Experimental Evaluation of Various Machine Learning Regression Methods for Model Identification of Autonomous Underwater Vehicles

Bilal Wehbe, Marc Hildebrandt and Frank Kirchner[1]

***Abstract*— In this work we investigate the identification of a motion model for an autonomous underwater vehicle by applying different machine learning (ML) regression methods. By using the data collected from the robot's on-board navigation sensors, we train the regression models to learn the damping term which is regarded as one of the most uncertain components of the motion model. Four regression techniques are investigated namely, artificial neural networks, support vector machines, kernel ridge regression, and Gaussian processes regression. The performance of the identified models is tested through real experimental scenarios performed with the AUV Leng. The novelty of this work is the identification of an underwater vehicle's motion model, for the first time, through machine learning methods by using the robot's on-board sensory data. Results show that the damping model learned with nonlinear methods yield better estimates than the simplified linear and quadratic model which is identified with least-squares technique.**

I. INTRODUCTION

Building reliable motion models for autonomous underwater vehicles (AUVs) is an essential procedure in the aim for improved navigation, guidance and control techniques as well as localization and mapping purposes. Although underwater vehicles are nowadays used extensively in many commercial and scientific applications, methods for controlling these vehicles in an optimal fashion are still a big area of research. Classically, motion model parameter identification for AUVs is done by implementing towing tank experiments of the vehicle itself or of a down-sized prototype, but this comes at cost of time consuming and expensive procedures. Furthermore this does only give a one-shot measurement and does not take into account any changes of the system which may happen in the future. Throughout the past two decades, underwater roboticists investigated alternative methods for motion model identification which involved basin or sea trials experimentation, by which the vehicle's dynamical behavior is observed using its on-board navigation sensors.

The necessity of accurate vehicle models in the light of improving sensor equipment of such vehicles is still apparent when employing AUVs in the field: sensors malfunction, create drop-outs (especially the Doppler velocity log (DVL) is prone to up to 20 % of drop-outs depending on the environment, see [1]), consume too much energy, or lose bottom-lock during long descents into large depths. In addition there is a number of situations where sensor information is severely limited by the environment, e.g. changes in salinity between the robot's deployment point and the location of the desired mission. The advent of long-term missions such as a mission to Jupiter's icy moon Europa [2] stress the requirement of a very reliable localization and navigation system, and thus result in renewed research activities with respect to improving mathematical vehicle modeling.

All authors are with DFKI RIC, Germany, 28359 Bremen. [1] Department of Mathematics and Informatics, University of Bremen
{bilal}.{wehbe}@dfki.de

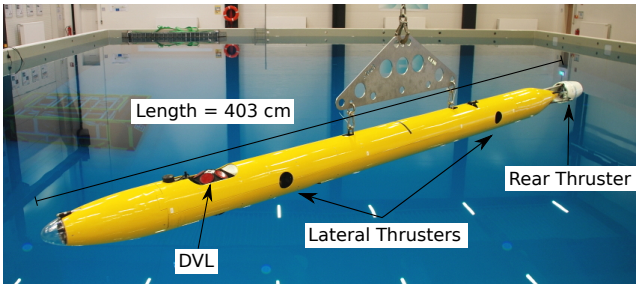


Fig. 1. The AUV Leng during testing with thrusters positions identified.

Identification methods are normally categorized into off-line and on-line methods. For off-line methods data sets are collected by the vehicle's sensors and then filtered and processed after. One of the most common off-line techniques that can be found in literature is the widely used least-squares (LS) identification method in [3], [4], [5]. Tiano et al. [6] proposed an observer Kalman filter to cope with measurements noise and mild nonlinearities. On the other hand, on-line identification methods collect data and estimate the system dynamics while the vehicle is in real-time operation. This allows for automatic update of the model parameters and adapting the vehicle's behavior on the run, which is more reliable in cases where the geometrical or physical characteristics of the vehicle can change or environmental conditions are not steady. An on-line adaptive identification method was proposed by Smallwood [7] that is independent of acceleration measurements and associates a Lyapunov function to assure the convergence of the identified parameters. Karras et al. [8] described the augmentation of an Unscented Kalman Filter (UKF) for an on-line parameter estimation algorithm of an underwater vehicle. In a comparative study, Britto et al.[9] showed that least-squares method performed slightly better than the adaptive identification method.

Generally, most of the techniques mentioned earlier tend to over-simplify the identified model, assuming only low speed models and thus neglecting the effects of high order nonlinearities and coupled motion. To cope with such nonlinearities, machine learning (ML) was first used in underwater robotic control with the work of Yuh [10] where he described a self adapting control system based on neural

networks. Van den Ven et al. [11] identified the damping terms of a simulated underwater robot using neural networks. Identification of a model underwater vehicle with support vector machines was presented by Xu et al. [12] by using towing tank tests. So far in literature there are no reports of implementing ML methods for motion model identification of underwater vehicles relying on real data for on-board navigation sensors.

The contributions in this work are as follow: first is using the data collected from an underwater vehicle's navigation sensors and actuators to train four different ML regression models along side with least squares identification. In addition to neural networks (NN) and support vector machines (SVM), we implement two other regression learning methods to identify our vehicle's motion model, that were not addressed in literature before, namely Gaussian process regression (GPR) and Kernel Ridge regression (KRR). For every case, the models are trained off-line with data sets collected in a basin trial, and then performance of the identified models is evaluated through cross validation by using data sets from several testing experiments, and thus ensuring the separation of the training data from the testing data. Second, we show that using the learning methods to identify the damping of an underwater vehicle outperforms the least-squares identification with the simplification of the damping to only linear and quadratic terms. In the conclusion we give an insight about the extendability of the ML methods for learning the damping term on-line.

II. Motion Model Identification

The classical nonlinear dynamic equations of motion of a six degrees of freedom underwater vehicle is expressed as the following [13]:

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

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

where $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T$ is the position and orientation vector expressed in the earth-fixed or inertial frame, $\nu = [u\ v\ w\ p\ q\ r]^T$ is the linear and angular vector in the body-fixed frame. $M = M_{RB} + M_A$ is the combination of the rigid body and added mass inertia matrix. $C(\nu)$ is a matrix representing the Coriolis and centripetal terms. $d(\nu)$ is the damping term, $g(\eta)$ is a vector representing the restoring terms due to gravity and buoyancy. τ is the control forces and moments vector. In general, the damping terms are considered to be the parts of motion model that are most uncertain, as Fossen describes the damping matrix to be the effect resulting from potential damping, linear and quadratic skin friction, wave drift damping, and damping due to vortex shedding combined. Modeling these terms accurately is considered to be a hard task. In this work we will express the damping term as a combination of a coupled and decoupled part,

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

The decoupled term maps separately the velocity in a certain DOF onto a resistive force/moment in the same DOF, whereas the coupled term accounts for the interaction between velocities from several DOFs. In this work, we focus on identifying the decoupled terms of the damping function, but being fully aware that the decoupled terms are not enough to predict coupled or complex motion. The identification of coupled dynamics is to be addressed by the authors in future work. The decoupled damping is then expressed as follows

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

where $i = 1...6$. $\mathrm{d}_i(\nu_i)$ are nonlinear functions, each mapping a linear or angular velocity into a damping force or moment in a single DOF. We assume the following properties of the model:

- The states of the system (ν, $\dot{\nu}$, η, and $\dot{\eta}$) are bounded, and can be measured.
- 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.

Given these assumptions the damping output can then be calculated by rearranging Eq. (2) as follows:

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

A. Least-Squares Identification

Least-squares identification as introduced by [3], [4] was implemented on simplified 1-DOF dynamic equation, using the decoupled damping. Assuming the vehicle operate at small velocities and geometrically symmetrical in all three planes, the damping was simplified into linear and quadratic skin friction, expressed as follows:

$$d_{\text{decoupled}}(\nu) = \mathrm{d}_i(\nu_i) = d_{l_i}\nu_i + d_{q_i}\nu_i|\nu_i|, \tag{6}$$

where d_{l_i}, d_{q_i} are constant damping coefficients. The least squares method is then applied to estimate those two coefficients for every DOF. Assuming a data set $\mathcal{D} = \{(x_k, y_k)|k = 1, ..., n\}$ where x_k represents a velocity sample and y_k a damping force/moment sample in a given DOF. Let $P = [x_k,\ x_k|x_k|]$ be a matrix of combined linear and quadratic velocities, and $\Theta_i = [d_l, d_q]$ is the vector of unknown parameters. The solution of Θ is then calculated by the pseudo-inverse of matrix P as,

$$\Theta = (P^T P)^{-1} P^T y_k\,. \tag{7}$$

B. Nonlinear Regression

An evident advantage of nonlinear regression, when applied to underwater robots model identification, is that only the input and output information of the system are taken into account, without the necessity of explicitly expressing the complex hydrodynamic effects. Therefore, assumptions done with the least-squares method are avoided. As mentioned earlier, determining the damping term $d(\nu)$ is considered to be the hardest task because of its high uncertainties. In

this section, we describe the identification of the decoupled damping term $d(\nu)$ assuming no simplifications, and with access to its inputs and outputs. We assume that the decoupled damping term is a vector of unknown nonlinear functions with the vehicle's velocities as inputs, and the outputs are resistive forces and moments which are bounded and can be calculated using Eq. (5).

In the following paragraph a brief background of the methods mentioned above is presented.

1) Neural Network Regression: A neural network (NN) or also called a multilayer perceptron (MLP) is a learning algorithm that learn a nonlinear function $y = f(x, w)$, where x is an feature input, y is a target output, and w is a weight vector. The simplest element of a neural network is a neuron which is a function itself that computes the weighted sum in its inputs and then apply an activation function to generate an output. Neurons can be connected to form a layer and so consecutive layers can be connected forming a so-called multilayer perceptron. The output is calculated through forward propagation where the output of every neuron is fed into the neurons connected ahead of it, the output of the last layer is then the generated output of the perceptron. For regression purposes, the weights w are adjusted through a training process which is an optimization algorithm that minimizes an error function.
A NN requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and a regularization parameter α. For our application, the design of the neural network follows the advice of [14], where the architecture of the network is kept as simple as possible. We use a single layer perceptron and tune the number of neurons and the regularization with a grid search cross-validation algorithm. Since we are dealing with a regression problems, a $\tanh$ as activation function for the hidden layers, and a linear activation for the output layer.

2) Kernel Ridge Regression: The kernel ridge regression is comprised of a combination of a ridge regression (least squares with an L2 norm regularization) with a kernel trick [15]. A kernel function $\kappa(x, z)$ is defined as a real-valued, symmetric, and non-negative function mapping two arguments x and z into real space $\mathbb{R}$. The regression method is realized by applying a kernel trick to the primal linear ridge regression problem, which is simply replacing all inner product operations by the kernel mapping. For our problem we use a Gaussian radial basis function (RBF) kernel defined as follows:

$$\kappa(x, z) = exp(-\gamma||x - z||^2). \tag{8}$$

The γ parameter determines how far is the influence of a single training example. For small γ the model is too constrained and cannot capture the complexity of the function, but a very large gamma means a bigger area of influence of a single training sample. The regularization hyperparameter α is a parameter that prevents over-fitting of the function. KRR uses squared error loss function to optimize the samples weights (for detailed explanation refer to [15]), where the regression function can be expressed as

$$f(x) = \sum_{i=1}^{n} \beta_i \kappa(x_i, x)\,. \tag{9}$$

3) Support Vector Regression: The goal of support vector regression (SVR) is to fit a function $f(x)$ onto a data set (x_i, y_i) without caring about errors that are less than a value ϵ [16]. Like KRR, SVR uses a kernel function to map the data in the input space to a high-dimensional space where the regression problem can then be processed in a linear form, but optimizes with a quadratic ϵ-intensive loss function. In addition to the hyperparameters γ of the RBF kernel, a regularization hyperparameter C is introduced which is a trade off between smoothness and over-fitting of the function. The hyperparameter ϵ represents the width of a tube $\{x_i / |f(x_i) - y_i| < \epsilon\}$, by which the corresponding weights of samples lying inside are zero and have no effect on the regression function.

$$f(x) = \sum_{j=1}^{n} (\alpha_j - \beta_j)\kappa(x_j, x)\,, \tag{10}$$

where dual variables α and β represent the weights of the support vectors. A more detailed explanation of SVRs can be found in [16].

4) Gaussian Processes Regression: In Gaussian processes the data are regarded as noisy observations. A GPR uses also a kernel but rather to determine the covariance of a prior Gaussian distribution over the target function, and uses the training data to define a likelihood function [15]. A prediction is then calculated as the mean of the (Gaussian) posterior distribution over the target function, based on Bayes theorem. One advantage of GPR is that hyperparameters of the kernel are optimized during fitting. Furthermore, since the GPR learns a probabilistic model of the target function, it can provide confidence intervals along with the predictions. The noise level in the data is learned explicitly by the GPR by adding a WhiteKernel to the original kernel [17].

III. Experimental Setup and Data Acquisition

A. Experimental Setup

Experiments were done with an autonomous underwater vehicle (AUV) called "Leng" Fig. 1, that was designed as a prototype vehicle during the project "Europa Explorer" [2] which is a pilot survey for future missions to Jupiter's moon Europa. The main purpose of the vehicle is under-ice autonomous exploration, which is also applicable for terrestrial missions such as Arctic environments. The vehicle design consists of a torpedo-shaped hydrodynamic hull with a length of 4.03 m, a diameter of 21 cm and a dry mass of 76.2 kg. It is equipped with numerous localization sensors, of which the DVL and 3-Axis-FOG (Fiber-Optic-Gyroscope) have been used in the experiments described here. Two lateral thrusters allow sway and yaw movements. In this paper, we describe experiments conducted in a $23 \times 19 \times 8m^3$ salty water basin without any waves or perturbation disturbances.

As mentioned in II the focus of this study is the identification of decoupled damping, thus the experiments were carried out by actuating a single degree of freedom at a time. The dynamics in sway/heave, and pitch/yaw are considered to be identical due to the robot's symmetry, whereas the roll is considered to be passively stable. In this work the vehicle was actuated in the sway and yaw DOF separately, by applying a sinusoidal open-loop command to the sideway thrusters. Motion in the sway DOF was achieved by applying identical commands to both thrusters, whereas the yaw motion was performed by applying commands of same magnitude but opposite direction to the same thrusters. The thrusters where actuated given their maximum achievable speed as amplitude with a periodicity enough to ensure that the full velocity range of the vehicle was covered.

B. Data Acquisition

Data from the on-board navigation sensors mentioned earlier as well as the thrusters rotational speed are logged with their corresponding time-stamp. The DVL sensor with a precision of +/-2 $\times\ 10^{-3} m/s$ is used to measure linear velocities, whereas the 3-Axis-FOG with precision of +/- $2 \times 10^{-4} deg/s$ is used for angular velocities, both measured in the robot or body frame. Accelerations are computed through numerical differentiation of the velocity data, which are both for smoothing purpose filtered with a low pass filter Gaussian smoothing filter. Thrust is calculated as a function of the propeller's rotational velocity ω, and the torque is calculated as a thrust force multiplied by the distance from the thrusters to the center of the vehicle (d_c), which are given by [13] as follows

$$\begin{aligned} Thrust &= K\omega|\omega|\,, \\ Torque &= Thrust\ \times\ d_c\,. \end{aligned} \tag{11}$$

For each DOF two data sets were collected, one set for hyperparameter optimization and training and another set for testing.

IV. Training and Evaluation

Evaluating the learned models with the same data used for training is a major mistake, the model in this case would have a very good score since it is tested with the samples it used for fitting. To avoid that, the training and testing data has to be separated. A common method when using machine learning is cross-validation (CV), where a test set is always left out for evaluation. In this work we use the k-fold CV approach by which the data set is split into k sets randomly, then the model is trained with (k-1) sets and evaluated with the one set remaining. This procedure is repeated for every fold, and the overall performance is then computed by the average of the testing scores computed at every step. To optimize the hyperparameters of the learning methods we apply a exhaustive grid-search with a 5 fold cross-validation and a mean absolute error (MAE) as scoring for each method. The model is trained with 4 folds of the data every time then the resulting model is tested with the remaining fold which was not used for training, then the average of the values computed. The parameter grid for each method is given as follows

- SVR: {kernel: [rbf], gamma: [1,5,10,20,30], epsilon: [10^{-3}, 10^{-2},..., 10^{1}], C: 10^{0}, 10^{1}, ... , 10^{4}]}.
- KRR: {kernel: [rbf], alpha: [10^{-3}, 10^{-2}, ... , 10^{0}], gamma: [10^{-3}, 10^{-2}, ... ,10^{2}]}.
- NN: {num. of nodes: [1,5,10,20,50], alpha: [10^{-3}, 10^{-2}, ... , 10^{0}]}.

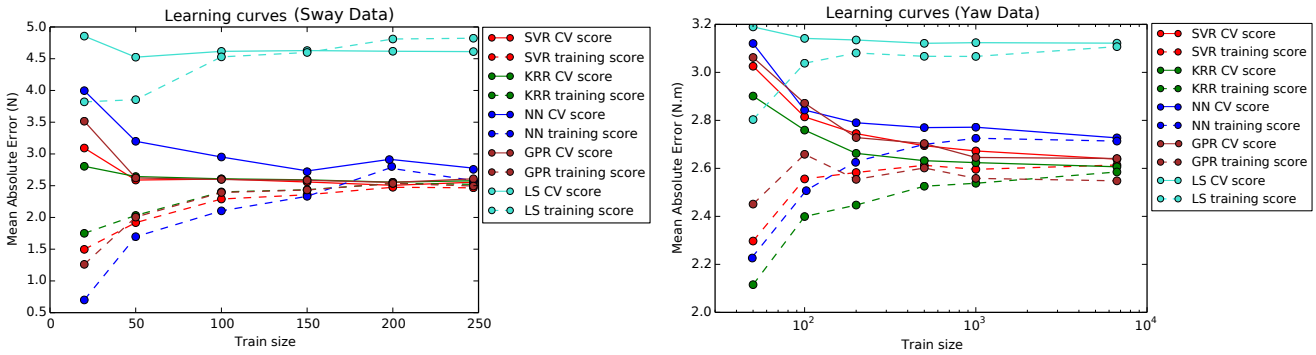
TABLE I

Cross-validation results for hyperparameter optimization.

regressor	DOF	best params	best score (MAE)
SVR	Sway	gamma: 10, epsilon: 1, C: 10^3	2.361 (N)
	Yaw	gamma: 30, epsilon: 1, C: 10^3	2.853 (N.m)
KRR	Sway	gamma: 10, alpha: 10^-2	2.617 (N)
	Yaw	gamma: 10^2, alpha: 10^-3	2.412 (N.m)
NN	Sway	num. of nodes: 5, alpha: 10^-5	2.725 (N)
	Yaw	num. of nodes: 5, alpha: 10^-2	2.338 (N.m)
GPR	Sway	l-s: 0.276, n-l: 1.74 (N)	2.256 (N)
	Yaw	l-s: 0.087, n-l: 1.76 (N.m)	2.749 (N.m)
LS	Sway	N/A	4.401 (N)
	Yaw	N/A	3.221 (N.m)

The GPR does not require a grid-search to optimize its hyperparameters but rather relies on a gradient-based optimization of the parameters during the training process. The kernel for the GPR was specified as a combination of an RBF and a WhiteKernel to learn the noise level in the data and prevent overfitting, therefore two hyperparameters were optimized, namely the length scale (l-s) of the RBF and the noise level (n-l) of the WhiteKernel. The least squares method has no parameters to optimize, but will only be evaluated with the 5-fold CV. Results of the for best performing parameters are reported in Table I. From the scoring metric in Table I we can observe in both sway and yaw DOFs a better performance of the nonlinear regression models over the least-squares model.

For a more generalized evaluation of the models, we need to check if the learned models are under-fitting or over-fitting the data. Under-fitting is when the training model is not complex enough to fit the training data, while over-fitting is when the model learns the noise in the data instead of the real function. One method is to calculate the training and the validation scores (or errors) of every estimator. If both the training and validation scores are low (low score means high error), then the estimator under-fits the data. If the training score is high but the validation score is low then the model is over-fitting. If both scores converge to a good value, then the estimator can generalize better. In Fig. 2 we show the variation of the training and testing scores of each method with increasing size of training samples, where the significance of these plots is to determine the sufficient number of samples needed for each estimator to converge as well as the comparing generalizing performance of each. If the validation and training scores converge to a low score (high error) with increasing size of training data, then the estimator will not benefit from adding more training data. Practically an estimator that doesn't require too much training samples yet can still achieve good performance is preferred, since it can save memory and computational



(a) Learning curves for sway data

(b) Learning curves for yaw data

Fig. 2. Learning curves: validation vs. training scores with varying number of training samples.

resources. Both Fig. 2a,2b show consistent results, where in both cases the nonlinear estimators outperforms the least-squares one. For a low number of samples, the least-squares estimator gives higher training and validation errors, where it under-fits the trained model. The NN tends to over-fit the data for low sample size since it has the biggest difference between the training and validation scores, while the other three kernel-based methods (KRR, SVR, GPR) still perform better even with small training samples sizes. Adding more samples does not improve the performance of the least-squares estimator a lot, whereas for the nonlinear estimators we can notice an improved performance as the training and validation scores converge to better values. We can also notice clearly that the NN estimator converges slower than (needs more samples) the other kernel-based estimators. The performance of the kernel-based estimators looks relatively close but nevertheless to further compare these methods, we list some of their advantages and disadvantages. The KRR and SVR employ different loss functions, where the KRR uses a squared error loss while the SVR uses an ϵ-sensitive loss. The KRR learns a function faster than the SVR for medium sized data sets ($<$10K), but the learned model is not sparse which means that the whole set is used to give a new prediction. The SVR on the other-hand learns a sparse model, since it only uses the support vectors for producing a prediction and thus becomes faster than the KRR for bigger data sets. The advantages of GPR is that it gives a probabilistic prediction by which therefore one can compute a confidence interval around it. With a very small dataset GPR can also be used to predict interpolation between the observations, as well as dealing with noisy data. The main disadvantages of this method is that training time grows exponentially with the training samples size which renders it not very efficient for on-line learning. Fig. 3 shows a comparison of learning and prediction time for the estimators using the same processor (core-i7 CPU). For a more robust evaluation, another two experiments for in the sway and yaw DOFs were conducted, where the trained models are used as feed-forward prediction of the damping term in the plant model of the vehicle. Fig. 4 shows a graphical comparison between the identified models and the measured

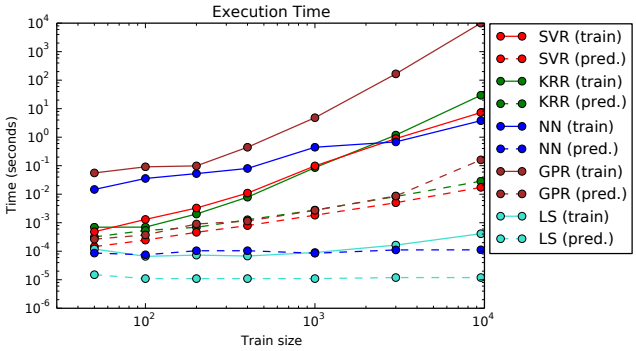


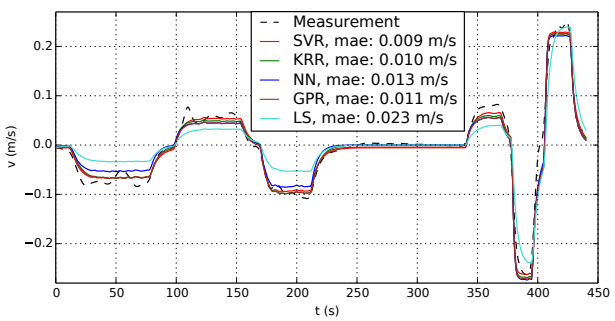
Fig. 3. Learning and prediction time of different estimators

sway and yaw velocities. By computing the mean absolute error between the model prediction and the measured linear and angular velocities in Fig. 4, we can observe that the mean absolute error of the kernel based methods is smaller than that of the LS and the NN estimators.

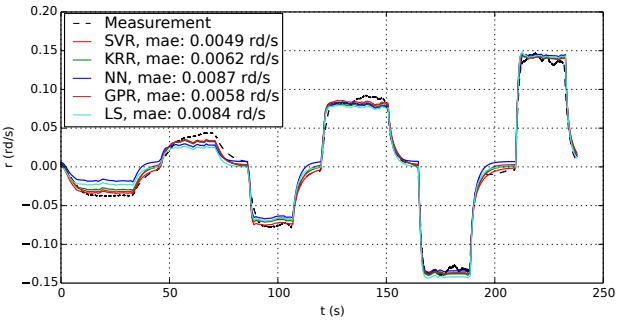
We conclude as follows, the kernel based nonlinear estimators (SVR, KRR, GPR) yield better estimations for hydrodynamic damping of underwater vehicles than NN and least squares estimators. Between the three kernel methods, first, GPR is a good candidate when training with small data set since it produces probabilistic predictions, can interpolate between samples and is capable to deal with noisy data. Second, KKR has a good performance with medium sized datasets when fast learning and prediction times are of concern. Third, SVR shows very good performance as well as effectiveness with high number of samples since it uses a subset of the samples for predictions which helps in saving memory resources. This sparseness feature of SVR renders it as an attractive for on-line implementation.

V. CONCLUSIONS

The main contribution of this work is the novel application of four machine learning regression methods (NN, SVR, KRR, and GPR) for the identification of the motion model of an underwater vehicle by using on-board navigation sensory data, which concludes that nonlinear regression methods



(a) Sway experiment.



(b) Yaw experiment

Fig. 4. Measured velocity compared to model predicted velocity in sway and yaw DOFs.

show better capabilities to learn the hydrodynamical properties of underwater vehicles. The least-squares method was applied estimate the parameters of the simplified form of the damping term which takes into account only the linear and quadratic skin friction. On the other hand, the nonlinear methods assumed an unknown form of the damping term and the model was estimated accordingly. The identified models were tested with two experimental scenarios for each degree of freedom, where results shows better performance of the nonlinear methods over the classical least-squares method. This work concludes that the simplified linear and quadratic model identified with least squares method is not enough to describe the hydrodynamic damping properties of underwater vehicles and that nonlinear machine learning regression methods are better alternatives to capture the unmodeled dynamics of such vehicles. With a comparative study the following can be summarized. For small data sets, Gaussian process regression learn a probabilistic model where it can provide a meaningful confidence interval around every prediction but can be computationally inefficient as the number of samples increase. Kernel ridge method is best suited for medium sized datasets where it can learn the model fast and give accurate predictions. Sparse methods such as SVRs are best suited for bigger number of samples but nevertheless produce very good results with smaller sample sizes, and can be also extended for on-line identification of motion models. As future work the learning of coupled motion dynamics has to be considered as well as optimizing the performance and reducing computational effort of such learning methods for on-line implementation.

ACKNOWLEDGMENT

This work was supported by the Marie Curie ITN program Robocademy FP7-PEOPLE-2013-ITN-608096. This work is part of the Europa-Explorer project (grant No. 50NA1217) which is funded by the German Federal Ministry of Economics and Technology (BMWi).

REFERENCES

[1] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter, "Visually Mapping the RMS Titanic: Conservative Covariance Estimates for SLAM Information Filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1223–1242, Dec. 2006. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364906072512

[2] M. Hildebrandt, J. Albiez, M. Wirtz, P. Kloss, J. Hilljegerdes, and F. Kirchner, "Design of an Autonomous Under-Ice Exploration System," in *In MTS/IEEE Oceans 2013 San Diego, (OCEANS-2013)*. IEEE, 2013, pp. 1–6.

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

[4] 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," *Journal of Intelligent & Robotic Systems*, vol. 66, no. 1-2, pp. 37–56, 2012.

[5] S. Natarajan, C. Gaudig, and M. Hildebrandt, "Offline experimental parameter identification using on-board sensors for an autonomous underwater vehicle," in *Oceans, 2012*. IEEE, 2012, pp. 1–8.

[6] A. Tiano, R. Sutton, A. Lozowicki, and W. Naeem, "Observer kalman filter identification of an autonomous underwater vehicle," *Control engineering practice*, vol. 15, no. 6, pp. 727–739, 2007.

[7] D. A. Smallwood and L. L. Whitcomb, "Adaptive identification of dynamically positioned underwater robotic vehicles," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 4, pp. 505–515, 2003.

[8] G. C. Karras, C. P. Bechlioulis, M. Leonetti, N. Palomeras, P. Kormushev, K. J. Kyriakopoulos, and D. G. Caldwell, "On-line identification of autonomous underwater vehicles through global derivative-free optimization," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3859–3864.

[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'15 MTS/IEEE Washington*, Oct 2015, pp. 1–6.

[10] J. Yuh, "A neural net controller for underwater robotic vehicles," *Oceanic Engineering, IEEE Journal of*, vol. 15, no. 3, pp. 161–166, 1990.

[11] 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 Engineering Practice*, vol. 15, no. 6, pp. 715–725, 2007.

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

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

[14] A. Fabisch, Y. Kassahun, H. Wöhrle, and F. Kirchner, "Learning in compressed space," *Neural Networks*, vol. 42, pp. 83–93, 2013.

[15] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

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

[17] 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 E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.