A Universal Control Architecture for Maritime Cranes and Robots Using Genetic Algorithms as a Possible Mapping Approach

F. Sanfilippo, L. I. Hatledal, H. G. Schaathun, K. Y. Pettersen and H. Zhang

***Abstract*— This paper introduces 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 by using the same universal input device regardless of their differences in size, kinematic structure, degrees of freedom, body morphology, constraints and affordances. The manipulators that are to be controlled can be added to the system simply by defining the corresponding Denavit-Hartenberg table and their joint limits. The models can be simulated in a 3D visualisation environment, which provides the user with an intuitive visual feedback.**

The presented architecture represents the base for the research of a flexible mapping procedure between a universal input device and the manipulators to be controlled. As a case study, our first attempt of implementing such a mapping algorithm is also presented. This method is bio-inspired and it is based on the use of Genetic Algorithms (GA). Using this approach, the system is able to automatically learn the inverse kinematic properties of different models.

Related simulations were carried out to validate the efficiency of proposed architecture and mapping method.

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. In most cases, the operator has to handle an array of levers, throttles or buttons to operate the crane joint by joint. Moreover, each input device can normally control only one specific crane model. When considering working efficiency and safety, this kind of control is extremely difficult to manage and extensive experience with high control skill levels 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.

Robotic arms and cranes are very similar in the way they operate and in the way they are designed. Both have a number of links serially attached to each other by means of joints that can be moved by some type of actuator. In both systems, the end-effector of the manipulator can be moved in space and be placed in any desired location within the system's workspace and can carry a certain amount of load. However, traditional cranes are usually relatively big, stiff and heavy because they normally need to move heavy loads at low speeds, while industrial robots are ordinarily smaller, they usually move small masses and operate at relatively higher velocities. This is the reason why cranes are commonly actuated by hydraulic valves, while robotic arms are driven by servo motors, pneumatic or servo-pneumatic actuators. Most importantly, the fundamental difference between the two kinds of systems is that cranes are usually controlled by a human operator, joint by joint, using simple joysticks where each axis operates only one specific actuator, while robotic arms are commonly controlled by a central controller that controls and coordinates the actuators according to some specific algorithm. In other words, the controller of a crane is usually a human while the controller of a robotic arm is normally a computer program that is able to determine the joint values that provide a desired position or velocity for the end-effector.

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 in fact 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 this paper, a general architecture is presented 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 of doing this consists of finding a flexible way to map the fixed degrees of freedom of the universal input device to the variable degrees of freedom 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 similar. The presented architecture allows for designing and testing different mapping procedures. As a case study, our first attempt at implementing a mapping method is also presented. This method is based on the use of Genetic Algorithms (GA) [1] and using this approach, the system is able to automatically learn the inverse kinematic (IK) properties of different models.

Note that, since the main focus of this preliminary work is on building the control architecture, all problems related

This work is supported by the Innovation Programme for Maritime Activities and Offshore Operations (MAROFF) which is promoted by the Research Council of Norway.

F. Sanfilippo, L. I. Hatledal and H. Zhang are with the Department of Maritime Technology and Operations (Avdeling for Maritim teknologi og Operasjoner), Aalesund University College (Høgskolen i Aalesund), Postboks 1517, 6025 Aalesund, Norway, [fisa, laht, hozh]@hials.no.

H. G. Schaathun is with the Department of Engineering and Natural Sciences (Avdeling for ingeniør- og realfag), Aalesund University College (Høgskolen i Aalesund), Postboks 1517, 6025 Aalesund, Norway, hasc@hials.no.

K. Y. Pettersen is with Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway, kristin.y.pettersen@itk.ntnu.no.

to rope pendulations or wave impacts on the payload are not considered in this paper but they can be included in the model at a later stage.

The paper is organised as follows. In Section II, a review of the related research work is given. In Section III, we focus on the description of the system architecture and, as a case study, our first attempt of implementing a flexible mapping method is also presented. In Section IV, related simulation results are shown. In Section V, conclusions and future works are outlined.

II. Related Research Work

In existing literature, not much work has been done to overcome the low control flexibility and non-standardisation problems for the current maritime crane control architecture. Lebans et al. [2] proposed a tele-robotic controlled handling system operated by an intuitive controller. 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 these 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 degrees of freedom.

A common assumption in all these previous works is that the IK 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 degrees of freedom, the inverse kinematics can have multiple solutions, and therefore 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 solution 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 different arms and new models could also be easily added.

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 GA models have been developed by applying biologically-inspired control mechanisms to robot control tasks. Zhang et al. [5] proposed an approach for a robot inverse velocity solution using GA. The authors used the principle of robot motion propagation from link to link to find the robot's recursive velocity formula, which then was used to determine the fitness function. Tabandeh et al. [6] used a GA approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering. The niching method was used to modify the GA fitness value to encourage convergence around multiple solutions in the search space. The authors concluded that the proposed algorithm could be generalised to solve the IK problem of a robot with unknown DOF and configuration, and that the method worked with good precision and speed.

On the other hand, most of these intelligent systems are only able to learn the control of a specific crane/arm. At the moment, 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 DOF for both the input device and the model to be controlled. In other words, the controlling device has to have the same kinematic structure of the controlled model.

III. System architecture and case study

A. Architecture

Since the main focus of this work is on building a universal control architecture for different models of maritime cranes and, more generally, robotic arms, a generalised manipulator model is considered. The generalised model consists of a kinematic chain that can be controlled by setting the position or the velocity of the joints.

From a kinematic point of view, the end-effector of an offshore crane usually consists of a wire which is used to lift and transfer objects, while robotic arms are commonly equipped with more complex devices like grippers or tools. In a wider sense, in both cases, the end-effector can be seen as the part of the manipulator that interacts with the work environment and it may be modelled as part of the same kinematic chain. The proposed architecture, however, also allows the end-effector to be modelled as a distinct sub-chain that can be controlled separately. So, in general, a mapping control method may or may not consider the control of the whole manipulator. Decoupling the model of the end-effector from the model of the manipulator can, in some cases, greatly simplify the mapping control algorithm since the complexity of the system generally increases more than linearly with the number of DOF.

With simplicity in mind, in this simplified model, the nature of the crane actuators - whether they are hydraulic, pneumatic, electric or mechanical - is not considered, but it may be included in the model at a later stage.

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. With the information provided by the visual feedback, the operator has a better sensing of the

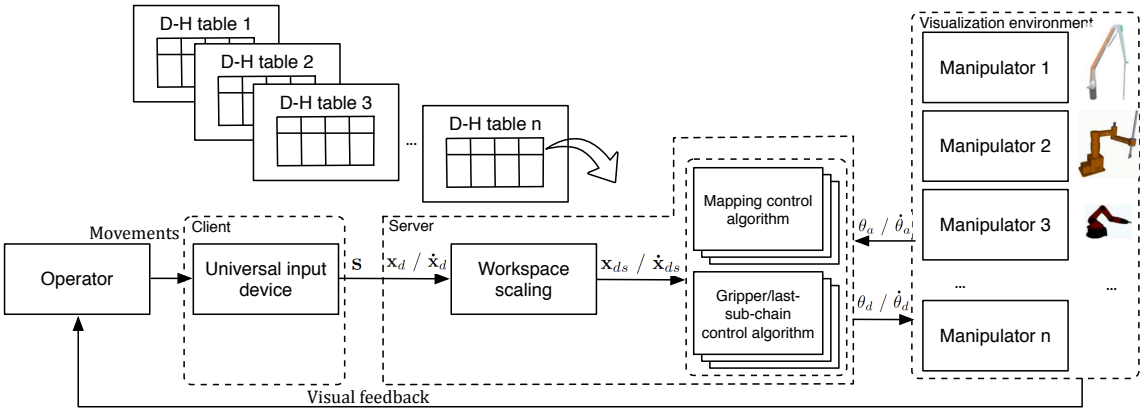


Fig. 1. The proposed control system architecture: a client-server architecture with the input device running as a client and communicating with a server where control algorithm logic is implemented.

working area and can easily drive the end-effector of the crane into the target position.

The control objective is that when the operator makes a movement such as lifting, handling, transportation or other manipulations by using the universal input device, the controlled robot should make an analogous motion. The proposed architecture provides the possibility of controlling the arms in position mode or velocity mode. The user experience is substantially different in each case. When using the position control mode, the operator simply controls the position of the tip of the crane with constant velocity; when operating in velocity control mode, the operator also sets the velocity of the end-effector by using the universal input device. In the first case, when the operator releases the input device, the tip of the crane moves back to its starting point, while in the second scenario, the crane just stops moving but it keeps the last given position.

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 operation 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. In fact, the input device and the robots to be controlled have generally very different sizes and, consequently, very different workspaces. 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:

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

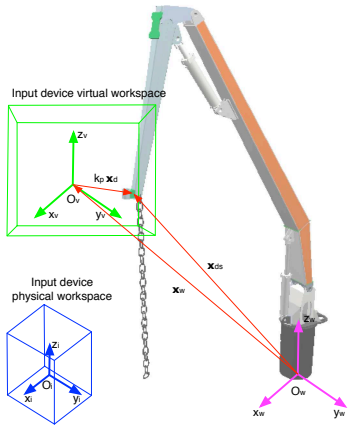


Fig. 2. The proposed architecture allows for expanding and shifting the small-scale physical workspace of the input device to a virtual expanded workspace, thus giving the robot arm the ability to make more accurate and precise movements.

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 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 inverse kinematic 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 actual joint velocities $\dot{\theta}_a$ are sent back to the server and can be used by the control mapping algorithm.

Note that the proposed architecture allow for implementing different mapping methods. Each mapping control algorithm has to realise the mapping between the fixed degrees of freedom of the universal input device and the variable degrees of freedom 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.

Another 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 [7] and their joint limits. Hence, by using the D-H method, the system is able to analytically auto-generate the forward kinematic model of the arms to be controlled. However, even if this approach provides a fundamental tool to compute the position of the end-effector from specified values for the joint values, nevertheless, a mapping procedure to determine the joint values that provide a desired position of the end-effector according to the universal input has to be developed. In particular, as a case study, our first attempt of implementing a flexible mapping method is presented in the next section.

B. Case study: a mapping method based on GA

The proposed mapping method is based on the use of a machine learning algorithm. In particular, a continuous GA is employed to automatically learn the mapping functions, (3) and (4), for the manipulators to be controlled. This approach only requires the forward kinematic model. Note 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 GA is continuously used; what differs are the semantics and the size of inputs and outputs.

The flowchart of the proposed algorithm is shown in Fig. 3. It is an iterative procedure and essentially, at each iteration, a population of candidate solutions or chromosomes is evolved toward better solutions according to a particular cost function. In the following, the key steps of the algorithm are described.

1) Define genetic representation and cost function: initially, the genetic representation, the cost function and the target cost are defined. In particular, each chromosome encodes its own properties or genes that consist of a set of joint

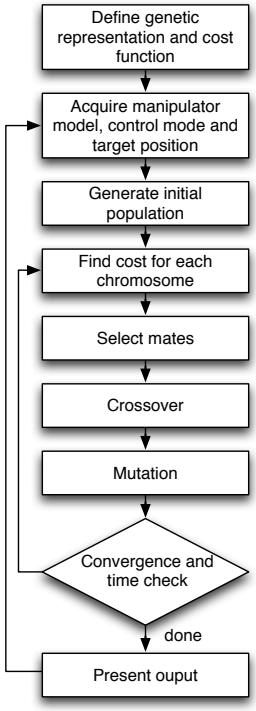


Fig. 3. Flow chart of the proposed mapping method.

angles, θ_g, constrained within their corresponding limits. The length of each chromosome is equal to the number of joints to be controlled.

The fitness of every individual in the population is evaluated by using a cost function that assesses the Euclidean distance between the target position $\mathbf{x}_t$ and the actual position $\mathbf{x}_a$:

$$d(\mathbf{x}_t, \mathbf{x}_a) = |\mathbf{x}_t - \mathbf{x}_a|, \tag{5}$$

where, the actual position is calculated by using forward kinematics, while the target position depends on the input and it is given by:

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

if operating in position control mode, or by:

$$\mathbf{x}_t = \mathbf{x}_a + \dot{\mathbf{x}}_{ds}\Delta t, \tag{7}$$

if operating in velocity control mode, where Δt is the time interval between two successive iterations.

2) Acquire manipulator model, control mode and target position: the main iteration loop starts acquiring the proper manipulator model and the control mode according to the operation scenario. Moreover, the corresponding target position is normalised according to the workspace of the manipulator to be controlled. This will result in relating the cost function to the corresponding workspace.

3) Generate initial population: subsequently, an initial population of 125 individuals is randomly generated.

4) Find cost for each chromosome: the evolution process, which is a sub-loop of the main loop iteration, starts from the initial randomly-generated population and, at each generation, the fitness of every chromosome is evaluated according

to the previously defined cost function (5). Additionally, any individual with genes that violate the corresponding joint limits gets its cost increased by a considerable factor. Then the modification process of each individual's genome starts in order to form a new generation.

5) Select mates: the selection of candidates that will be used as parents in the crossover process is obtained by using the *stochastic universal sampling method* [8], which is a fitness proportionate selection method.

6) Crossover: the crossover function is defined as a hybrid function that stochastically switches - with a 50% crossover probability - between using a single-point and a uniform crossover method, to create new offspring from the selected parent chromosomes.

7) Mutation: mutation may occur in a chromosome by stochastically adding a random value of $\pm 5\%$ to the value of its genes. In particular, there is a 0.5% mutation chance for each gene. Additionally, a form of elitism is also used and 10% of the fittest chromosomes survives unaltered between generations. Note that, using a form of elitism with 10% of the fittest chromosomes surviving between successive iterations and consecutive target positions makes sense because, since the operator executes continuous movements while operating the manipulator, consecutive input vectors, stochastically, do not differ much in terms of magnitude and direction.

8) Convergence and time check: this evolution process is repeated until a termination condition is reached. In particular, the GA population stops evolving and the fittest chromosome is returned when the cost drops below 0.01 or when the overall time spent evolving the population exceeds 100 ms. Note that, since the target position is normalised according to the workspace of the manipulator to be controlled and consequently the cost function is somehow related to the latter, there will be a correlation between the target cost and the considered model. In this way, a value of 0.01 as cost target results in being weighed and proportionate to each specific workspace. A time limit of 100 ms allows the population to reach a good level of evolution in the first few steps of iterations without effecting the operator experience in terms of perception. Moreover, after the first few iterations, the time limit is stochastically seldom reached for target positions that are located inside the boundaries of the workspace.

9) Present output: the genes of the fittest chromosome are then presented as output. In particular, denoting these genes as θ_f and according to the operation scenario, the output is obtained as:

$$\theta_d = \theta_f, \tag{8}$$

when operating in position control mode, or as:

$$\dot{\theta}_d = \frac{\theta_f - \theta_a}{\Delta t}, \tag{9}$$

when operating in velocity control mode.

Note that, 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 presented architecture. In particular, three extra input signals are used to set the *Roll-Pitch-Yaw* of the end-effector.

IV. Simulations and Experimental Results

In this preliminary study, a joystick from *Logitech*, the *Extreme 3d pro* was used as a universal input device on the client side. Each degree of freedom 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 cases, when the operator's hand is removed from the joystick, the latter returns to automatically its starting point. Note that, thanks to the modularity of the architecture, any other joystick or input device can be used without affecting the effectiveness of the system.

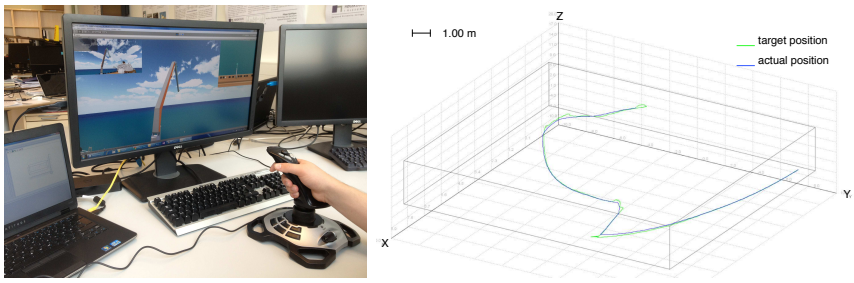
From an implementation point of view, the logic of the control architecture lies on the server side, which is implemented by using the *Java* programming language. Each manipulator to be controlled is modelled as a *Java* class which embodies a D-H table, a set of joints, a workspace as attributes and 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 GA 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 were used. In particular, the *Efficient Java Matrix Library* [9] was adopted to add support for matrix manipulations, while the Genetic Algorithm was implemented by using the *Watchmaker Framework for Evolutionary Computation* [10]. Moreover, the manipulators to be controlled can easily be added to the system by simply defining their corresponding D-H tables and their specific joint limits in a *XML* document.

Regarding the visualisation environment, in this preliminary work, the game engine *Unity3D* [11] was used to visualise the different models. However, any other visualisation environment could be used without affecting the effectiveness of the proposed architecture.

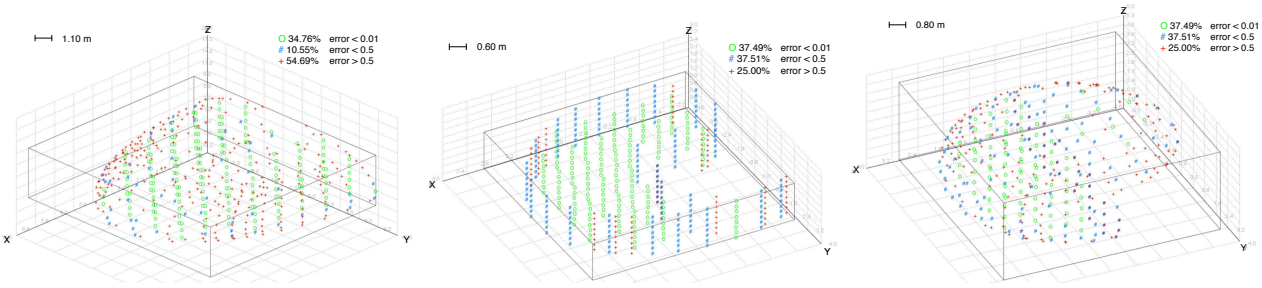
The system is based on a distributed structure and the communication between client, server and visualisation environment is realised by using the *TCP/IP* protocol. This makes it also possible to remotely control the different manipulators.

Related simulations were carried out in order to test the architecture within the particular case study of the proposed mapping method. In detail, a knuckle boom crane, which is



(a) (b)

Fig. 4. The simulation of (a) the knuckle boom crane model and (b) the trajectory tracking of its Cartesian paths in X, Y and Z coordinates.



(a) (b) (c)

Fig. 5. 3D Scatter plots showing error distribution for 512 equally-spaced target positions in the volume box that encloses the workspace of (a) the knuckle boom crane model, (b) the *SCARA* robot and (c) the *KUKA youBot* robot.

shown in Fig. 4-a, a *SCARA* robot and a *KUKA youBot* robot are modelled and simulated.

For each of these models, a trajectory tracking analysis of the Cartesian paths for X, Y and Z coordinates was performed and the results for the knuckle boom crane model are shown in Fig. 4-b. Generally, the proposed system has demonstrated a quite fast reaction to the inputs. However, this kind of test is task-dependent.

In addition, for each of these models, an error distribution analysis was performed considering a set of 512 equally-spaced target positions in the volume box that encloses their corresponding workspace. For the three models considered, these error distributions are shown in Fig. 5 as 3D scatter plots. It is probable that the target positions at the boundaries of this imaginary box are less reachable by the manipulators due to their joint constraints, which is why the corresponding errors are stochastically greater. However, even for these points, the GA is able to find the closest position match, thus avoiding potential singularity problems.

V. Conclusion and Future Work

Regarding the presented case study, the adopted cost function that is currently related to the workspace of the controlled model, could be also be related, as future work, to the particular manipulation task to be performed. In particular, the use of a task-oriented cost function could considerably improve the performance of the system by allowing for heavier weighing of selected tasks that require higher accuracy than others.

In the future, new mapping methods which also take heave compensation and swing related problems into account could also be implemented and integrated allowing for better flexibility and reliability of the proposed framework.

References

[1] 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.

[2] G. Lebans, K. Wilkie, R. Dubay, D. Crabtree, and T. Edmonds, "Telerobotic shipboard handling system," in *OCEANS'97. MTS/IEEE Conference Proceedings*, vol. 2. IEEE, 1997, pp. 1237–1241.

[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] Y.-G. Zhang, Y.-M. Huang, Y.-Z. Lin, X. Cheng, and F. Gao, "An approach for robot inverse velocity solution using genetic algorithm," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 5. IEEE, 2004, pp. 2944–2948.

[6] S. Tabandeh, C. M. Clark, and W. Melek, "A genetic algorithm approach to solve for multiple solutions of inverse kinematics using adaptive niching and clustering," *Computer Science and Software Engineering*, p. 63, 2006.

[7] 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.

[8] J. E. Baker, "Reducing bias and inefficiency in the selection algorithms," 1985.

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

[10] D. W. Dyer. (2013, july). [Online]. Available: http://watchmaker.uncommons.org/

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