Online Model Identification for Underwater Vehicles through Incremental Support Vector Regression

Bilal Wehbe[1,2], Alexander Fabisch[1,2], and Mario Michael Krell[2,3,4]

***Abstract*— This paper presents an online technique which employs incremental support vector regression to learn the damping term of an underwater vehicle motion model, subject to dynamical changes in the vehicle's body. To learn the damping term, we use data collected from the robot's on-board navigation sensors and actuator encoders. We introduce a new sample-efficient methodology which accounts for adding new training samples, removing old samples, and outlier rejection. The proposed method is tested in a real-world experimental scenario to account for the model's dynamical changes due to a change in the vehicle's geometrical shape.**

I. INTRODUCTION

Improving control and guidance of unmanned underwater vehicles (UUVs) is still an active area of research, which is challenged by lots of uncertainties when it comes to modeling the motion behavior of such robots. Such uncertainties arise from modeling complex physical behaviors due to the interaction between the vehicle's body and its surrounding fluid which are hard to observe explicitly (e.g., induced added mass, viscous hydrodynamic damping, body oscillations, and vortex shedding [1]). The necessity of accurate motion models is evident for improved implementations of model-based control schemes, simulation purposes, and navigation. Nevertheless, when deployed in the field, sensors malfunction and drop-outs still pose a threat to UUV's mission or might even lead to losing the vehicle, in such cases an accurate motion model can be a life saver.

Motion models for underwater vehicles have been established generically in several previous studies such as [1], [2], [3]. A motion model is an explicitly defined function which maps a vehicle's control inputs (forces and moments) to the vehicle's states (position, velocity, acceleration), which is comprised of parameters such as the vehicle's inertia, hydrodynamic added mass, buoyancy, and drag coefficients. Furthermore, it was shown in previous work [4], [5] that experimentally validated motion models perform much more accurately than analytically/empirically derived ones. In practice, even a good working static model would produce wrong predictions when the actual vehicle dynamics change. In applications such as underwater robotics, temperature, viscosity, or density fluctuations of the water, change of vehicle's payload, adding or removing of external components, body wear and damage, actuator malfunctions or failure, and bio-fouling are all factors that can lead to a change in the robot's expected behavior. In long-term underwater missions, a robot can encounter unforeseen changes, and therefore adapting to such situations is an essential aspect of such missions. An example of a long-term mission is the mission to Jupiter's icy moon Europa - "Europa Explorer" [6], where a robotic system is designed for exploring an unknown environment such as the ocean of Europa. For such purpose, machine learning appeals as a promising technique for learning the interaction between UUVs and their environment, adapting their models, and detecting anomalies and failures.

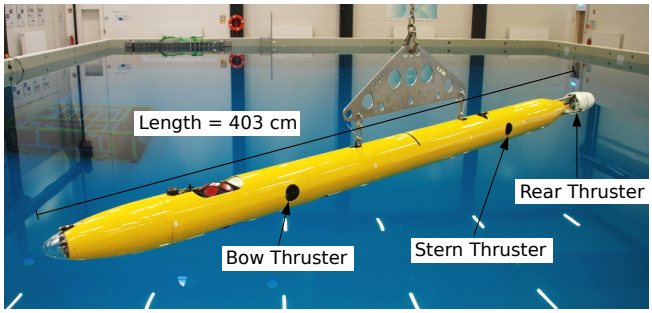


Fig. 1. AUV Leng during experiments showing locations of the thrusters.

This paper addresses the problem of learning online the damping model of an underwater vehicle subject to dynamical changes due to physical changes in the vehicle's body. As an extension to the work in [7], we present a novel approach of an incremental support vector machine regressor (IncSVR) to learn the decoupled damping term in a motion model of an UUV. The IncSVR is used here as a data driven-model, which means that the prediction of the trained model is a function of the training data samples rather than an explicit function with constant parameters.

In this work, we present the following contributions. First we introduce a new framework to learn dynamically the damping term of a motion model of an UUV, which is based on an IncSVR. We provide a novel method for including new and excluding old data samples, as well as a criterion for outlier rejection. Second, we evaluate experimentally the performance of our method by adapting online the model of the robot "Leng" (Fig. 1) in the yaw degree of freedom due to change in its dynamics. As a proof of concept, we consider the dynamical changes resulting from adding a hull to its rear thruster as shown in Fig. 2a and 2b.

We introduce the mathematical models for UUVs in Section II after a short literature review. In Section III, we introduce our machine learning approach and evaluate it in Section IV, followed by the conclusion.

[1]Robotics Innovation Center, German Research Center for Artificial Intelligence GmbH, Bremen. [2]Robotics Group, University of Bremen, Bremen, Germany. [3,4]International Computer Science Institute and University of California Berkeley, Berkeley, USA.
bilal.wehbe@dfki.de

A. Literature Review

Several studies addressed the problem of experimental and simulated identification of UUV models. Most studies addressed offline identification of either decoupled, one degree-of-freedom (DOF) models [5] where the model parameters were estimated through least squares (LS). Identification of a 6-DOF second-order coupled model of a low speed open frame vehicle with total least squares was reported in [8]. Fewer studies reported online identification, the authors of [9] employed an adaptive identifier of a decoupled model based on minimization of a Lyapunov candidate function. In [10], the identification of a 3-DOF coupled model was presented, where damping is modeled as linear and turbulent skin friction. In [3], the authors compare the performance of an adaptive identifier to LS identification of a coupled 6-DOF model. Their results show similar performance for both models. In other comparative studies between the adaptive identifier and the LS method, the authors of [11], [12] show that LS identification performs slightly better than the adaptive method. All of the above mentioned literature implement techniques to identify parameters of explicitly defined motion models, where the damping effect is approximated as a second-order function. Very few studies report data-driven models for UUVs employing machine learning techniques.

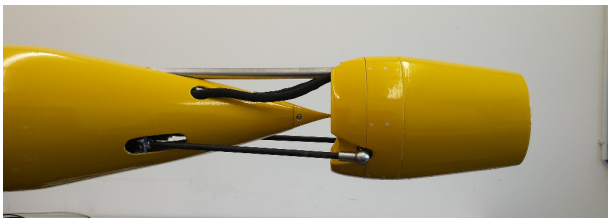
Supervised model learning has been widely used in several robotic applications such as manipulators' inverse kinematics and dynamics [13], and modeling of aerial vehicles [14]. In the field of underwater robotics, machine learning is so far rarely used. The authors of [15] used neural networks to identify the damping terms of a simulated underwater robot, where the neural network model showed better performance than the least squares one. Identification of a model underwater vehicle with a least squares support vector regression (LS-SVR) was presented in [16] by using towing tank tests. In these works, the models were trained offline. In [7], [17] we showed that data-driven methods outperforms model-based methods for both the coupled and decoupled models. In this work we extend our findings by introducing an online adaptation for the damping model based on an incremental SVR rusing real data from on-board navigation sensors of an AUV. For an overview on data selection strategies for online SVM refer to [18].

II. Mathematical Models of Underwater Vehicles

A. The model plant

The general nonlinear equations of motion of a 6-DOF underwater vehicle can be described as two main parts, the kinematic and the dynamic equations. The kinematic part maps the robot's body frame velocities $\nu = [u\ v\ w\ p\ q\ r]^T$ (surge, sway, heave, roll, pitch, yaw) onto earth-fixed-frame velocities which are namely the derivatives of the robot's position and orientation given by the vector $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T$. The dynamic plant takes into account the vehicle's added mass and Coriolis, drag, buoyant, gravitational, and external control forces. We follow the general notation of [1] for

(a)



(b)

Fig. 2. Leng's rear thruster: (a) without hull and (b) with hull.

expressing the kinematic (1) and dynamic (2) equations:

$$\dot{\eta} = J(\eta)\nu\,, \tag{1}$$

$$M\dot{\nu} + C(\nu)\nu + d(\nu) + g(\eta) = \tau\,. \tag{2}$$

We follow the definitions of [1] for the inertia, Coriolis, and gravitational/buoyant terms. The inertia matrix $M = M_{RB} + M_A$ combines both the rigid-body and the added-mass terms, as well as the Coriolis and centripetal matrix $C = C_{RB} + C_A$. For the gravitational/buoyancy (restoring) forces, a similar definition to that of [1] is followed. The damping term $d(\nu)$ is explained in detail in Section II-B. The dynamic model is assumed to have the following properties:

- The states of the system (ν, $\dot{\nu}$) are bounded, instrumented and can be measured directly or indirectly (numerically differentiated).
- The inertia matrix $M \in \mathbb{R}_{6\times6}$ is positive, symmetric, and known.
- The Coriolis matrix $C(\nu) \in \mathbb{R}_{6\times6}$ is skew-symmetric ($C(\nu) = -C(\nu)^T$), and known.
- The restoring forces and moments $g(\eta) \in \mathbb{R}_{6\times1}$ are known and constant.
- The applied input forces and moments $\tau \in \mathbb{R}_{6\times1}$ are bounded and can be measured.

B. The damping term

The hydrodynamic damping term $d(\nu) \in \mathbb{R}_{6\times1}$ represents the dissipative forces and moments, experienced by a rigid body moving through a viscous fluid. These forces are induced by the potential damping due to forced body oscillations, skin friction (due to laminar and turbulent boundary layers), wave drift, and vortex shedding [1]. Modeling these phenomena analytically is yet considered an unsolved task that is prone to a lot of uncertainties arising from the fluid's viscosity, density, temperature, and salinity or even complex geometrical shape of the robot. Several previous studies suggested rough approximations of the damping term by considering first and second order velocities only.

Here, we will express the damping term as a combination of a coupled and decoupled part,

$$d(\nu) = d_{\text{coup.}}(\nu) + d_{\text{decoup.}}(\nu)\,, \tag{3}$$

where the decoupled term maps each velocity onto a resistive force/moment in its respective DOF, whereas the coupled term accounts for the interaction between velocities from several DOFs and maps them into resistive forces and moments in the given DOFs. In this work, we identify the decoupled terms of the damping function, but keeping in mind that the decoupled terms are not enough to predict coupled motion. The online identification of the fully coupled damping term will be addressed in future work. Nevertheless, decoupled models can still be useful for cases where the robot is in hovering mode or performing sharp maneuvers in confined spaces. To ensure that only the effect of the decoupled terms is observed, only one DOF is actuated at a time, and observing that velocities in other DOFs are zero. The decoupled damping function can then be expressed as

$$d_{\text{decoup.}}(\nu) = \mathrm{d}_i(\nu_i), \tag{4}$$

where $i = 1...6$. $\mathrm{d}_i(\nu_i)$ are nonlinear functions, each mapping a linear or angular velocity into a damping force/moment in a single DOF. Given the assumptions in Section II-A, the output of the damping term can be observed by rearranging Eq. (2) as follows:

$$d(\nu) = \tau - M\dot{\nu} - C(\nu)\nu - g(\eta)\,. \tag{5}$$

We assume the following properties of the damping term:

- The inputs to the damping function are the vehicle's linear and angular velocities ν.
- The outputs of the damping term are bounded and can be measured and calculated in real-time using Eq. (5).
- The decoupled damping term is a vector of unknown nonlinear functions.
- The damping term is not a static model but can change dynamically due to several factors mentioned before in Section I.
- Variations in the damping term have a higher impact factor on the model compared to other terms, unless there is a drastic change in the shape or mass of the vehicle.

In the next Section, we introduce our extension of the support vector regression to learn the damping term and update it in real-time.

III. Incremental Support Vector Regression

First, this section introduces the support vector regression (SVR) and describes its application for solving nonlinear regression problems. Second, the incremental SVR (IncSVR) is introduced which is a method for updating the SVR with new incoming samples. Third, we introduce a novel method for handling the training data which involves criteria for adding samples and forgetting samples, and removal of outliers.

A. Overview

Support vector regression is a supervised learning method effective for modeling and interpolating nonlinear functions. SVR is explained in detail in [19]. The advantage of this method is that its final data model is represented using a small subset of the training dataset, namely the support vectors. This fact renders SVR as an attractive sparse method for robotic applications which usually suffer from low computational and memory resources. We justify further the selection of this method by the ability to perform density analysis on the support vectors which is not possible in the case of model-based or other regression methods such as neural networks.

The basic idea of SVR is to fit a function $f(x) = \langle w, x\rangle + b$ onto a training data set $\mathcal{D} = \{(x_i, y_i)|i = 1, ..., n\}$. To acquire the sparseness property, errors below some margin ϵ are not penalized. In our case, the SVR is used to fit the damping term. Hence the model can be simplified by omitting the offset parameter b, knowing beforehand that no drag forces can be produced when the vehicle is not moving and no currents are present. The regression function can be written then as $f(x) = \langle w, x\rangle$, and the weights vector w can be obtained by solving the optimization problem:

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \\ s.t. \quad & \epsilon + \xi_i \geq \langle w, x_i\rangle - y_i \geq -\epsilon - \xi_i^* \\ & \xi_i, \xi_i^* \geq 0 \quad \forall i : 1 \leq i \leq n\,. \end{aligned} \tag{6}$$

The slack variables ξ and ξ^* represent the deviation from the ϵ-tube. The hyperparameter C weights between having a more generalizing model with low weights and having too large deviations. Solving this problem requires the application of the Lagrangian multiplier technique, which by itself leads to a dual optimization problem (The reader may refer to [19] for detailed explanation):

$$\begin{aligned} \min_{\alpha,\beta} \quad & \begin{cases} \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \beta_i)(\alpha_j - \beta_j)\kappa(x_i, x_j) \\ +\epsilon\sum_{i=1}^{n}(\alpha_i + \beta_i) - \sum_{i=1}^{n} y_i(\alpha_i - \beta_i) \end{cases}, \\ s.t. \quad & 0 \leq \alpha_i, \beta_i \leq C \qquad \forall i : 1 \leq i \leq n \end{aligned} \tag{7}$$

where $\kappa(x_i, x_j)$ is a kernel function. It replaces the dot product in the optimization function and it is used to account for nonlinearities. Omitting the offset parameter b simplified the optimization problem by removing one constraint from the original formulation stated in [19]. The advantages of the dual optimization problem is that the optimization function is transformed into a quadratic form and the constraints for the dual variables α and β are simplified. For solving the optimization problem, we follow a similar approach as introduced in [20] for support vector machines (SVMs). First, we initialize the dual variables α_j and β_j to zero, and then we loop over the training samples updating the values of dual variables α_j and β_j using the following update formulas (see

[20] for more details):

$$\begin{cases} \alpha_j^{new} = \alpha_j^{old} - \dfrac{1}{\kappa(x_j,x_j)}(\epsilon + y_j - \\ \qquad\qquad \sum_i (\alpha_i^{old} - \beta_i^{old})\kappa(x_i,x_j)), \\ \beta_j^{new} = \beta_j^{old} - \dfrac{1}{\kappa(x_j,x_j)}(\epsilon - y_j + \\ \qquad\qquad \sum_i (\alpha_i^{old} - \beta_i^{old})\kappa(x_i,x_j)), \\ \text{if} \quad \alpha_j^{new} < 0 : \alpha_j^{new} = 0, \text{ if} \quad \beta_j^{new} < 0 : \beta_j^{new} = 0, \\ \text{if} \quad \alpha_j^{new} > C : \alpha_j^{new} = C, \text{ if} \quad \beta_j^{new} > C : \beta_j^{new} = C. \end{cases} \tag{8}$$

The stopping criterion can be set to a maximum number of iterations or when the differences between two consecutive update loops is less than a certain threshold. The regression function can finally be expressed as $f(x) = \sum_{j=1}^{n}(\alpha_j - \beta_j)\kappa(x_j, x)$, where the index j represents the support vectors.

B. Incremental SVR

To implement an online learning algorithm, we iterate over a small batch of training samples to optimize the target function using the method described above in every update step. For every update, a new fixed-size batch of samples is supplied to the SVR. Whenever a new training batch is added, we reuse a certain amount of old sample weights and update them instead of deleting all the weights of the previous support samples. This is a "warm start" approach which will save time, since we start from a more appropriate solution. To save memory resources, all inactive samples or the samples lying inside the ϵ-tube are deleted since their weights are zero and they have no effect on the regression function. Another aspect of the online algorithm is selecting a maximum threshold for the number of reused samples, where this threshold has to be selected depending on the dynamical properties of the system being identified. For a fast changing system, a low number of reused samples has to be selected, whereas for a slow dynamic system a larger threshold can be selected, depending on the available memory.

C. Online Implementation

In this part, we introduce a sample-efficient method for online training, where we describe our inclusion, forgetting, and outlier removal criteria to keep the number of support vectors limited and to prevent the regressor to learn only local parts of the sampling domain or to rely on artifacts in the data.

1) Inclusion criterion: At each new training step, the IncSVR will take a batch of n new training samples. The most common approach is to include all samples. This approach is computationally expensive since it can lead to adding new samples that might not have any significant influence on the regression function, but only consume memory and processing power. For example, adding new samples that fall inside the ϵ-tube or samples that are too close to previous support vectors, will not have a major effect on the regression function. To save computational and memory resources, we propose the following approach: whenever a new sample (x_i, y_i) is observed it will be discarded if one of the the following is true:

i) the sample lies inside the ϵ-tube, $|f(x_i) - y_i| < \epsilon$
ii) the sample is relatively close to an already existing support vector, $(|x_i - x_{sv}| < a,\ |y_i - y_{sv}| < b)$

where a and b are thresholds that are chosen heuristically depending on the range and resolution of the sample's domain ($\approx range/n\_samples$). For example if the samples x_i represent a velocity bounded between $-1m/s$ and $1m/s$, and we require 100 samples to describe our regression function, then a value of the threshold a would be $0.01 - 0.02\,m/s$.

2) Forgetting criterion: To keep the size of the training set bounded, old samples have to be removed or forgotten. A common way is to forget the oldest samples when new samples are added. One consequence of this approach is that samples can get concentrated locally in certain areas of the sampling domain and removed from others, which would lead to losing information about the areas with less or no samples. An example would be a robot operating at low speeds for a long period of time, which can lead to a loss of information about its behavior in higher speed ranges. To prevent such problem, we propose an approach to keep distribution of the support vectors balanced over the sample domain and remove old samples in areas of high densities first. The algorithm of forgetting samples is described as follows:

I Divide the sample domain into a number of bins with a certain resolution.
II Get the number of samples of each bin.
III While the number of support vectors is greater than the maximum number of reused samples:
  a) Find the bin with the highest number of samples.
  b) Delete the oldest sample in this bin.

3) Outlier removal: For practical applications, data samples collected by a robot's sensor are prone to faulty measurements which appear as outliers in the training data. After every training step, we check the support vectors for outliers by calculating the residuals between the support samples and their corresponding predictions

$$\text{res}_{sv} = f(x_{sv}) - y_{sv}, \tag{9}$$

where (x_{sv}, y_{sv}) is the set of support vector samples. Considering the limited computational resources of a mobile robot, we use a simple approach based on the interquartile range method. Quartiles divide the residuals set into four equal parts where $Q1$, $Q2$, and $Q3$ denote the first, second and third quartile respectively. Observations that fall outside of the range $[Q1{-}1.5(Q3{-}Q1), Q3{+}1.5(Q3{-}Q1)]$ are considered as outliers and thereby their corresponding support vectors and their weights are discarded.

4) Overall Framework: The full training chain consists of 6 main nodes as depicted in Fig. 3. The first node is the data sync node, in our case the yaw angular velocity and damping moment, where the data are logged with their corresponding timestamps and then synchronized into an array. Whenever

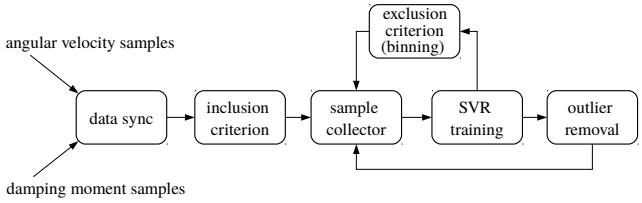


Fig. 3. Online training workflow.

the number of samples reaches a certain value n, the array is fed as a training batch to the second node where the inclusion criterion as described in Section III-C.1 accounts for discarding repeated samples or samples within the ϵ-tube. In some cases, if the whole batch of new samples are very close to existing support samples or falling within the ϵ-tube, then the full batch will be discarded and thus no training would take place in this iteration, which contributes in saving computational power. The exclusion criterion is performed by another node, where the old support vectors are removed following the algorithm described in Section III-C.2. The remaining support samples are then stacked together with the new training samples in a collector node which feeds the stacked set to the trainer and updates the sample weights. Finally, the clean-up node is responsible for removing the outlier support samples. The model could slightly change after removing outliers but we adapt to these changes in the next loop to also limit the processing effort.

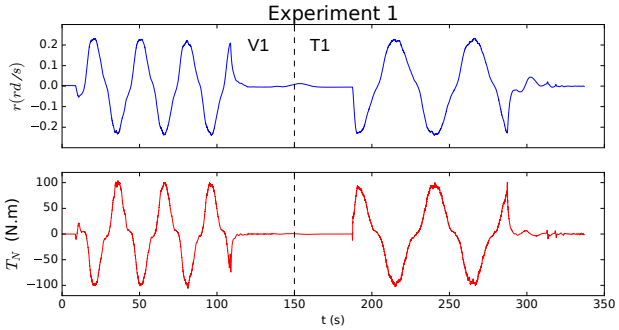
IV. Experimental Evaluation

In this Section, we describe an experiment using the AUV "Leng" (Fig. 1), where we implement the method described above to adapt the damping term in the yaw DOF before and after adding a hull module around the main rear thruster of the vehicle as shown in Fig. 2. The addition of the new module produces a clear noticeable change in the damping term of the vehicle. Experiments are carried out in a salty water basin with no waves or induced disturbances. We note that the aim of this work is not to study the effects of currents on model identification, the reader may refer to [10] for that.

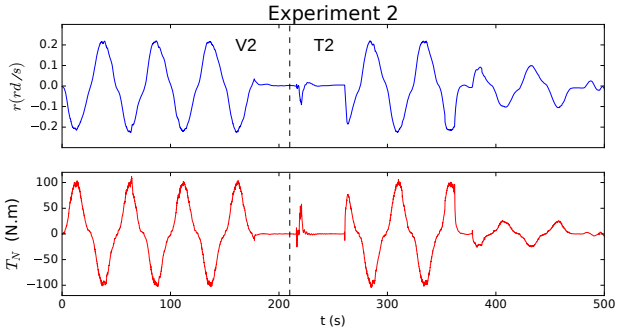
For our evaluation, we used pySPACE [21].

A. Instrumentation and Data Acquisition

AUV "Leng" is a torpedo shaped vehicle that can be actuated in five degrees of freedom excluding the roll. The mechanical properties are as follows: a length of 3.75 m, a diameter of 21 cm, and a dry mass of 76.2 kg. The moment of inertia (dry + added mass) around the pitch and yaw axes is 350 $kg \cdot m^2$. The vehicle is equipped with several navigation sensors, where in this work we use a KVH-1750 3-Axis-FOG (Fiber-Optic-Gyroscope) to measure the angular velocities. Angular accelerations are computed through numerical differentiation of the FOG data, which are then filtered with a low pass Gaussian smoothing filter. Two bow thrusters were installed in the vehicle which allow it to perform sharp turns as well as moving sideways. The thrusters are equipped with Hall-sensors to measure their



(a) Dataset from first experiment before adding the hull.



(b) Dataset from second experiment after adding the hull.

Fig. 4. Datasets from both experiments showing the training (T1 and T2) and the validation (V1 and V2) subsets.

rotational speed (ω). The produced moment in the yaw DOF is computed as ($\tau_{yaw} = \sum_i d_i(K_i^1\omega_i|\omega_i| + K_i^2 v_i|\omega_i|)$, where d_i is the distance of each thruster to the vehicle's center of mass, v_i is the relative speed of the vehicle at the point where the thruster is mounted, and (K_i^1, K_i^2) are thruster coefficients derived from CFD simulations of the thrusters.

To ensure that only the effect of the decoupled yaw damping is observed, we make sure that all other measured velocities are zero. The vehicle is thus actuated only in the yaw DOF by applying an open loop command to its bow and stern thrusters and therefore allowing the vehicle to rotate about its vertical axis. A dataset $\mathcal{D}$ can then be defined as $x_i = r_i$ representing the vehicle's yaw angular velocity, and $y_i = d_i(r_i)$ is the angular damping moment sample, computed using Eq. (5). Two experiments were carried out with the vehicle fully submerged. The first experiment was performed before adding the hull module to the rear part of the vehicle, whereas the second experiment was done after the hull was installed. The experiments were executed by actuating the thrusters with a sinusoidal command, giving maximum thrust value as amplitude and a period large enough to allow the vehicle to reach its maximum angular speed. Each of the two datasets is split into two subsets, a training and a validation set. The validation datasets, denoted as V1 and V2, are collected first to evaluate the performance of the learned damping term. The training subsets, denoted as T1 and T2, are used to train the regressor. The dashed lines Fig. 4 show the separation between the training and validation sets.

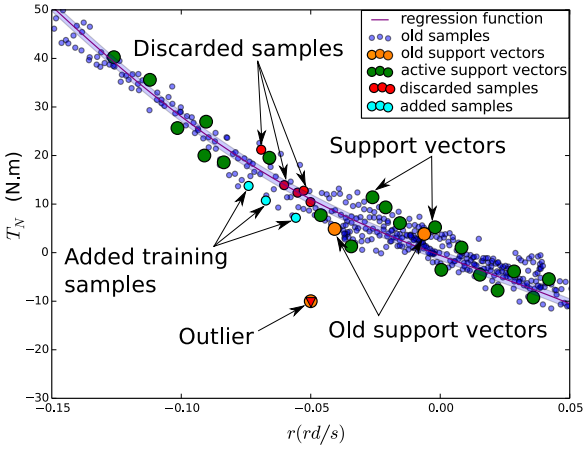


Fig. 6. Zoomed-in view showing samples added and discarded by the inclusion criterion, old support vectors removed by the exclusion criterion, and outliers removal.

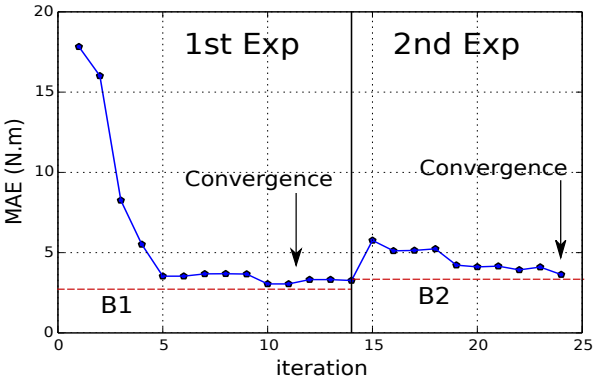


Fig. 7. MAE with validation sets vs. number of iterations

B. Parameter Tuning

The parameters of the inclusion and exclusion criteria were selected manually. Knowing that the maximum range of the yaw angular velocity, achieved by our vehicle, is between $[-0.25\,rd/s, 0.25\,rd/s]$ and the maximum torque output of the thrusters is between $[-100\,N\cdot m, 100\,N\cdot m]$, we choose the inclusion criterion thresholds as $a = 0.01\,rd/s$ and $b = 5\,N\cdot m$. The size of one training batch was set to 50 samples, and 60 old support samples reused in every new iteration. To optimize the hyperparameters of the regressor, the training dataset from the first experiment (T1) was processed offline, where we divide this dataset into batches of 50 samples to mimic the online process. We implement a grid-search algorithm with a non-randomized k-fold cross-validation, where in every run we leave out one batch for evaluation and feed in the remaining for training the regressor. After every training step, the regressor is evaluated with the batch left out using the mean absolute error metric (MAE), and then MAE is averaged over all iterations.

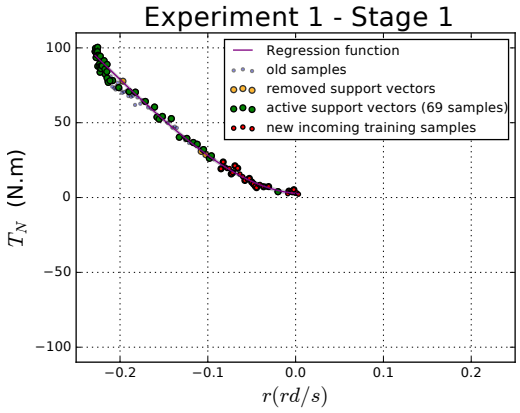
This process is repeated for every combination of hyperparameters in the following grid $\{$kernel : "RBF", C : $[10^1, 10^2, 10^3, 10^4]$, ϵ : $[0.1, 1, 2, 5]$, γ : $[5, 10, 20, 30]\}$. A radial basis function (RBF) kernel was chosen over other types of kernels due to its good capabilities to deal with nonlinear systems. An RBF kernel is expressed as $\kappa(x, z) = \exp(-\gamma||x-z||^2)$, where γ determines how far the influence of a single training example. The grid-search shows that the parameters $\{C = 1000, \gamma = 10, \epsilon = 0.1\}$ result in the best performance.

C. Online Training and Evaluation

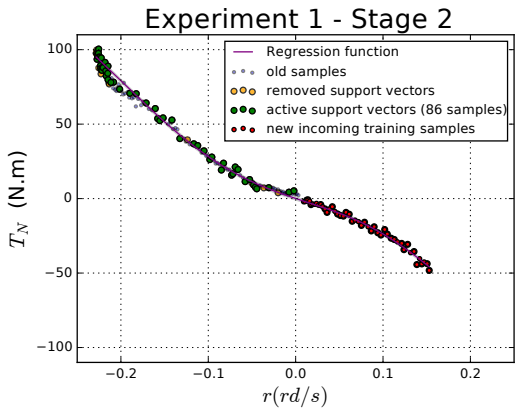
For the online training experiments, we use the set of parameters calculated in Section IV-B. The first experiment was carried out without the thruster hull where the damping model was learned after 12 iterations. While evaluating with the validation sets, convergence was assumed when the difference of the MAE between two consecutive updates is minimal and close to the value provided by the cross-validation grid-search (Fig.7). Fig. 5 shows some of the iterations where we plot the angular velocity $r\,(rd/s)$ versus the damping moment $T_N\,(N\cdot m)$. The old samples are shown in blue. The new samples, added in each iteration, are shown in red. The support samples (or support vectors) are shown in green, and the old or deleted support samples are shown in yellow. Figures 5a and 5b show the first stages of the training where samples are being accumulated along with the learned function, whereas Fig. 5c shows the last iteration of the first experiment where the full damping term has been learned. A zoomed-in view of a training iteration is depicted in Fig. 6, where the samples falling inside the ϵ-tube or in the proximity of an existing support vector are discarded by the inclusion criterion, old support vectors from high density regions are removed by the exclusion criterion, and false measurements are removed by the outlier removal node.

The second experiment was carried out after the thruster hull was installed. In Fig. 5d, the data shift can be observed due to the change of the vehicle dynamics which can be physically interpreted as a shift of the yaw drag due to the change of the geometrical shape of the vehicle's body after adding the thruster hull. At this point, an increase in the error in iteration 15 from Fig. 7 can be observed, which is explained by evaluating the old model with a new dataset. It can be noticed from Figures 5d and 5e how the old support samples where deleted after the addition of new up-to-date samples as well as the shift in the regression function towards the newly added samples. Fig. 5f shows the final update of the damping function which was reached after 10 iterations as shown in Fig. 7. Fig. 8 depicts a comparison between the old model, learned at the end of the first experiment, to the new model learned at the end of the second experiment, along with the data samples used for both experiments.

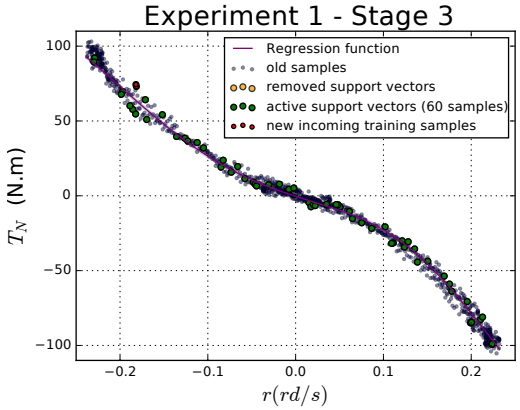
For a more concrete evaluation, we establish two baselines in which we train offline two SVRs with the same training data sets collected in both experiments, but using the full set at one time, to determine the best performance that could be achieved with an offline SVR. We optimize hyperparameters on the respective training dataset with the same grid as in the online case with a 5-fold cross-validation. We use the



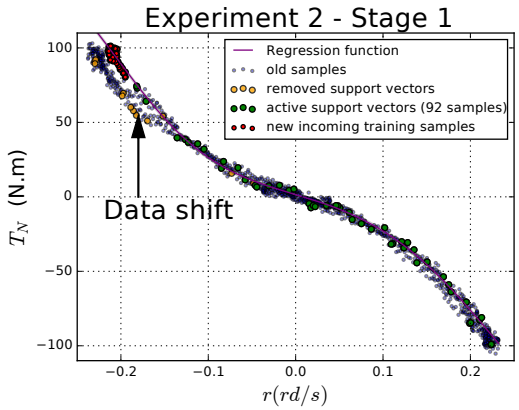
(a) First stages of the training algorithm.



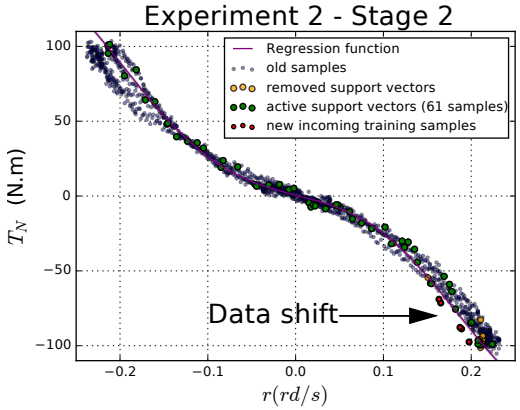
(b) Intermediate stage where data samples are accumulated.



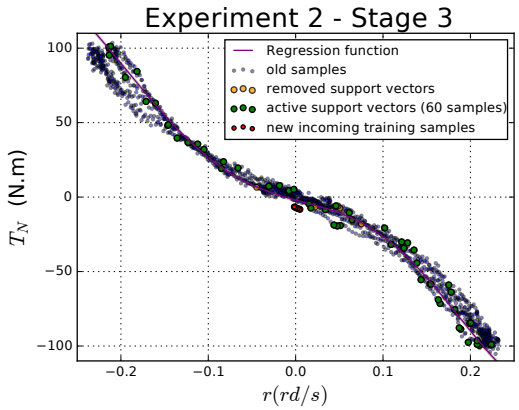
(c) Full model learned at the end of first experiment.



(d) Data shift due to change of the robot's shape.



(e) Intermediate stage of second experiment showing data shift.



(f) Adapted model at the end of second experiment.

Fig. 5. Online training iteration examples showing the learned damping function. Experiment 1: 5a-5c, Experiment 2: 5d-5f

TABLE I

METRIC EVALUATION OF LEARNED DAMPING FUNCTION.

Regressor	Validation set	MAE	$RMSE$	active sv
B1	V1	3.14 (Nm)	3.94 (Nm)	418
IncSVR1	V1	3.26 (Nm)	4.36 (Nm)	60
B1	V2	5.65 (Nm)	7.6 (Nm)	418
B2	V2	3.44 (Nm)	4.48 (Nm)	538
IncSVR2	V2	3.52 (Nm)	4.61 (Nm)	60

following terminology: for the first experiment the baseline SVR is denoted by B1, the online SVR at the first convergence point by IncSVR1, and the validation set by V1, and similarly for the second experiment (B2, IncSVR2, V2). For evaluation we use the following metrics, the mean absolute error (MAE), the root mean squared error $(RMSE)$ and the number of support vectors for each regressor. The results can be seen in Table I. In the first two rows, we show that the performance of the online method after the first convergence (with 60 support vectors) is comparable to the first baseline B1 (418 support vectors) even though it requires much less resources. The same holds for the last two rows, which also proves that the adaptation with the IncSVR2 actually works with much less active support vectors. But note that in a real-world setting, the new training which led to B2 does

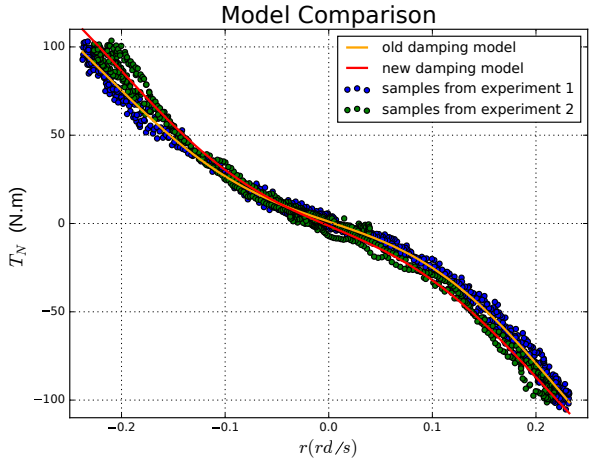


Fig. 8. Graphical comparison between the damping models before and after adding the thruster hull.

not work, but B1 would be kept static instead, for saving resources. So in fact in the more realistic comparison (row three vs. five) where we keep B1 fixed and evaluated on the new data V2, our approach shows better performance due to the incremental adaptation. We conclude as follows. The presented online method shows a good capability to adapt to the data shift resulting from the change in hydrodynamic damping. Using only a small number of support vectors, it keeps a good performance when compared to the baseline regressors that are trained offline with the full data batches.

V. CONCLUSION

In this work, we addressed the problem of learning online the damping term of a motion model of an underwater vehicle subject to dynamical changes due to physical changes in the vehicle's body, and evaluation was done on the robot "Leng" in the yaw DOF. The method presented showed good capability of adapting to the changes in the damping after adding an extra hull to the vehicle. Benefiting from the property of support vectors, the inclusion and exclusion criteria helped to keep a balanced distribution of the samples and prevented learning only local parts of the model. We note that controlling the online adaptation in such fashion is not possible in cases of neural networks or analytical-models since only weights are learned and no information about the density of the distribution is taken into account. The identification of the fully coupled damping term will be addressed in future work by combining our approach with the WSDE-SVR [17] and replacing histograms by multidimensional kernel-density estimation.

ACKNOWLEDGMENT

This work was supported by the Marie Curie ITN program "Robocademy" FP7-PEOPLE-2013-ITN-608096. The project has received funding from the German Federal Ministry of Economics and Technology (BMWi), project Europa-Explorer (grant No. 50NA1217). This work was supported by the Federal Ministry of Education and Research (BMBF, grant no. 01IM14006A) and by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD).

REFERENCES

[1] T. I. Fossen, *Marine Control Systems - Guidance, Navigation and Control of Ships, Rigs and Underwater Vehicles*. Trondheim, Norway: Marine Cybernetics, 2002.

[2] M. Gertler and G. R. Hagen, "Standard equations of motion for submarine simulation," DTIC Document, Tech. Rep., 1967.

[3] C. J. McFarland and L. L. Whitcomb, "Comparative experimental evaluation of a new adaptive identifier for underwater vehicles," in *ICRA*. IEEE, 2013, pp. 4614–4620.

[4] T. Prestero, "Development of a six-degree of freedom simulation model for the remus autonomous underwater vehicle," in *OCEANS*, vol. 1. IEEE, 2001, pp. 450–455.

[5] M. Caccia, G. Indiveri, and G. Veruggio, "Modeling and identification of open-frame variable configuration unmanned underwater vehicles," *IEEE J. Ocean. Eng.*, vol. 25, no. 2, pp. 227–240, 2000.

[6] M. Hildebrandt, J. Albiez, M. Wirtz, P. Kloss, J. Hilljegerdes, and F. Kirchner, "Design of an Autonomous Under-Ice Exploration System," in *OCEANS*. IEEE, 2013, pp. 1–6.

[7] B. Wehbe, M. Hildebrandt, and F. Kirchner, "Experimental evaluation of various machine learning regression methods for model identification of autonomous underwater vehicles." in *ICRA*. IEEE, 2017, pp. 4885–4890.

[8] S. C. Martin and L. L. Whitcomb, "Experimental identification of six-degree-of-freedom coupled dynamic plant models for underwater robot vehicles," *IEEE J. Ocean. Eng.*, vol. 39, no. 4, pp. 662–671, 2014.

[9] D. A. Smallwood and L. L. Whitcomb, "Adaptive identification of dynamically positioned underwater robotic vehicles," *IEEE Trans. Contr. Syst. Technol.*, vol. 11, no. 4, pp. 505–515, 2003.

[10] O. Hegrenaes, O. Hallingstad, and B. Jalving, "Comparison of mathematical models for the hugin 4500 auv based on experimental data," in *2007 Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies*. IEEE, 2007, pp. 558–567.

[11] J. Britto, D. Cesar, R. Saback, S. Arnold, C. Gaudig, and J. Albiez, "Model identification of an unmanned underwater vehicle via an adaptive technique and artificial fiducial markers," in *OCEANS*, Oct 2015, pp. 1–6.

[12] S. B. Gibson, B. McCarter, D. J. Stilwell, and W. L. Neu, "A comparison of hydrodynamic damping models using least-squares and adaptive identifier methods for autonomous underwater vehicles," in *OCEANS*. IEEE, 2015, pp. 1–7.

[13] S. Schaal, C. G. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Appl. Intell.*, vol. 17, no. 1, pp. 49–60, 2002.

[14] A. Punjani and P. Abbeel, "Deep learning helicopter dynamics models," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3223–3230.

[15] P. W. Van De Ven, T. A. Johansen, A. J. Sørensen, C. Flanagan, and D. Toal, "Neural network augmented identification of underwater vehicle models," *Control Eng. Pract.*, vol. 15, no. 6, pp. 715–725, 2007.

[16] F. Xu, Z.-J. Zou, J.-C. Yin, and J. Cao, "Identification modeling of underwater vehicles' nonlinear dynamics based on support vector machines," *Ocean. Eng.*, vol. 67, pp. 68–76, 2013.

[17] B. Wehbe and M. M. Krell, "Learning coupled dynamic models of underwater vehicles using support vector regression." in *OCEANS 2017 - Aberdeen*. IEEE, 2017, pp. 1–7.

[18] M. M. Krell, N. Wilshusen, A. Seeland, and S. K. Kim, "Classifier transfer with data selection strategies for online support vector machine classification with class imbalance." *Journal of neural engineering*, vol. 14, no. 2, p. 025003, 2017.

[19] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[20] M. M. Krell, "Generalizing, decoding, and optimizing support vector machine classification," Ph.D. dissertation, Bremen, Universität Bremen, 2015.

[21] M. M. Krell, S. Straube, A. Seeland, H. Wöhrle, J. Teiwes, J. H. Metzen, E. A. Kirchner, and F. Kirchner, "pySPACE – a signal processing and classification environment in Python," *Front. Neuroinform.*, vol. 7, no. 40, 2013.