Learning Coupled Dynamic Models of Underwater Vehicles using Support Vector Regression

Bilal Wehbe[1] and Mario Michael Krell[2]
[1]DFKI - Robotic Innovation Center Bremen, Germany
[2]International Computer Science Institute, University of California, Berkeley, California, USA.
bilal.wehbe@dfki.de, krell@icsi.berkeley.edu

***Abstract*—This work addresses a data driven approach which employs a machine learning technique known as Support Vector Regression (SVR), to identify the coupled dynamical model of an autonomous underwater vehicle. To train the regressor, we use a dataset collected from the robot's on-board navigation sensors and actuators. To achieve a better fit to the experimental data, a variant of a radial-basis-function kernel is used in combination with the SVR which accounts for the different complexities of each of the contributing input features of the model. We compare our method to other explicit hydrodynamic damping models that were identified using the total least squares method and with less complex SVR methods. To analyze the transferability, we clearly separate training and testing data obtained in real-world experiments. Our presented method shows much better results especially compared to classical approaches.**

Fig. 1. AUV DAGON during experimentation.

I. Introduction

The need of accurate motion models for autonomous underwater vehicles (AUVs) is evident for implementations of model-based control schemes, improved numerical simulation, and robust navigation purposes. When deployed in the field, AUVs do not usually acquire continuous pose updates, and therefore rely in some cases completely on their inertial navigation system, which is prone to drift with time. Most systems nowadays are equipped with a doppler velocity log (DVL) to aid the navigation system by providing ground relative velocity. Nevertheless due to its acoustic nature, such sensor suffers from measurements drop-outs when it loses the bottom lock. One way to remedy that situation is to use a mathematical model to aid the navigation system of the vehicle. In [1] for example, it was shown how the accuracy and robustness of the inertial navigation system of an AUV can be significantly improved by incorporating a motion model as an alternative velocity source.

Accurate modeling of rigid body submersibles can be achieved by infinite-dimensional analysis of the dynamics of the surrounding fluid, nevertheless, this comes at the cost of a huge computational complexity which renders such methodology infeasible for most practical applications [2]. Accordingly, various finite-dimensional approaches to model such vehicles were generically established in literature [3]–[5]. Generally, most of the afore mentioned modeling approaches tend to simplify the dynamics of the vehicle by making assumptions about the flow regimes and/or the geometrical symmetries of its body and thus neglecting the effects of high order nonlinearities. Challenges therefore arise when fitting mathematical model to actual data collected by the robot's sensors. In this manner, machine learning appeals as a promising technique for learning complex nonlinear models provided their inputs and outputs, and can therefore account for unmodeled aspects of the vehicle's hydrodynamics.

This paper presents an alternative method that learns a nonlinear function directly from a training dataset, without the need to predefine an explicit mathematical model and identify its parameters. The dynamic model of an AUV is viewed as a multi-input multi-output nonlinear function. Therefore without assuming any simplifications, a support vector regression (SVR) is used to estimate the underlying model. In this work the AUV DAGON (Fig. 1) is used as a study subject, where the proposed methodology is applied and the performance is evaluated using real data collected by its sensors. Finally, the methodology proposed is compared to the hydrodynamic models presented in [3], [5] that were identified using the total least squares method.

The rest of the paper is organized as the following: the mathematical models for UUVs in Section II after a brief review of the literature. In Section III, we introduce our machine learning approach used for identifying the dynamic model. In Section IV, we present the AUV DAGON and describe the experimental setup that was used for data acquisition. Section V evaluates the performance of the model being learned, followed finally by the conclusion.

A. Literature Review

The usual procedure followed for explicit model representations, is to identify the so-called hydrodynamic derivatives which quantify the forces and moments acting on the vehicle with respect to its motion states (i.e., position, velocity, acceleration). For such purpose, the least squares (LS) method is regarded as the most common technique to estimate the model parameters. An Early study [6] used the LS method to identify experimentally a 1-degree-of-freedom (DOF) decoupled model of an underwater vehicle, followed later by the work of [7] that used the same methodology but included more DOFs. An on-line adaptive identification method for a 1-DOF decoupled model was proposed by [8] which did not require acceleration measurements. In a comparative study [9], it was shown that the LS method performed slightly better than the adaptive identification method. In [10] a sensor fusion approach was used with LS to estimate the dynamics of a model decoupled into three simplified slightly interacting subsystems (speed, steering and diving), where the steering and diving subsystems are linearly coupled. In [2], the identification of a 3 DOF coupled model was presented, where the authors considered the hydrodynamic damping as three terms: a linear and turbulent skin friction. A fully coupled 6-DOF second-order model consisting of 241 parameters was presented in [5] where total least squares and an adaptive method were used to identify this model for a low speed open frame vehicle. In [11], a study showed that the Mcfarland-Whitcomb model [5] with 241 parameters performed very close to the model of Gertler and Hagen [3] with only 88 parameters. All literature mentioned so far consider techniques to identify parameters of an explicitly defined motion model, where the damping effect is approximated as a first or second order function. Fewer studies report data-driven methodologies to model the motion of underwater vehicles. In [12], a neural network was used to identify only the damping term of the model, where a simulation of an AUV was used to train the network and no real sensory data was considered. In [13] least squares support vector regression (LS-SVR) was used to identify the Coriolis and centripetal acceleration term combined with the damping terms of a model underwater vehicle by using a dataset from a towing tank experiments. Validation of the model was only done with simulated experiments. In [14], the locally weighted projection regression algorithm was used as an augmented term to correct the mismatch between the hydrodynamic model and the sensory data. So far, no studies have yet reported an estimation of a coupled model of an underwater vehicle solely based on a data-driven approach.

II. Underwater Vehicle Modeling

The motion of an underwater vehicle in general is described in 6 DOF defined as, three translations: surge, sway, and heave and three rotations: roll, pitch, and yaw. Respectively, the position and orientation with respect to an inertial coordinate frame are denoted as a vector $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T \in \mathbb{R}^{6\times 1}$. The linear and angular velocities are decomposed in the robot's body-frame and denoted as the vector $\nu = [u\ v\ w\ p\ q\ r]^T \in \mathbb{R}^{6\times 1}$. The kinematic equation of motion maps the robot's body frame velocities ν onto the inertial-frame velocities $\dot{\eta}$, namely the first order derivative of the robot's position and orientation vector η, and is given as the following:

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

where $J(\eta) \in \mathbb{R}^{6\times 6}$ is a nonlinear transformation matrix. The dynamic equation relates the robot's acceleration vector $\dot{\nu}$ in the body-frame to the robot's velocity, orientation and external forces applied to its body. Following the notation of [4] the dynamic equation is expressed as follows:

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

where M is a matrix representing the combination of the vehicle's rigid body inertia and added mass. $C(\nu)$ summarizes the Coriolis and centripetal forces as function of the rigid body and added mass. $d(\nu)$ is the hydrodynamic damping term which is defined by [4] to be a combination of potential damping, skin friction, wave drift, and vortex shedding. The term $g(\eta)$ accounts for the buoyant and gravitational forces. τ is a vector representing the actuators forces and moments. In this work we aim to estimate a data-driven model that represents the model given by (2) without the need of explicitly defining the governing mathematical equations behind it. Thus, a further detailed description of the components of (2) is not discussed in this paper. The reader may refer to [3]–[5].

By rearranging (2) as the following:

$$\dot{\nu} = M^{-1}[\tau - C(\nu)\nu - d(\nu) - g(\eta)]\,, \tag{3}$$

we can view the dynamic model as a nonlinear multivariate function that takes instances of the pose, velocity and control signal (η, ν, τ) as input, and predicts an acceleration instance $\dot{\nu}$ as output. We express the right hand side of (2) as $\mathcal{F}$. Thus the dynamic model can be written as:

$$\dot{\nu} = \mathcal{F}(\eta, \nu, \tau)\,. \tag{4}$$

The system states ν, $\dot{\nu}$, η, and $\tau \in \mathbb{R}_{6\times 1}$ of the model (4) are assumed to be bounded, and can be measured directly or inferred from other measurements.

In this work, we consider 3-DOF motion in the horizontal plane (surge, sway, and yaw). Thus, the velocity and acceleration vectors can be written as $\nu = [u\ v\ r]^T$ and $\dot{\nu} = [\dot{u}\ \dot{v}\ \dot{r}]^T$, respectively. Furthermore, we will not model the dynamics of actuators separately but rather account for it in the model directly by considering the actuators readings as direct inputs to the vehicle's model. For our case, the actuators available on Dagon are thrusters and therefore the rotational speeds (n_i) of each thrusters are given as input to the model. The model (4) can thus be rewritten as follows

$$\dot{\nu} = [\dot{u}\ \dot{v}\ \dot{r}]^T = \mathcal{F}(u, v, r, n_i). \tag{5}$$

III. Learning the Model

An evident advantage of nonlinear regression, is that only the input and output information of the system are taken into account, without the necessity of explicitly expressing the equations underlying the model function. Therefore, we account for unmodeled dynamics and reduce the assumptions and simplifications being done in previous studies. In this Section, we describe the identification methodology and the regression algorithm being used.

A. Support Vector Regression

SVR is a supervised learning method effective for modeling and interpolating nonlinear functions. One of the main advantages of this method is that it uses only a subset of the training data to represent the fitted model. This fact makes it also attractive for online implementations on mobile robots that suffer usually from limited computational and memory resources. We consider a dataset of n instances $\mathcal{D} = \{(x_i, y_i)|i = 1, ..., n\}$, where x_i denotes a feature vector (vector of descriptive variables) that characterizes the velocity and actuators variables, and y_i denotes a target vector that characterizes the robot's acceleration. The basic idea of SVR is to fit a function $f(x) = \langle w, x\rangle + b$ onto a training dataset $\mathcal{D}$, without penalizing errors that lie below some margin ϵ. The hyperparameter ϵ defines a region known as the ϵ-tube where errors are allowed. Data samples lying at the border or outside of this tube are referred to as support vectors. The weights w are then determined by solving the convex optimization problem:

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

ξ and ξ^* are slack variables that 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. To solve this problem, [15] proposed the application of the Lagrangian multiplier technique, which transforms the convex problem (6) to the following dual optimization problem:

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

where α and β are called the dual variables, and $\kappa(x_i, x_j)$ is the kernel function which is explained in Section III-B in more detail. The reader may refer to [15] for detailed explanation of the derivation of (7). Several methods can be used to solve the optimization problem (7) [16]. In this work, the sequential minimal optimization as implemented in the LibSVM interface of scikit-learn [17], [18] is used. The resulting regression function is finally written as

$$f(x) = \sum_{j=1}^{n}(\alpha_j - \beta_j)\kappa(x_j, x) + b\,, \tag{8}$$

where the index j represents the support vectors.

B. The Kernel

In nonlinear regression, the kernel corresponds to mapping the feature vectors in the input space to a higher-dimensional space where the regression problem is solved in linear form. This is done by replacing the inner product operations in the primal optimization problem (6), which is known as the kernel trick. One of the most commonly used kernels for modeling nonlinear data is the Gaussian radial-basis-function (RBF) defined as:

$$\kappa(x, x') = \exp(-\gamma||x - x'||^2), \tag{9}$$

where γ is a hyperparameters inversely proportional to the width of the kernel, and modeling the smoothness of the function. We recall from the equations of the dynamic model (5), that each of the feature inputs contributes with a different complexity to the outputs. An effective way to model functions having more than one input is to multiply several RBF kernels with different values of γ for each individual dimension [19]. In other words, different weights can be assigned for each feature of the input vector, which therefore accounts for the difference in complexity of each of the input features. We denote the resulting kernel as weighted-distance-squared-exponential (WDSE) kernel, given as:

$$\kappa(x, x') = \exp(-\sum_i \gamma_i||x_i - x_i'||^2). \tag{10}$$

where x_i is one input feature and γ_i is its corresponding weight.

In addition to the RBF and WDSE kernels mentioned above, we use a third kernel variant that is similar in sense to the WDSE kernel but uses only two hyperparameters instead. One hyperparameter is used for the velocity inputs whereas the second is used for the thrusters input. We denote this type of kernel as "2-weights-RBF".

C. Cross-validation and tuning the hyperparameters

The hyperparameters of a SVR can be summarized as the following: the loss margin parameter ϵ which determines the thickness of the ϵ-tube and therefore determines the number of support vectors used to learn the model. The regularization parameter C which acts as an overall trade of between smoothness and over-fitting of the regression function. Additional hyperparameters are introduced with the kernel. The WDSE kernel assigns a different weight γ for every dimension of the input space, thus the total set of hyperparameters are thus given as $(\epsilon, C, \gamma_u, \gamma_v, \gamma_r, \gamma_{n_i}, ...)$. With the "2-weights-RBF" kernel the hyperparameters are given as $(\epsilon, C, \gamma_\nu, \gamma_n)$, whereas with the RBF kernel we have only (ϵ, C, γ).

When evaluating the performance of a certain regression function, testing the trained function on the same data used for the training will not reflect how good the model can generally fit the data. In such case, the model would repeat the values of the samples that it has already seen but might fail to give a good prediction for yet unseen samples. To obtain a model that can learn a general description of the data, it is important to hold out a part of the data to be used for testing. This procedure is called cross-validation (CV). In this work we use an approach known as k-fold CV, where a dataset is split into k subsets. Then, the regression function is trained with k-1 subsets and the left-out set is used for testing. This procedure is repeated for every split, and the overall score is then computed as the average of the testing scores for each split. To find the optimal hyperparameters of the regression function, an exhaustive grid search algorithm was used where the parameters are selected every iteration from a grid of several candidate parameters, and therefore the ones that yield the best possible performance are chosen. As a scoring metric, we use the the coefficient of determination (R^2) which is a measure of how good the model can predict samples that were not seen before. The best possible score is 1, which indicates that the learned model can predict new output samples without error. R^2 is also a unit-less measure which helps providing a unified metric for both linear and angular DOFs.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i^{predicted} - y_i^{true})^2}{\sum_{i=1}^{n}(\bar{y} - y_i^{true})^2}, \quad \text{where} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i^{true}\,. \tag{11}$$

The following terminologies are used from now on: The *training score* is the score resulting from evaluating the model with the training data, and the *testing score* is the result of evaluation with the testing data.

In summary, for the training phase, observations of the vehicle's velocity and actuator readings (u, v, r, n_1, n_2, n_3) are given to the regressor as a set of feature inputs, and observations of the vehicle's acceleration $(\dot{u}, \dot{v}, \dot{r})$ are given as targets. Whereas in the prediction phase, the regressor calculates instances of acceleration given samples of velocity and actuator readings. We use one dataset to tune the regressor's hyperparameters and a separate dataset to evaluate how good it would predict on unseen and independent data samples.

IV. Experimental Setup

In this Section, we describe the experimental setup and the data acquisition procedure.

To test the proposed identification methodology, an experiment was setup using the AUV Dagon (Fig. 1) in a $23 \times 19 \times 8m^3$ salty water basin at the labs of DFKI - RIC in Bremen, Germany. An overview of the vehicle specifications and a description of the conducted experiment is presented.

A. Vehicle Specifications

The AUV Dagon was developed in the labs of DFKI-RIC for the purpose of underwater visual mapping and surveying near-shore continental shelf. Dagon is relatively a small sized vehicle with outer dimensions of $150 \times 80 \times 40\,cm$, and weighing $75\,kg$ in air. The vehicle is passively stable in the roll DOF, and slightly buoyant due to safety reasons. Dagon is equipped with five thrusters as shown in Fig. 2. Thrusters

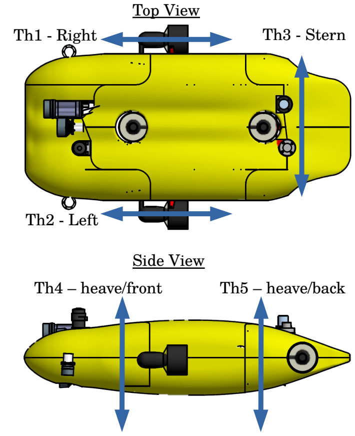


Fig. 2. Dagon's thrusters configuration: top and side views.

TABLE I
List of sensors available on Dagon.

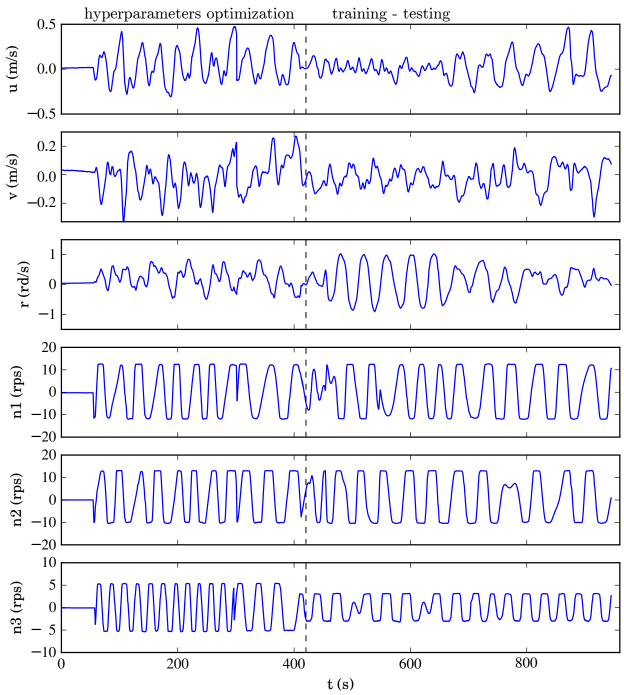
Sensor	Variable	Precision	Rate
XSens MTi AHRS	R/P/Y	(R/P): 0.5°, (Y): 1°	120Hz
KVH DSP-3000 FOG	yaw rate	1-6$°/h$	100Hz
Desert Star SSP-1 pressure sensor	depth	0.1%RMS	10Hz
Teledyne RDI Explorer DVL	ground relative velocity	$\pm 0.02 m/s$	3-4Hz

Th_1 and Th_2 are used for surge movement whereas when combined with the lateral thruster Th_3 can provide sway and yaw motion. The two vertical thrusters Th_4 and Th_5 are used for heave as well as pitch movements when actuated differentially. All thrusters are fitted with Hall sensors to measure their rotational speeds.

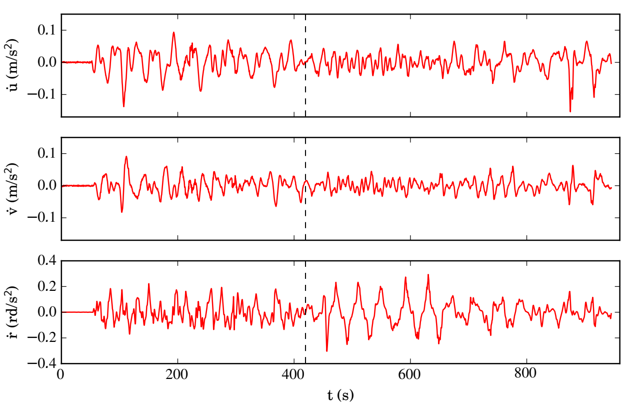
Dagon is equipped with a variety of sensors, where we only state here the ones that were used for our experiments. An attitude heading reference system (AHRS) sensor is combined with a single axis fiber-optic-gyroscope (FOG) to estimate the vehicle's orientation (roll, pitch, yaw). A Doppler velocity log (DVL) is used to measure the vehicle's linear velocity and a pressure sensor is used to estimate its depth. The specifications of the mentioned sensors can be found in TABLE I.

B. Experiment Description

As mentioned earlier, the experiments with Dagon were carried out in a salty water basin, which is a volume of static water with no waves or induced disturbances. By using the feedback of the depth and pitch readings from the vehicle's navigation system and the vertical thrusters (Th_4 and Th_5), a control loop was setup to stabilize the pitch and depth to a fixed value throughout the whole experiment. The vehicle was then driven freely in the horizontal plane by actuating its three horizontal thrusters (Th_1, Th_2 and Th_3). The thrusters received separate sinusoidal commands with zero offset and varying periods. The reason for actuating the vehicle in such fashion is to cover as much coupling as possible between different DOFs. The linear and angular velocity of the vehicle was logged in sync with the thrusters rotational velocities.



(a)



(b)

Fig. 3. Data samples: (a) input features samples representing the surge, sway, and yaw velocities as well as the thrusters rotational speed and (b) output target samples representing the accelerations in surge, sway and yaw dofs. The dotted line represents the separation between data used for hyperparameter optimization and training-testing data.

The vehicle's acceleration was calculated by numerically differentiating the velocity samples and then a mild Gaussian smoothing was applied to filter the measurement noise. The full dataset is shown in (Fig. 3), where the input feature samples are the linear velocities in the surge, sway, and yaw velocities and the rotational speeds of the 3 thrusters, i.e., $x = [u \;\, v \;\, r \;\, n_1 \;\, n_2 \;\, n_3]$, and the target data are the robot's surge, sway, and yaw accelerations, i.e., $y = [\dot{u} \;\, \dot{v} \;\, \dot{r}]$.

V. Results and Evaluation

A. Model Validation Results

In this Section, we test the performance of the model using the WDSE kernel as described in Section III. We compare the performance of the proposed model to four other modeling approaches and provide a general scheme for generating training and testing data to provide a better overall picture. As mentioned earlier, two explicit dynamic models are identified by using the total least squares method, namely the Gertler-Hagen model [3] and the Mcfarland-Whitcomb model [5]. Two other SVR approaches are also tested, the first is denoted as "RBF-SVR", that uses an RBF kernel as in (9) where only one hyperparameter of the kernel (γ) is optimized. The second SVR uses the "2-weights-RBF" mentioned in Section III-B, assigning one weight to the velocity inputs ($u \;\, v \;\, r$) and one weight to the thrusters inputs ($n_1 \;\, n_2 \;\, n_3$), and therefore optimizing only two hyperparameters of the kernel instead of six. We denote this regressor as "2-weights-SVR". The idea behind these two alternatives is to analyze the improvement in performance due to the additional hyperparameters of the WDSE kernel. In order to bring the feature inputs to a close range, the feature inputs for the RBF-SVR and 2-weights-SVR are normalized to have a zero mean and unit variance. This procedure is not necessary for the WDSE kernel since every feature of the input vector has its own weight (γ_i). We use first a subset of the data to optimize the hyperparameters of the SVRs, and the rest of the dataset to train and evaluate the model as shown in Fig. 3. We note that the sole purpose of the optimization dataset is to find the best hyperparameters, this dataset it not used for training or testing purposes. The hyperparameter optimization was done on this dataset as described in section III-C using the machine learning tool pySPACE [20]. A grid of five candidates for each hyperparameter was given to the algorithm, and the results yielding the best performance are reported in Table II. The difference in the values of γ's of the WDSE kernel indicates the difference of complexity each input feature contributes to the model underlying the data. We describe next the training and testing procedure. As mentioned earlier, a rule of thumb is to keep the training and testing data separated as well in order to evaluate how good the regressor can generalize or predict accurately unseen data. Fixing the hyperparameter values to the results of the optimization procedure, we run a

TABLE II
Results of Hyperparameter Optimization

Regressor	Hyperparameter	Surge	Sway	Yaw
WDSE-SVR	ϵ	0.01	0.01	0.01
	C	10	10	10
	γ_u	0.085	0.1	0.1
	γ_v	0.5	0.01	5
	γ_r	0.2	0.1	0.2
	γ_{n1}	10^{-5}	10^{-6}	10^{-3}
	γ_{n2}	10^{-5}	10^{-4}	10^{-3}
	γ_{n3}	10^{-5}	10^{-6}	10^{-4}
2-weights-SVR	ϵ	0.001	0.001	0.005
	C	1	1	2
	γ_ν	0.019	0.019	0.01
	γ_n	0.001	0.001	0.09
RBF-SVR	ϵ	0.001	0.01	0.001
	C	5	10	1
	γ	0.01	0.01	0.04

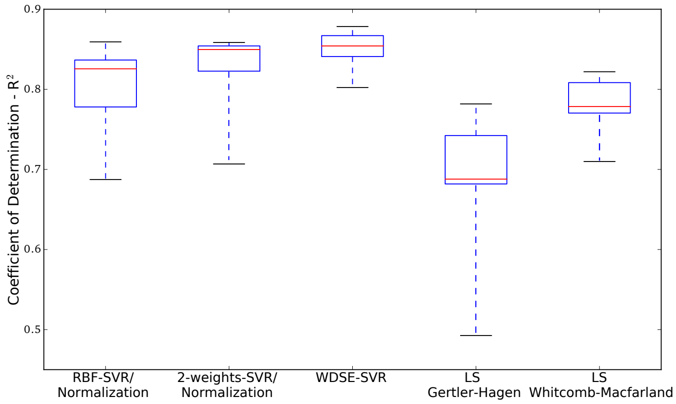


Fig. 4. Testing scores of different models evaluated through 5-fold cross-validation using R^2 performance metric.

5-fold cross-validation scheme on the training-testing dataset shown in Fig. 3. Note that the k-fold scheme does not use any randomization for picking the training and testing subsets. This operation results in five scores for each model that are shown as box-plots in Fig. 4. The red line represents the median value, and the bottom and top edges of the box represent the interquartile range, and the whiskers represent the minimum and maximum values. The distances between the different parts of the box-plot can be seen as a measure of dispersion in the results, therefore a tight box-plot indicates a less dispersed data and a wide box-plot indicates a wide dispersion. Table III shows the mean and standard deviation of the evaluation scores. The results are discussed next.

B. Discussion

Results from Fig. 4 and Table III show clearly that when compared to the other models, the SVR using the WDSE kernel has the highest average and the minimal spread of the performance scores. This indicates that the WDSE-SVR keeps consistently a good performance when trained with different splits of the dataset and thus being able to capture more accurately the underlying dynamics of the vehicle. By comparison to the two other SVR variants, the WDSE kernel demonstrates a good capability of learning the capturing the complexity of each degree of freedom on the model due to its additional hyperparameters. Additionally, the WDSE-SVR avoids over fitting the training data (or learning the noise), since the training and testing scores yield close values. On the other hand, the explicit models show a poorer performance, the Gertler-Hagen model for example shows a wide spread of the performance scores which indicates that when trained with different splits of the data, the model's performance varies between having a good fit for certain parts of the data and for other parts not. In summary, the WDSE-SVR outperforms the other approaches presented, followed by the "2-weights-SVR" with a reduced number of its kernel hyperparameters. The RBF-SVR and the Mcfarland-Whitcomb model show relatively close performance, whereas the Gertler-Hagen model falls last.

TABLE III

MEAN AND STANDARD DEVIATION OF CROSS-VALIDATION TRAINING AND TESTING SCORE RESULTS (R^2).

Model	training score		testing score	
	Mean	S.D.	Mean	S.D.
RBF-SVR	0.935	0.002	0.797	0.061
2-weights-SVR	0.906	0.005	0.818	0.057
WDSE-SVR	0.909	0.005	**0.849**	**0.026**
Gertler-Hagen	0.782	0.011	0.677	0.099
Mcfarland-Whitcomb	0.861	0.003	0.778	0.039

VI. Conclusion

In this work we reported a new methodology to model coupled dynamic models for underwater vehicles based upon support vector regression. The dynamic model was assessed as an unknown nonlinear multivariate function that maps the vehicle's body velocities and actuator inputs onto the body accelerations. A SVR algorithm incorporating a kernel that associates different weights to the feature inputs of the model was used to learn the model dynamics with access to its inputs and outputs. The AUV Dagon was used as a platform to collect the data necessary for the model identification. The method presented showed good capabilities fitting unseen data that was not used in the training process. Furthermore the proposed method was cross-validated against four other modeling approaches, where it showed the best performance among the other candidates.

As for future work, evaluating this methodology on different types of underwater vehicles with more complex geometrical structures will be addressed, as well as testing this approach on-line with the possibility of varying dynamics of the robot considered.

Acknowledgment

The authors would like to thank all colleagues at DFKI GmbH, Robotics Innovation Center, Bremen for their support and feedback. This work was supported by the Marie Curie ITN program "Robocademy" FP7-PEOPLE-2013-ITN-608096 and by a fellowship within the FITweltweit program of the German Academic Exchange Service (DAAD).

References

[1] O. Hegrenas, E. Berglund, and O. Hallingstad, "Model-aided inertial navigation for underwater vehicles," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1069–1076.

[2] 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.

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

[4] T. I. Fossen, *Marine control systems: guidance, navigation and control of ships, rigs and underwater vehicles*, 2002.

[5] 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.

[6] 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.

[7] J. P. J. Avila, J. C. Adamowski, N. Maruyama, F. K. Takase, and M. Saito, “Modeling and identification of an open-frame underwater vehicle: The yaw motion dynamics,” *J. Intell. Robot. Syst.*, vol. 66, no. 1-2, pp. 37–56, 2012.
[8] 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.
[9] 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.
[10] K. M. Fauske, F. Gustafsson, and O. Hegrenaes, “Estimation of auv dynamics for sensor fusion,” in *Information Fusion, 2007 10th International Conference on*. IEEE, 2007, pp. 1–6.
[11] 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.
[12] 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.
[13] 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.
[14] G. Fagogenis, D. Flynn, and D. M. Lane, “Improving underwater vehicle navigation state estimation using locally weighted projection regression,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6549–6554.
[15] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.
[16] M. M. Krell, “Generalizing, decoding, and optimizing support vector machine classification,” Ph.D. dissertation, Bremen, Universität Bremen, 2015.
[17] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, feb 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078195
[19] D. Duvenaud, “Automatic model construction with gaussian processes,” Ph.D. dissertation, University of Cambridge, 2014.
[20] 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.