A Mapping Approach for Controlling Different Maritime Cranes and Robots Using ANN

F. Sanfilippo, L. I. Hatledal and H. Zhang
Department of Maritime Technology and Operations
Aalesund University College
Postboks 1517, 6025 Aalesund, Norway
{fisa, laht, hozh}@hials.no

K. Y. Pettersen
Centre for Autonomous Marine Operations and Systems
Department of Engineering Cybernetics
Norwegian University of Science and Technology
7491 Trondheim, Norway
kristin.y.pettersen@itk.ntnu.no

***Abstract*— In [1], a flexible and general control system architecture that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms was previously presented by our research group. Each manipulator can be controlled by using the same universal input device regardless of differences in size, kinematic structure, degrees of freedom (DOFs), body morphology, constraints and affordances. The architecture presented establishes the base for the research of a flexible mapping procedure between a universal input device and the manipulators to be controlled, which is the topic of this paper.**

Based on the same architecture, as a validating case study, a new method for implementing such a mapping algorithm is introduced in this paper. This method is based on the use of Artificial Neural Networks. Using this approach, the system is able to automatically learn the inverse kinematic properties of different models. Learning is done iteratively based only on observation of input-output relationship, unlike most other control schemes. Related simulations are carried out to validate the efficiency of the proposed mapping method.

***Index Terms*— Control architecture, Artificial Neural Networks, manipulators.**

I. Introduction

In the maritime field, even though the operating environment can be very challenging, it is still quite common to use relatively simple control interfaces to perform offshore crane operations. Moreover, each input device normally controls only one specific crane model. When considering working efficiency and safety, this kind of control is extremely difficult to manage and extensive experience is required of the operators. Therefore, low control flexibility and non-standardisation are two crucial issues of the current maritime crane control architecture that need to be overcome.

Maritime cranes, compared with robotic arms, rely on a much more complex model of the environment with which they interact. These kinds of cranes are widely used to handle and transfer objects from large container ships to smaller lighters or to the quays of the harbours. Therefore, their control is always a challenging task, which involves many problems such as load sway, positioning accuracy, wave motion compensation and collision avoidance.

In [1], our research group presented a general architecture that allows for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device. The main challenge in doing this consists of finding a flexible way to map the fixed DOFs of the universal input device to the variable DOFs of the cranes or robots to be controlled. This process has to be realised regardless of their differences in size, kinematic structure, body morphology, constraints, affordances and so on. The architecture allows for designing and testing different mapping procedures.

By using the same architecture, as a validating case study, a new way to implement such a mapping algorithm is presented in this paper. This method is based on the use of Artificial Neural Networks (ANN) [2] and using this approach, the system is able to automatically learn the inverse kinematic (IK) properties of different models.

The paper is organised as follows. In Section II, a review of the related research work is given. In Section III, we briefly summarize the control architecture proposed in our previous work and shortly discuss a possible trajectory tracking approach. As a validating case study, a new way to implement a flexible mapping method based on the use of ANN is presented. Related simulation results are shown in Section IV. Finally, in Section V, conclusions and future works are outlined.

II. Related Research Work

In existing literature, only a few results have been reported on overcoming the low control flexibility and non-standardisation problems for the current maritime crane control architecture. In [3], Li and Wang presented a visual simulation system for a shipborne crane. The system can realise the visual simulation for trajectory-planning, joint control and dynamic analysis. However, most of the previous studies only concern the control of a specific crane/arm. Very little work has been done regarding the possibility of controlling different arms by using the same input device.

In [4], our research group presented a modular prototyping system architecture that allows for modelling, simulation and control of different robotic arms by using the *Bond Graph Method*. The main drawback of this approach is that the complexity of the system tends to rise when considering a large number of DOFs.

A common assumption in all these previous works is that the forward kinematic (FK) model of the arm to be controlled is a priori knowledge. Classically, this assumption enables researchers to either introduce analytical methods, which offer exact solutions for simple kinematic chains, or propose solutions based on numerical methods. However, when considering arms with redundant DOFs, the IK can have multiple solutions, and as such, singularity problems could arise. In addition, this method is not very flexible, especially when planning to control different arms using a universal input device because several IK models are needed: one for each arm or crane to be controlled. An alternative approach to the problem might consist of using methods that do not assume a priori knowledge for the IK model of the arm: a solution that derives its kinematic properties from a machine learning procedure. In this way, the system would be able to automatically learn the kinematic properties of the manipulator to be controlled. This idea has been pursued by several scientists. In fact, during the last few years, there has been increasing interest regarding research on learning algorithms and many efforts have been made to understand how to apply this technology to various control problems. In particular, several ANN models have been developed by applying biologically-inspired control mechanisms to robot control tasks. Especially, in order to deal with complex robotic systems and with the related non-linear problems that arise when considering sophisticated types of actuators, several ANN models have been developed. In [5], Wang et al. presented a Lagrangian neural network for the IK computation of redundant manipulators based on the Euclidean norm of the joint velocities. This was developed at first to show its feasibility. Next, in the same work a primal-dual neural network for minimum infinity norm kinematic control was presented. To reduce the model complexity and increase the computational efficiency, a dual neural network was finally introduced with the advantages of simple architecture and exponential convergence. However, the simulation results are based only on a specific industrial robot, the *PA10* robot manipulator. Moreover, the model does not exhibit biologically realistic behaviour. In [6], Bouganis et al. presented a spiking neural network architecture that autonomously learns to control a 4 DOFs robotic arm after an initial period of *motor babbling* (motor babbling can be observed in babies, where a repetitive action-perception cycle generates associative information between the various representations). The spiking neurons have been simulated according to Izhikevichs model [7], which exhibits biologically realistic behaviour and yet is computationally efficient. These works demonstrate that ANNs can be used to model complex relationships between inputs and outputs.

On the other hand, most of these previous intelligent systems are only able to learn the control of a specific crane/arm. To date, it is not possible to use a common universal input device to control various cranes/arms with different kinematics. Moreover, most of these works require the same DOFs for both the input device and the model to be controlled.

III. System Architecture and Case Study

A. Architecture

In this section we briefly summarize the control architecture proposed in our previous work. For further details, please see [1]. The proposed control system architecture is shown in Fig. 1. It is a client-server architecture with the input device running as a client and communicating with a server where the logic of the control algorithm is implemented. The controlled arms are simulated in a 3D visualisation environment, which also acts as a client and provides the user with an intuitive visual feedback. The proposed architecture provides the possibility of controlling the arms in position mode or velocity mode. To realise these two possible operation modes, when the operator manoeuvres the manipulator, a vector signal with no semantic, $\mathbf{s}$, is sent from the universal input device to the server. Here, according to the operational scenario, the vector signal is interpreted as the desired position $\mathbf{x}_d$ or the desired velocity vector $\dot{\mathbf{x}}_d$.

Additionally, in order to adjust the size of the input device's workspace to the arm to be controlled, a scaling factor is introduced to calculate the coordinate of the point to be reached. The proposed architecture allows for expanding and shifting the small-scale physical workspace of the input device to a virtual expanded workspace allowing the robot arm for more accurate and precise movements. In particular, referring to Fig. 2 and denoting the reference frame of the input device's physical workspace with O_i, the reference frame of the input device's virtual workspace with O_v, and the reference frame of the manipulator workspace with Ow, the desired scaled position, $\mathbf{x}_{ds}$, is calculated as follows:

$$\mathbf{x}_{ds} = k_p \mathbf{x}_d + \mathbf{x}_w, \tag{1}$$

where k_p is the position scaling factor and $\mathbf{x}_w$ is a shifting vector that defines the position of the virtual reference frame with respect to the global reference frame. Similarly, the desired velocity vector can also be scaled to allow the operator to execute slower or faster movements according to the task to be accomplished. The desired scaled velocity vector, $\dot{\mathbf{x}}_{ds}$, can be obtained as follows:

$$\dot{\mathbf{x}}_{ds} = k_v \dot{\mathbf{x}}_d, \tag{2}$$

where, k_v is the velocity scaling factor.

Then, according to the desired mode of operation, the mapping control algorithm parses those values to the desired

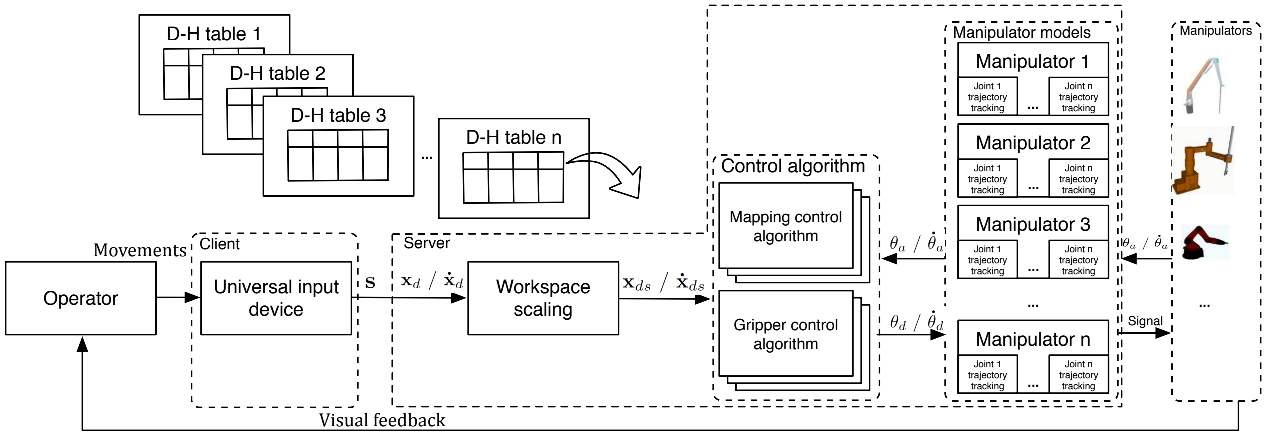


Fig. 1: The proposed control system architecture.

joint angles θ_d or desired joint velocities $\dot{\theta}_d$ of the manipulator, respectively. Essentially, for all the different models to be controlled, the mapping methods have to implement the classic IK functions that can be generalised as follows:

$$\theta_d = f_p^{-1}(\mathbf{x}_{ds}), \tag{3}$$

concerning position control, and

$$\dot{\theta}_d = f_v^{-1}(\theta_a, \dot{\mathbf{x}}_{ds}), \tag{4}$$

for velocity control, where θ_a is the the actual joint angles vector.

The calculated desired joint angles θ_d or joint velocities $\dot{\theta}_d$ are then forwarded from the server to the visualisation environment in order to actuate the crane model. As feedback from the visualisation environment, the actual joint angles θ_a and joint velocities $\dot{\theta}_a$ are sent back to the server and can be used by the control mapping algorithm.

Notice that the proposed architecture allow for implementing different mapping methods. Each mapping control algorithm has to realise the mapping between the fixed DOFs of the universal input device and the variable DOFs of the manipulator to be controlled. It is important that each control algorithm be implemented as an independent and interchangeable module and that it satisfies the interface specified by the system, (3) and (4), in order to respect the modularity of the proposed architecture.

A relevant feature of the proposed architecture is that the robot model can be separated from the control algorithm to be used. In particular, no matter which control algorithm is used, the manipulators to be controlled can be added to the system simply by defining their corresponding standard Denavit-Hartenberg (D-H) tables [8] and their joint limits.

For all the models to be controlled, the different mapping methods calculate the corresponding sampling point configurations for the desired end-effector's positions. In other words, each mapping method works as a motion planner. In order to ensure smooth movements for the manipulators it is necessary to generate trajectories out of these given sampling points. A well-suited trajectory is the basic prerequisite for the design of a high-performance tracking controller and ensures that no kinematic nor dynamic limits are exceeded. Such a controller guarantees that the controlled robot will follow its specified path without drifting away. Therefore, feedback control has to be applied to be able to compensate external disturbances as well as disturbances from communication time delays. Note that time data is a free parameter because the sampling time of the mapping algorithm is generally not constant.

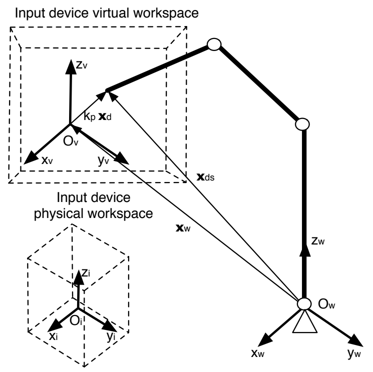

Input device virtual workspace
Input device physical workspace

Fig. 2: Input device physical and virtual workspaces.

A possible solution for generating well-suited trajectories consists of using a Proportional Integral Derivative (PID) controller for each joint, as shown in Fig. 3. Notice that using this approach, the nature of the crane actuators - whether they are hydraulic, pneumatic, electric or mechanical - can be also taken into account. However, since the main focus of this work is on building an effective mapping method, all problems related to rope pendulations or wave impacts on the payload are not considered in this paper but they can be

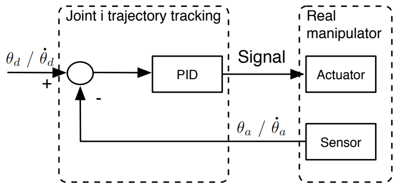

Joint i trajectory tracking
Real manipulator
θ_d / $\dot{\theta}_d$
+
-
PID
Signal
Actuator
θ_a / $\dot{\theta}_a$
Sensor

Fig. 3: Trajectory tracking using a PID controller.

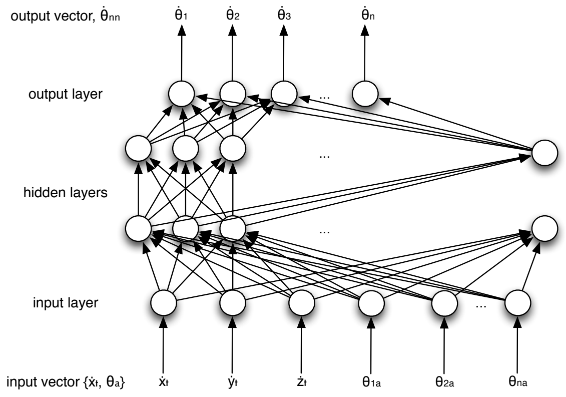


Fig. 4: The proposed ANN architecture.

included in the model at a later stage.

B. A Mapping Method Based on ANN

As a validating case study, an alternative mapping method that does not assume a priori knowledge for the IK model of the arm to be controlled is presented in this subsection. This approach is based on the use of a supervised feed forward ANN. In particular, the system automatically learn the mapping function, (4), for the different manipulators to be controlled. This approach only requires the FK models. Note that the unique feature of this method compared to previous works is that the same set-up of the proposed algorithm is adopted independently of which manipulator is being controlled and whether the selected control mode is position or velocity. Moreover, when controlling each specific manipulator and once selecting the particular control mode, the same instance of ANN is continuously used; what differs are the semantics and the size of inputs and outputs which are dynamically and automatically set by the system.

The network architecture of this validating case study is shown in Fig. 4. It is a three-layer ANN which consists of one input, one output and one hidden layer. Each neuron in the network is fully connected with every other neuron of the next layer. Adaptive weights are associated to the neuron's interconnections. A sigmoid transfer function is used as an activation function. Note that these particular arbitrary choices have been selected for this preliminary case study, however different learning patterns can be easily tested and used at a later stage. The input vector is given by $\{\dot{\mathbf{x}}_t, \theta_a\}$, where $\dot{\mathbf{x}}_t$ is the target velocity whereas θ_a is the actual joint configuration. The target velocity, $\dot{\mathbf{x}}_t$, depends on the operation scenario and it is given by:

$$\dot{\mathbf{x}}_t \simeq \mathbf{x}_{ds} - \mathbf{x}_a, \tag{5}$$

if operating in position control mode, where $\dot{\mathbf{x}}_a$ is the actual end-effector velocity. If instead operating in velocity control mode, it is given by:

$$\dot{\mathbf{x}}_t = \dot{\mathbf{x}}_{ds}. \tag{6}$$

From the top side, the output vector, denoted by $\dot{\theta}_{nn}$, consists of the target joint velocities. Notice that the number of neurons in the input and output layers changes according to the number of DOFs of the manipulator to be controlled. The number of hidden neurons is experimentally chosen to be equal to $3/2$ of the sum of the size of the input layer and the size of the output layer. In this specific case study, the control of the end-effector's orientation is not considered as part of the mapping algorithm but it can easily be included at a later stage without affecting the effectiveness of the architecture presented. In the following, the key steps of the algorithm are described.

1) Error Function: the error of every sample is evaluated by using a error function that assesses the Mean Square Error (MSE) between the desired joint velocity $\dot{\theta}_d$ and the actual joint velocity $\dot{\theta}_a$ for n training samples:

$$MSE(\dot{\theta}_d, \dot{\theta}_a) = \frac{1}{n}\sum_{i=1}^{n}(\dot{\theta}_d - \dot{\theta}_a)^2. \tag{7}$$

2) Training Data: One of the biggest challenges when studying the IK of a serial manipulator by using ANNs consists of generating a suitable training data set, a set of example pairs $(\{\dot{\mathbf{x}}_t, \theta_a\}, \dot{\theta}_{nn})$. Different researchers have applied different methods for gathering training data, some of them use the kinematics equations, some of them use the network inversion method, some of them use the cubic trajectory planning, some of them use a simulation program for this purpose, and some use data recorded experimentally from sensors fixed on each joint. In this specific case study, the Jacobian matrix is used. First, a set of samples is randomly generated in the Joint space, then the Jacobian matrix is built by using the differential approach. Successively, a set of random joint velocities is created and used to calculate the corresponding velocities in the Cartesian space.

3) Training Process: in supervised learning, the aim is to find a function in the allowed class of functions that matches the examples. In other words, the objective is to infer the mapping implied by the data. To do that, the weights that are associated with the neuron's interconnections are continuously updated during the training process. The *Resilient Propagation* (RPROP) learning heuristic is used as training method. The RPROP approach was proposed in [9] by Riedmiller et al. to overcome the inherent disadvantages of pure gradient-descent. RPROP performs a local adaptation of the weight-updates according to the behaviour of the error function. Contrary to other adaptive techniques, the effect of the RPROP adaptation process is not blurred by the unforeseeable influence of the size of the derivative, but only dependent on the temporal behaviour of its sign. This leads to an efficient and transparent adaptation process.

4) Present Output: According to the operational scenario, the output is obtained by:

$$\theta_d = \int \dot{\theta}_{nn} \, dt, \tag{8}$$

when operating in position control mode, or as:

$$\dot{\theta}_d = \dot{\theta}_{nn}, \tag{9}$$

when operating in velocity control mode.

IV. Simulations Results

In this study, a *Microsoft Xbox 360* joystick controller is used as a universal input device on the client side. Each DOF of the joystick corresponds to a translational axis in the workspace of the crane to be controlled. When operating in position control mode, the joystick works as a position proportional replica whose motion maps exactly to the motion of the crane end-effector with constant speed, while, when operating in velocity control mode, a movement of the joystick in a particular direction will produce a translational motion in the same direction at a velocity proportional to the joystick displacement. In both operation cases, when the operator's hand is removed from the joystick, the latter automatically returns to its starting point. Note that thanks to the modularity of the architecture, any other joystick or input device can be used without influencing the effectiveness of the system.

From an implementation point of view, the logic of the control architecture lies on the server side, which is implemented by using the *Java* programming language. Each manipulator to be controlled is modelled as a *Java* class which embodies a D-H table, a set of joints, a workspace as attributes and a *Solver* as an abstract subclass. The *Solver* abstract subclass has two methods - *positionSolver* and *velocitySolver* - which have the prototypes that the mapping functions have - (3) and (4) respectively. The ANN mapping method described in the previous section is a particular implementation of this *Solver* but new mapping methods can easily be added by simply providing a corresponding implementation of the same abstract subclass.

To speed up the developing process and to improve the reliability of the system, several libraries are used. In particular, the *Efficient Java Matrix Library* [10] is adopted to add support for matrix manipulations, while the ANN is implemented by using the *Encog Java neural network framework* [11]. Moreover, the manipulators to be controlled can easily be added to the system by simply defining their

TABLE I: D-H table of the knuckle boom crane, where $L_1 = 2.62m$, $L_2 = 7.01m$ and $L_3 = 3.46m$

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	L_1	θ_1
2	$\frac{\pi}{2}$	0	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	0

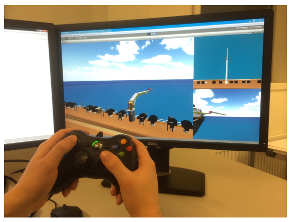


Fig. 5: The simulated knuckle boom crane model.

corresponding D-H tables and their specific joint limits in an *XML* document.

The system is based on a distributed structure and communication between client, server and the visualisation environment is realised by using the *TCP/IP* protocol. This also makes it possible to remotely control the different manipulators. Regarding the visualisation environment, in this preliminary work, the game engine *Unity3D* [12] is used to visualise the different models. However, any other visualisation environment could be used without affecting the effectiveness of the proposed architecture.

Related simulations are carried out in order to test the architecture within the particular case study of the proposed mapping method. In detail, as shown in Fig. 5, a knuckle boom crane with 3 DOFs is modelled and simulated. See Table I for the crane's D-H table. In this specific case, the number of neurons in the input layer is 6, the number of neurons in the output layer is 3 and the hidden layer consists of 14 neurons. The learning curve for the network configuration is shown in Fig. 6. The training process is ended after 300.000 iterations with a CPU time of 2094s (based on a Intel Core i7-3820QM machine) and a MSE of 0.0066. For the same model, a trajectory tracking analysis of the Cartesian paths for X, Y and Z coordinates is also performed, measuring the difference between desired and calculated position before the PID regulation process. The results are shown in Fig. 7. The proposed system demonstrates quite a fast reaction to the inputs and reasonable output error considering the dimension of the controlled model.

V. Conclusion and Future Work

Based on the control system architecture that our research group recently presented for modelling, simulation and control of different models of maritime cranes and, more generally, robotic arms by using the same universal input device, a new mapping algorithm has been presented in this

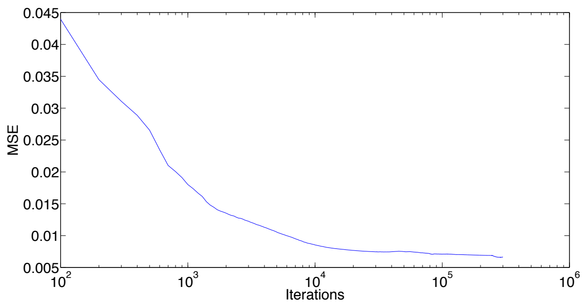
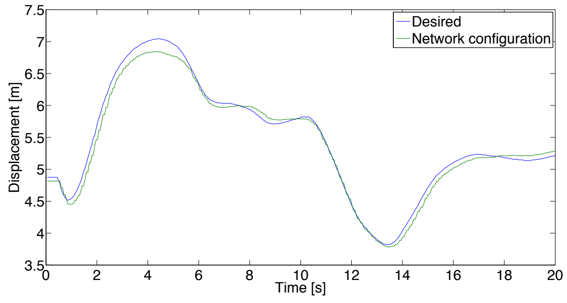
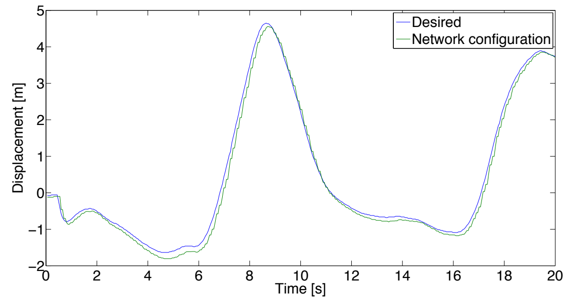


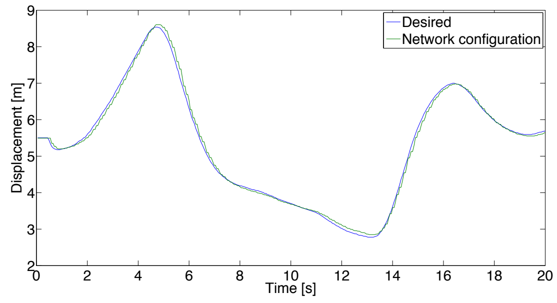
Fig. 6: The learning curve after 300.000 iterations (abscissa is scaled logarithmically).



(a)



(b)



(c)

Fig. 7: Trajectory tracking after the training is finished for (a) the **X** coordinate, (b) the **Y** coordinate and (c) the **Z** coordinate.

paper as a validating case study. This method is based on the use of ANNs. Using this approach, the system is able to automatically learn the IK properties of different models. Learning is done iteratively based only on observation of input-output relationships, unlike most other control schemes.

As future work, it would be interesting to compare different mapping methods and their corresponding performances. In order to do this, a machine learning framework that provides a selection of existing learning approaches and allows for implementing new algorithms has been developed by our research group [13]. This framework will be used to develop a standard benchmark suite for testing and measuring the effectiveness and accuracy of the compared mapping methods, especially for maritime cranes.

VI. Acknowledgments

This work is partly supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 and the Innovation Programme for Maritime Activities and Offshore Operations, project number 217768.

References

[1] F. Sanfilippo, L. I. Hatledal, H. G. Schaathun, K. Y. Pettersen, and H. Zhang, "A universal control architecture for maritime cranes and robots using genetic algorithms as a possible mapping approach," in *Proceeding of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China, December 2013*, 2013, pp. 322–327.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[3] P. Li and C. Wang, "Design and implementation of visual simulation system for shipborne crane control," in *Control and Decision Conference, 2009. CCDC'09. Chinese*. IEEE, 2009, pp. 5482–5486.

[4] F. Sanfilippo, H. P. Hildre, V. Æsøy, H. Zhang, and E. Pedersen, "Flexible modeling and simulation architecture for haptic control of maritime cranes and robotic arm," in *European Conference on Modelling and Simulation*, 2013, pp. 235–242.

[5] J. Wang and Y. Zhang, "Recurrent neural networks for real-time computation of inverse kinematics of redundant manipulators," *Machine intelligence: quo vadis*, pp. 299–319, 2004.

[6] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *International Joint Conference on Neural Networks*. IEEE, 2010, pp. 1–8.

[7] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[8] J. Denavit, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. of the ASME. Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.

[9] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The rprop algorithm," in *International Conference on Neural Networks*. IEEE, 1993, pp. 586–591.

[10] A. Peter. (2013, july) Efficient java matrix library. [Online]. Available: https://code.google.com/p/efficient-java-matrix-library/

[11] J. Heaton and H. Reasearch, "Encog java and dotnet neural network framework," *Heaton Research, Inc.*, vol. 20, p. 2010, 2010.

[12] U. Technologies. (2013, july) Unity3d. [Online]. Available: http://unity3d.com/

[13] L. I. Hatledal, F. Sanfilippo, and H. Zhang, "Jiop: a java intelligent optimisation and machine learning framework," in *Proceeding of the European Conference on Modeling and Simulation (ECMS 2014)*, in press.