The following zonotope operations are used in this paper:

$$\begin{aligned} &A \times Z, \text{ where} && (A \times Z).\mu = AZ.\mu, \\ & && (A \times Z).G = AZ.G; \\ &Z \oplus Z', \text{ where} && (Z \oplus Z').\mu = Z.\mu + Z'.\mu, \\ & && (Z \oplus Z').G = [Z.G|Z'.G], \end{aligned}$$

where $Z \oplus Z'$ is the *Minkowski sum* of the two sets and $[Z.G|Z'.G]$ denotes the concatenation of the two matrices along their columns.

IV. Problem Formulation

In this section we formalize the dynamics change detection problem (Problem 1) and the adaptation to changes problem (Problem 2).

A. Dynamics Change Detection

Let $x^+ = f^n(x, u)$ be the discrete-time nominal dynamic model of a vehicle, $x(t) = [p(t), v(t)]^T$ be the instantaneous state where $p(t)$ is the position and $v(t)$ is the velocity at time t. We define a maneuver m_i as a control input $u_i \in \mathcal{U}$ applied for some fixed duration T, starting with initial velocity $v(t_0) = 0$. Let $v_{m_i}(t)$ be the velocity when maneuver m_i is being applied. We aim to find N maneuvers such that $\arg\max_{[m_1 \dots m_N]} \min_{(i,j),\ i\neq j} \Sigma_{t=0}^{T} (v_{m_i}(t) - v_{m_j}(t))$. In other words, we identify maneuvers such that the velocity of the vehicle is as different from any other maneuvers velocity as possible. If the vehicle dynamics change, the velocities of the maneuvers might also change, thus the goal is to detect the change in velocities of these maneuvers.

Let $\mathcal{I} = \{i | i \in \mathbb{N}^+, i \leq N\}$ be the set of identifiers for N maneuvers, with $i = 0$ indicating that a maneuver is OOD and $\mathcal{V}$ be the set of possible velocities. Let $C(i, v(t)) : \mathcal{I} \times \mathcal{V} \longrightarrow \mathcal{I} \cup \{0\}$ be a function that returns a predicted maneuver identifier $\hat{i}$ given an intended maneuver identifier i and velocity $v(t)$. Let $\mathcal{V}_{ID}$ be the set of velocities for all maneuvers under nominal conditions and $\mathcal{V}_{OOD} = \mathcal{V} \setminus \mathcal{V}_{ID}$ be the set of velocities under abnormal conditions. The aim of dynamics change detection is then defined as constructing a classifier C that can maximize the probability of correctly classifying ID and OOD samples, $P(\hat{i} = i | i \in \mathcal{I}, v(t) \in \mathcal{V}_{ID})$ and $P(\hat{i} = 0 | v(t) \in \mathcal{V}_{OOD})$.

B. Data-driven control design

We assume to be given a data set D in which every data point $d \in D$ is a tuple $< x_d, u_d, x_d^+ >$ collected from the actual underlying dynamics of the vehicle $x^+ = f(x, u, w)$ where $w \in W$ corresponds to a bounded exogenous disturbance. Using D, we wish to construct a model of the dynamics $x^+ = \hat{f}(x, u, \hat{w})$ with $\hat{w} \in \hat{W}$ that is *conservative*, i.e.

$$\forall x, u, w\, \exists \hat{w} \in \hat{W} \text{ s.t. } f(x, u, w) = \hat{f}(x, u, \hat{w}). \qquad (2)$$

We aim to design a controller for the data-driven model $\hat{f}$ that can replicate the desired maneuvers from the nominal model f^n. We note that if D is collected entirely from the unaltered vehicle dynamics, we expect the undisturbed data-driven model $\hat{f}(x, u, \mathbf{0})$ to resemble $f^n(x, u)$.

V. Approach

The main parts of our proposed approach are as follows:

1) Augment simulation data with real data.
2) Using the augmented simulation data, train a classifier to detect changes to dynamics.
3) Use a BT to switch between the nominal-controller subtree and the data-driven subtree.
4) Verify if the mission is feasible given the changed dynamics.

A. Simulation and Data Augmentation

We simulate the LoLo AUV [3] in the Stonefish simulator [24] (Figure 1). This allows us to collect dynamics data for both nominal and abnormal conditions, without risking the real vehicle. In order help bridge the simulation gap, we augment the simulated data with data collected from the real vehicle.

In order to transfer the biases and noise models of the real vehicle to the simulated one, we make use of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [25], Principal Component Analysis (PCA) [26] and Gaussian Mixture Models (GMM) [27]. See Figure 2 for the system diagram.

Following the definitions in Section IV-A, we identify $N = 9$ maneuvers for LoLo, corresponding to two extreme positions and one neutral position for the rudders and the elevator. For all 9 maneuvers, we keep the the two thrusters at maximum RPM, since this maximizes the effect of the rudders and the elevator, further separating the velocities of each maneuver. See Figure 3 for a plot of these velocities.

For each maneuver, DBSCAN is used to identify any outliers and clusters in the real data. The distance parameter of DBSCAN is set to be the median acceleration. This

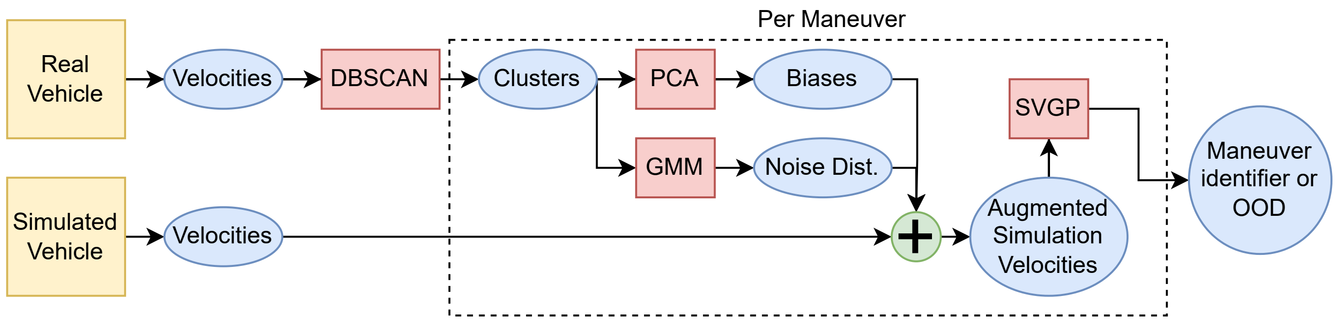

Fig. 2: The process for simulation data augmentation. The algorithm runs throughout the mission.

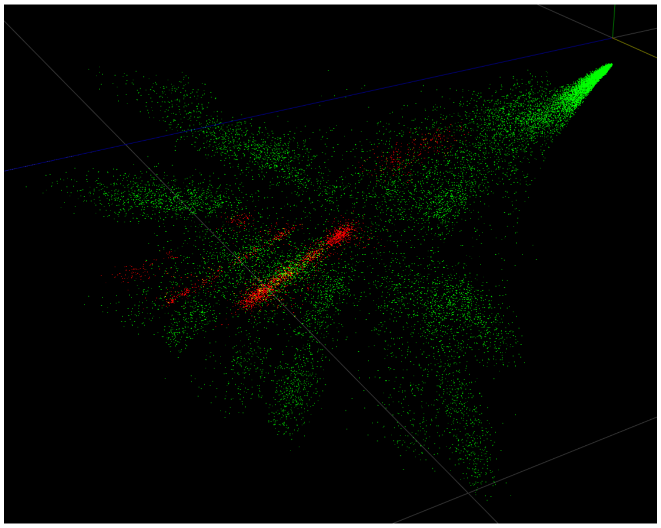

Fig. 3: The dataset of real vehicle velocities (red) and augmented simulated vehicle velocities (green). The maneuvers are not shown separately.

effectively filters out any outliers. The clustered data is then used in PCA, to identify its scale and bias, and in GMM, to identify its noise distribution. The number of clusters in a maneuver is used as the number of components hyper-parameter in GMM. The scale, bias and noise distribution of each maneuver is then applied to the corresponding simulated data to generate the augmented simulation velocities. The augmented simulation data now has the same biases and noise models as the real vehicle.

B. Maneuver Classification

We train and test two different classification methods based on SVGPs [5] and KD-trees. [21]. The KD-Tree method serves as a baseline to compare the proposed SVGP based method against.

For each maneuver, one SVGP is trained on the augmented simulated velocities. Since we only have access to positive samples at training time, and negative samples are considered to be unknown unknowns, we further augment the dataset with uniformly random negative samples.

Remark 1: It is possible to include known negative samples during training as well, but due to the unknown unknown nature of physical faults, doing so risks wrongfully biasing the classifier i.e. shifting the distribution of negative sample classification.

To make predictions, the SVGP that corresponds to the intended maneuver is queried for a prediction with the current velocity. If the confidence of the prediction is above a certain limit, a hyper-parameter, it is accepted as in distribution (ID) for the maneuver, otherwise it is labeled as OOD.

Similarly, a KD-Tree is constructed on the same training data as SVGPs. For predictions, the KD-Tree is queried for all neighbors within some distance, a hyper-parameter, and then a distance-weighted vote is carried out for the maneuver selection. If there were no neighbors within the queried ball, then the sample is labeled OOD.

C. Data-driven piecewise-affine modelling

In this section we present our approach to construction of the model of the vehicle dynamics $\hat{f}$ that is ultimately used for control synthesis. We recall that we have selected PWA structure for $\hat{f}$ because of its modularity so that when OOD data is detected we can adapt the learned model only in the parts of PWA corresponding to that data. We assume to have a given partitioning of the state-input space $\mathcal{X} \times \mathcal{U}$ into a set of $I \in \mathbb{N}$ modes $M_i : X_i \times U_i$ such that $\cup_{i=1}^{\mathbb{N}} M_i = \mathcal{X} \times \mathcal{U}$. Automated partitioning of the state-input space into modes for PWA identification is possible through mixed-integer linear programming [28] or clustering [29] among other approaches but is outside the scope of this paper. The initial partitioning into modes can also be suggested by an expert based on the nominal model. In section VI (Fig. 7) we show how such a partitioning is designed for the AUV based on the collected data, i.e. three modes are assigned to each maneuver.

The dynamics in each mode is affine, i.e. in the form $x^+ = Ax + Bu + c + \hat{w}$ where c is a constant vector. To identify the parameters A, B, and c in the mode M_i we use the standard least-square linear regression of the points pertaining to that mode, i.e. $D_i = \{d \in D | (x_d, u_d) \in X_i \times U_i\}$. We now need to compute $\hat{W}$ in to satisfy the approximation requirement (2). The model uncertainty is a result of the linearization error in mode i W_{lin} as well as the uncertainty W of the actual underlying model f. The linearization error can be computed as maximum error observed between the linear model and the data points:

$W_{lin} = max_{d \in D_i} |x_d^+ - (Ax_d + Bu_d + c)|$. W is not known however; since we have a GP representation of the data points we can evaluate the highest standard deviation $\overline{\sigma}$ for the whole data set as an approximation of the underlying uncertainty. We can then write the model error bound as $\hat{W} = W_{lin} + s\overline{\sigma}$ where s is a design variable to choose the probabilistic confidence in the conservativeness of the bound, e.g. with $s = 3$ the given bounds are conservative with a 99.73% probability [30].

D. Piecewise affine RRT

In this section we present an adaptation of the RRT algorithm for the identified PWA dynamics model $\hat{f}$ with bounded disturbance $\hat{W}$. We initially explain the requirements on the identified model for feasibility of the control and then present a method to adapt the standard *steer* and *near* functions for affine dynamics.

The k-step extension of the dynamics $x^+ = Ax + Bu + c + w$ can be written as follows:

$$\begin{aligned} x^{k+} = A^k x + [B, AB, ..., A^{k-1}B][u^{(k-1)+}, ..., u] \\ + [\mathbf{1}, A, ..., A^{k-1}][c, ..., c] \\ + [\mathbf{1}, A, ..., A^{k-1}][\hat{w}^{(k-1)+}, ..., \hat{w}]. \end{aligned} \quad (3)$$

For simplicity we denote (3) hereafter as $x^{k+} = \mathbf{A}x + \mathbf{B}\mathbf{u} + \mathbf{c} + \hat{\mathbf{w}}$ with $\mathbf{A} = A^k$, $\mathbf{B} = [B, AB, ..., A^{k-1}B]$, ...

Remark 2: If the (A, B) pair is controllable, (3) is guaranteed to be fully actuated for any $k > n$, as $\mathbf{B}$ is full row rank by definition and therefore the k-step dynamics (3) is fully actuated.

This imposes the first required assumption for the correctness of PWA-RRT algorithm:

Assumption 1: In every mode M_i the (A, B) pair is controllable.

From Remark. 2 we know that the $\mathbf{B}$ is full-row rank and as a result $u = \mathbf{B}^{\#}v$ where $\mathbf{B}^{\#}$ is it's Moore-Penrose pseudo-inverse, yields the least-squares solution to the problem of steering the state by vector v. We use this property to define the near, nearest, and steer functions for the k-step affine dynamics (3) in Algorithm 1. The AffineSteer function returns the input value u_{steer} required to steer the system from x^0 to x^1 under the given fully actuated affine dynamics; u_{steer} may not necessarily be inside the input set U. The AffineNear$^+$ returns a subset X^{near} of the discrete set of states $X^d \subset \mathcal{X}$ that can reach the state x with inputs that are contained in the input set U; we can similarly define the function AffineNear$^-$ to denote the states that can be reached *from* the state x. The AffineNearest$^+$ returns the state $x' \in X^d$ that can reach the state x with the least normalized effort. The normalized effort of input u is computed by expressing u as the least squares combination of the input space zonotope generators $U.G$ using it's Moore-Penrose pseudo-inverse.

Using the adapted required functions for affine dynamics, we present the PWA-RRT* in Algorithm 2. We have implemented the tree construction and tree rewiring as independent functions that can be called separately to enable an agile tree adaptation to changes in the dynamics or the cost function where it suffices to only rewire an existing tree. Each node in the tree is a 3-tuple of <state, parent state, input> (Line 2). In the construction of the tree, starting from the initial state x^s as the tree root (Line 3), we expand the tree by randomly sampling from state and input spaces (Lines 5-6), selecting nearest nodes based on the affine distance metric defined (Lines 7-8) and extending them using random inputs (Lines 9-11). For the rewiring of the tree, at each iteration a node from the constructed tree is randomly selected (Line 16) and its reachable neighboring nodes are found (Line 17-18). The cost function accumulates the costs from a node to it's parent until it reaches the root node. A candidate child is rewired to the node if it reduces its cost (Line 19-25).

Algorithm 1: Affine adaptation of RRT functions

```
1  Function AffineSteer(A, B, c, $x^0$, $x^1$):
2      $x^{aut} \leftarrow \mathbf{A}x^0 + \mathbf{c}$
3      $u_{steer} \leftarrow \mathbf{B}^{\#}(x^1 - x^{aut})$
4      return $u_{steer}$
5
6  Function AffineNear$^+$(A, B, c, $U$, $X^d$, $x$):
7      $X^{near} \leftarrow \emptyset$
8      for $x' \in X^d$ do
9          $x^{aut} \leftarrow \mathbf{A}x' + \mathbf{c}$
10         $u_{steer} \leftarrow \mathbf{B}^{\#}(x - x^{aut})$
11         if $u_{steer} \in U$ then
12             $X^{near} \leftarrow X^{near} \cup \{x'\}$
13     return $X^{near}$
14
15 Function AffineNearest$^+$(A, B, c, $U$, $X^d$, $x$):
16     $x^{nearest} \leftarrow arg\,min_{x' \in X^d} U.G^{\#}\mathbf{B}^{\#}(x - \mathbf{A}x' + \mathbf{c})$
17     return $x^{nearest}$
```

In case of a dynamics change within an affine mode M_i of the dynamics, the parents of the nodes pertaining to that mode will be invalidated and as a result the cost of those nodes becomes infinite. Those nodes can be assigned new parents by only calling the rewiring function in Algorithm 2 without having to re-construct the tree. In case that the tree becomes disjoint, the mission will be considered infeasible.

So far, we have considered the PWA model without disturbance in the PWA-RRT algorithm. to ensure that the controller is able to keep the state uncertainty bounded over time, we must be able to design a feedback controller that counteracts the model disturbance $\hat{W}_i$. The *reachable set* of the k-step dynamics (3) from a state in an affine mode i can be written as the following Minkowski sum of Zonotopes:

$$R(x) \in \{\mathbf{A}x + c_i\} \oplus \mathbf{B}U_i \oplus \hat{\mathbf{W}}, \quad (4)$$

where $\mathbf{A}x + c_i$ is the autonomous evolution of the state, $\mathbf{B}U_i$ is the input reachable set and $\hat{\mathbf{W}}$ is the disturbance reachable set. Disturbance $\hat{w} \in \hat{W}$ can only be measure in the subsequent time step, as a result, a stabilizing input

Algorithm 2: PWA-RRT*

```
1  Function ConstructTree(PWA model: $\hat{f}$, initial state: $x^s$, $\mathcal{X}$, $\mathcal{U}$, iterations: $n_{it}$):
2      $root \leftarrow < x^s, None, None >$
3      $Nodes \leftarrow \{root\}$
4      for $0 \le it \le n_{it}$ do
5          $x^{rand} \leftarrow$ rand($\mathcal{X}$)
6          $u^{rand} \leftarrow$ rand($\mathcal{X}$)
7          $< A, B, c, X, U > \leftarrow$ getMode($\hat{f}, x^{rand}, u^{rand}$)
8          $x^{near} \leftarrow$ AffineNearest$^+(A, B, c, U, Nodes.x, x^{rand})$
9          $< A, B, c, X, U > \leftarrow$ getMode($\hat{f}, x^{rand}, u^{near}$)
10         $x^{new} \leftarrow Ax^{near} + Bu + c$
11         add $< x^{new}, x^{near}, u^{rand} >$ to $Nodes$
12     return $Nodes$
13
14 Function RewireTree(PWA model: $\hat{f}$, $Nodes$, iterations: $n_{it}$):
15     for $0 \le it \le n_{it}$ do
16         $node \leftarrow$ rand($Nodes$)
17         $< A, B, c, X, U > \leftarrow Mode(\hat{f}, node.x, .)$
18         $X^{candidates} \leftarrow$ AffineNear$^-(A, B, c, U, Nodes.x, node.x)$
19         for $x \in X^{candidates}$ do
20             $u^{new} \leftarrow$ AffineSteer($A, B, c, node.x, x$)
21             currentCost $\leftarrow$ cost($x, x^{parent}, u$)
22             rewiredCost $\leftarrow$ cost($x, node.x, u^{new}$)
23             if rewiredCost $<$ currentcost then
24                 remove $< x, x^{parent}, u >$ from $Nodes$
25                 add $< x, node.x, u^{new} >$ to $Nodes$
```

should be able to cancel $\mathbf{A}\hat{w}$, resulting in the second required assumption for the correctness of PWA-RRT algorithm:

Assumption 2: In every mode there exists $0 \le \alpha < 1$ such that $\mathbf{A}\hat{\mathbf{W}} \subset \alpha \mathbf{B} U_i$. Intuitively this means that a portion α of the input space should be able to counteract the disturbance. This part will be allocated as the feedback stabilizing controller ($\mathbf{B}^{\#}e$ where e is the difference between the actual state and PWA prediction) and will not be used in the feedforward PWA-RRT algorithm.

In Fig. 4 we have shown how addition of this feedback term can ensure that the actual trajectory from the underlying nonlinear system will remain within the error bound $\hat{W}$ of the PWA model as time goes on.

E. Behavior Tree

A BT is used to coordinate the switching behavior needed. The CBF-BT is encapsulated with a pre-condition of the vehicle nominal model being unchanged, which prevents it from running if the dynamics of the vehicle has changed. As a fallback to such an event, the data-driven controller (shown

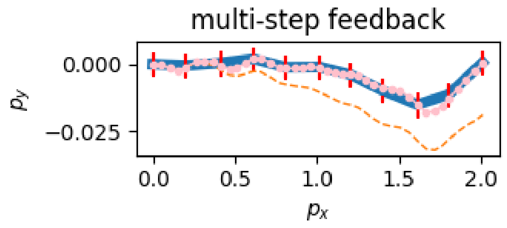

Fig. 4: The solid blue line is a trajectory planned based on the PWA model (projected on x-y plane). The dashed orange line is the trajectory followed by the actual non-holonomic system when the inputs from the PWA model were applied in an open-loop fashion. The dotted pink link is when a 4-step feedback controller is utilized on the actual system to keep the state within the error bound of the PWA model. The red lines appearing every 4 time steps indicate this error bound.

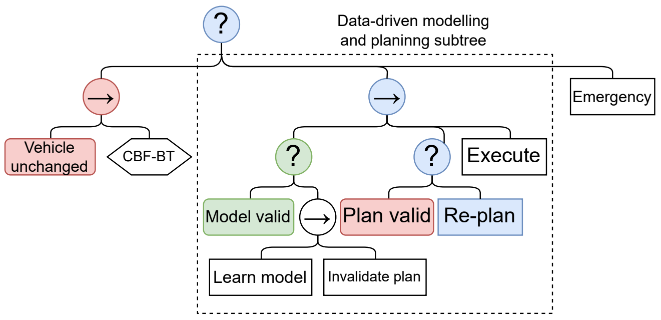

Fig. 5: The BT module used to switch between the nominal-controller subtree (CBF-BT shown as a single node for brevity) and the data-driven subtree. Red, green and blue colors indicate a return of failure, success and running respectively at the current tick. In this example, the vehicle dynamics has changed, the learned model is still valid but the plan is not, thus the tree is re-planning.

inside dotted lines in Figure 5) is placed as an alternative to the nominal controller. This subtree first checks if the current learned model is still valid, and re-learns a model if it is not, followed by using said model to produce a motion plan. Finally, if both the model is valid and there is a plan, the plan is executed. For the case that the vehicle has diverged from its nominal model and attempted to adapt to this change and failed, a drastic emergency action is placed as the final alternative. Usually for AUVs, such an emergency action is dropping of a weight that makes the vehicle buoyant, rising to the surface for recovery.

VI. Results

Physical Change Detection: We compare the nearest-neighbor based method with the proposed SVGP based method. Both methods were trained on the same training dataset that contains all 9 maneuvers executed 10 times each, with the vehicle in nominal conditions. Each maneuver was executed for $T = 3$ minutes, followed by a wait for 0 velocity, at neutral buoyancy. The simulation data was then augmented with the real vehicles data as explained in subsection V-A to produce the training set. In addition to the training set, we generated 5 different validation sets:

- Each maneuver done once, vehicle in nominal condition (validation).
- Both the elevator and the rudders are limited to half of their nominal limits (damaged).
- The vehicle is over-weight, corresponding to a damaged buoyancy system or accumulated bio-mass (over-weight).
- The rudders are stuck in one direction (rudders stuck).
- One thruster at low RPM.

These validation sets correspond to some of the foreseeable types of changes the vehicle might undergo during a mission and have balanced classes. Since each method has one hyper-parameter that defines its "strictness", confidence score for SVGPs and distance for KDTree, we performed a linear search to tune them using the validation sets. As can be seen in Figure 6, the SVGP method outperforms the KDTree method significantly. Note that since there are 9 maneuvers and 1 extra class for OOD, the random chance probability of correct classification is 0.1, which both methods surpass for all sets at particular hyper-parameters. We observe that the SVGP method, while better than the KDTree, also has different accuracy results for different kinds of changes. For example, it is better at identifying when both the elevator and rudders are damaged compared to when a single thruster is at low RPM. Such differences in accuracy are explained by how drastic the change in behavior is with the different damage types. Moving slower due to a partially working thruster is only a slight change from the norm compared to not being able to turn to one side, thus the classifier needs to be a lot more strict to differentiate these more subtle changes. Given there is overlap between the different maneuvers, especially at low velocities due to the dynamics of the vehicle, such accuracy differences are inevitable. At the same time, the higher threshold leads to more false-negatives in the validation data, the model identifies noisy velocities as OOD.

Using the results of the validation sets, we pick 0.9 (chosen as the point where nominal validation accuracy and abnormal validation accuracies overlap) as the hyper-parameter of the proposed SVGP method and test real vehicles data with known and balanced maneuver labels. In the end, we achieve 79% accuracy on the real data, showing that our proposed method is indeed effective on the real vehicle. In comparison, the KDTree method with hyper-parameter 0.007 (chosen as the point where all validation accuracies are above the chance threshold at the same time) achieves 6% accuracy.

Control Adaptation:

As can be seen in left side of Figure 8, when the AUVs rudders are damaged such that the turning radius when turning left is larger than the nominal behavior, the proposed method can instead plan right-turns to reach the desired target. When when the left-turn has enough space, left-turns are still utilized in this case. As an extreme example, on the right side of Figure 8, left-turns are disabled entirely. Our proposed method plans a trajectory that only uses right-turns in this case, and still finishes the given mission.

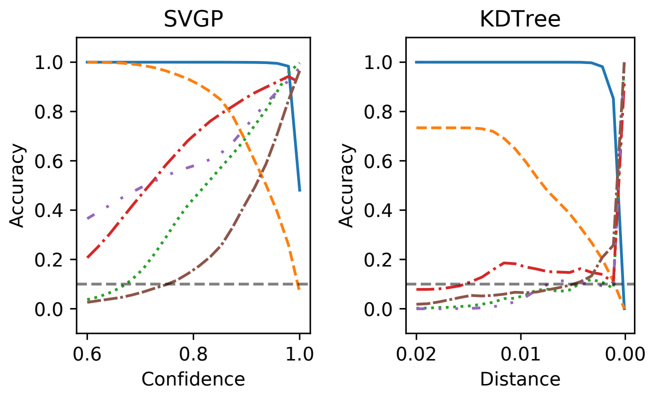

Fig. 6: The accuracy of the two classifiers on different datasets and thresholds. Solid blue: training, dashed orange: validation, dash-dot red: elevator and rudders damaged, dotted green: over-weight, dash-dot-dot purple: rudders stuck, dash-dash-dot brown: one thruster at low RPM. Random guess is shown for 9+1 classes as a grey dashed line. Note that the x-axis is flipped for the right figure to keep the graphs semantically parallel (moving "right" makes the method more strict).

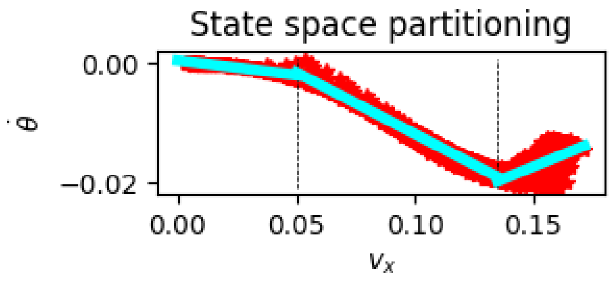

Fig. 7: Turning rate (Radians per second) as a function of linear velocity (meters per second) for fixed rudder angles. An example of partitioning suggested by the expert based on the nonlinearity observed in the data and the subsequent PWA modelling using linear regression.

VII. Conclusions

In this paper we have identified a need for a data-driven method to detect and accommodate physical changes in a robotic system. We have proposed an SVGP based classifier that can detect such changes in the system, and have shown that it also works with high accuracy on real robot data.

Future work includes the identification of the reason for the change in dynamics to further aid in accommodation and the relaxation of our assumptions in this work. The key assumptions that can be difficult to fulfill in certain underwater applications are accurate state estimation, controllability of local linearizations, and known bounds for the model error. We believe that set-based state representations

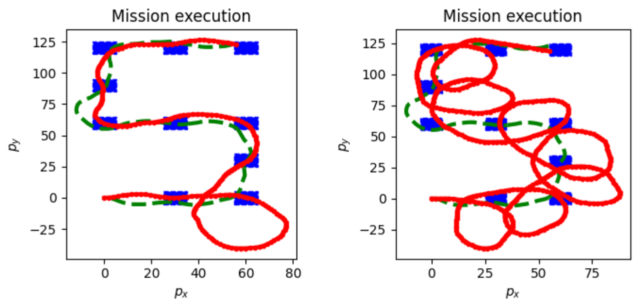

Fig. 8: The simulated AUV mission is to visit the blue regions in a specified order (all units are in meters). The green dashed trajectory is planned using maneuvers based on the nominal dynamics. The red solid line shows the adapted plan when the left turning maneuver dynamics is altered due to a failure in the rudder resulting in higher turning radius (left) and left-turns being disabled entirely (right).

can offer relaxations to the first two assumption while models with unknown error bounds may necessitate probabilistic approaches.

Our current change detection approach works on individual time instants, we will further investigate looking at the dynamics over a period of time which might improve the accuracy possibly at the cost of reactivity.

References

[1] Clinton D Haldeman, David K Aragon, Travis Miles, Scott M Glenn, and Antonio G Ramos. Lessening biofouling on long-duration auv flights: Behavior modifications and lessons learned. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8. IEEE, 2016.

[2] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *CoRR*, abs/2110.11334, 2021.

[3] Clemens Deutsch, Lázaro Moratelli, Sebastian Thuné, Jakob Kuttenkeuler, and Filip Söderling. Design of an auv research platform for demonstration of novel technologies. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–8, 2018.

[4] Özer Özkahraman and Petter Ögren. Combining control barrier functions and behavior trees for multi-agent underwater coverage missions. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5275–5282. IEEE, 2020.

[5] James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Artificial Intelligence and Statistics*, pages 351–360. PMLR, 2015.

[6] Steve Cruz, Cora Coleman, Ethan M Rudd, and Terrance E Boult. Open set intrusion recognition for fine-grained attack categorization. In *2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–6. IEEE, 2017.

[7] Matthew D Scherreik and Brian D Rigling. Open set recognition for automatic target classification with rejection. *IEEE Transactions on Aerospace and Electronic Systems*, 52(2):632–642, 2016.

[8] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[9] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[10] Stephen Marsland, Ulrich Nehmzow, and Jonathan Shapiro. On-line novelty detection for autonomous mobile robots. *Robotics and Autonomous Systems*, 51(2-3):191–206, 2005.

[11] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal processing*, 99:215–249, 2014.

[12] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In *2009 IEEE International Conference on Robotics and Automation*, pages 2859–2865. IEEE, 2009.

[13] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[14] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *2012 IEEE International Conference on Robotics and Automation*, pages 2537–2542. IEEE, 2012.

[15] Linjun Li, Yinglong Miao, Ahmed H Qureshi, and Michael C Yip. Mpc-mpnet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints. *IEEE Robotics and Automation Letters*, 6(3):4496–4503, 2021.

[16] Zachary Kingston, Mark Moll, and Lydia E Kavraki. Exploring implicit spaces for constrained sampling-based planning. *The International Journal of Robotics Research*, 38(10-11):1151–1178, 2019.

[17] Albert Wu, Sadra Sadraddini, and Russ Tedrake. R3t: Rapidly-exploring random reachable set tree for optimal kinodynamic planning of nonlinear hybrid systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4245–4251. IEEE, 2020.

[18] Pouria Tajvar, Pierre-Jean Meyer, and Jana Tumova. Closed-loop incremental stability for efficient symbolic control of non-linear systems. *IFAC-PapersOnLine*, 54(5):121–126, 2021.

[19] Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106, 2004.

[20] C. I. Sprague, Ö Özkahraman, A. Munafo, R. Marlow, A. Phillips, and P. Ögren. Improving the Modularity of AUV Control Systems using Behaviour Trees. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6, November 2018.

[21] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[22] Bastian Schürmann and Matthias Althoff. Guaranteeing constraints of disturbed nonlinear systems using set-based optimal control in generator space. *IFAC-PapersOnLine*, 50(1):11515–11522, 2017.

[23] Antoine Girard, Colas Le Guernic, and Oded Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In João P. Hespanha and Ashish Tiwari, editors, *Hybrid Systems: Computation and Control*, pages 257–271, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[24] Patryk Cieślak. Stonefish: An Advanced Open-Source Simulation Tool Designed for Marine Robotics, With a ROS Interface. In *OCEANS 2019 - Marseille*, June 2019.

[25] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & knowledge engineering*, 60(1):208–221, 2007.

[26] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[27] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

[28] Jacob Roll, Alberto Bemporad, and Lennart Ljung. Identification of piecewise affine systems via mixed-integer programming. *Automatica*, 40(1):37–50, 2004.

[29] Giancarlo Ferrari-Trecate, Marco Muselli, Diego Liberati, and Manfred Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39(2):205–217, 2003.

[30] NL Johnson, S Kotz, and N Balakrishnan. Normal distributions. *Continuous univariate distributions*, 1:156–157, 1987.