VERSITA

Applied Computer Systems

doi: 10.2478/acss-2013-0009

2013/ 14

Use of Learning Methods to Improve Kinematic Models

Kintija Priedniece, Agris Nikitenko, Aleksis Liekna, Guntis Kulikovskis (*Riga Technical University, Faculty of Computer Science and Information Technology, Institute of Applied Computer Systems*).

***Abstract* – Kinematic model is the basic aspect in robot design and motion planning. Kinematic models are idealized, however there exist certain specific aspects of particular robot or environment, so that during navigation, the robot can significantly deviate from the planned trajectory. To increase the accuracy of motions, kinematic model can be improved and to achieve that the artificial intelligence methods can be used. In case of fixed base robots different approaches are used to train kinematics, at the same time, for the mobile base robots it proves to be a more complicated task. The reason is that a mobile robot can move unbound with respect to environment thus it is difficult to control the platform without deviation from the target position, which leads to inaccuracy in the position estimate. This paper presents the method meant for improvement of the accuracy of motion of differential drive platform. Genetic programming is used to obtain the wheel velocity function, from which the coefficient, which describes different factor influence on motion, is obtained. As a result, the kinematic model of a particular platform for a particular task is obtained. This method is effective because the developed kinematic model is more specific than the general one.**

Keywords – genetic programming, learning kinematic, mobile platforms

I. Introduction

Robot kinematics deals with the analytical description of displacement, speed, velocity, and acceleration without giving regard to the force that causes the motion, thus being a fundamental aspect of robot design, analysis, control, and simulation. Kinematic models are commonly used to control motion and predict behavior of a robot. Improvement of these models can reduce localization errors, ensure repeatability of motion, and help to avoid obstacles, etc. To improve these models artificial intelligence methods can be applied. It deals with imprecise, variable, dynamic, or nonlinear decision making. A branch of artificial intelligence research is machine learning, which focuses on automatic recognition of complex patterns and decision-making based on sensor data. As a result, the degree of autonomy can be increased.

There are two types of kinematic models: fixed and mobile base robots. While the fixed based robot kinematics is more or less easy to build, obtaining mobile robot kinematic models is a much more challenging task. The main reason for this is the number of parameters that are known yet difficult to calculate, in particular, in terms of practical implementation. Good examples of such parameters are: static and dynamic traction factors of a wheel on particular soil, exact position of the mass center, etc. Besides, there is no direct way to measure the robot's position and it must be integrated over time, so it leads to inaccuracies of the position estimates over time. Our aim is to provide the solution of how to apply learning in order to improve the kinematic model of a particular mobile robot.

This paper presents the method to improve kinematic model of a differential drive robotic platform with the help of supervised learning. In this way the accuracy of the existing differential drive can be improved and specified for a particular task.

The paper is structured as follows. Section II summarizes the related work in learning kinematic by using artificial intelligence methods. Section III explains the mobile robot kinematics by emphasizing the differential drive. Section IV presents the research method in detail. Section V explains the choice of the learning method for a particular task. Experiments and details related to them are described in Section VI. Finally, the results of the proposed method are discussed in Section VII.

II. Related work

Different artificial intelligence methods have been used to improve fixed base manipulator kinematic models. Neural networks have proved to be useful for learning the kinematics of robot manipulators. Some authors used multi-layer perceptron networks of self-organizing maps to train the forward kinematics model of various simple systems [1], however, forward models of serial robots can easily be computed by analytical method, therefore, in general, the evaluation is based on the accuracy of the model itself rather than on its control capabilities. Training inverse kinematic models is more difficult. New neural network architectures have been introduced to overcome the discontinuity of the inverse kinematic function of typical robot arms with joint constraints [2]. A new expert neural network based approach has been developed, where each of the experts approximates the continuous part of inverse kinematics function [3]. There is the method, which solves some specific issues in robotic neuro-controller learning [4]: it avoids any neural network learning algorithm, which relies on the classical supervised input-target learning scheme, it can converge beyond the locally optimal solutions; and can learn the neural network topology. Instead of finding a direct approximation of inverse kinematics, it is also possible to model the joint probability distribution of end-effector positions and joint angles [5]. Robot manipulator can be decomposed into two or more virtual robot arms to speed up the learning of the inverse kinematics [6]. These solutions are not applicable to mobile robots, because mobile robots are not fixed. They move within the environment thus requiring to constantly estimate

their position over time. This means that there are no fixed reference points to initiate kinematic calculations.

III. Mobile robot kinematics

When a mobile platform is analyzed, the kinematic model is assumed to be idealized, some kinematic constraints are assumed to exist [7]: movement is on a horizontal plane, wheels are not deformable, there is no slipping, skidding or sliding and no friction for rotation around contact point, etc. It is not possible to create an ideal system and the real situation always differs in some way from the predicted one. Differential drive robot was selected to analyze mobile platform because these platforms are ubiquitous in robotics. These robots use a simple drive mechanism that consists of the two drive wheels mounted on a common axis. Moreover, each wheel can be independently driven both in forward and reverse directions which is controlled by the speeds v_l and v_r set separately for the left and the right wheels. Position in the global frame is determined by three parameters (x, y, θ), where θ denotes the direction of movement. When left and right wheel rotate with different speeds, the robot rotates around the common point denoted *Instantaneous Center of Curvature (ICC)*. By varying v_r and v_l, ICC moves and causes different trajectories and motion curves of the robot. R in Fig. 1 is the signed distance from the ICC to the midpoint between the wheels, *l* – the distance between the centers of the two wheels.

There are three different motion cases that must be considered:

1. *If* $v_l=v_r$, then forward linear motion in a straight line (R→∞);
2. *if* $-v_l = v_r$, then R=0, and there is rotation about the midpoint of the wheel axis;
3. *if* $v_l=0$, then we have rotation about the left wheel. In this case R =l/2. Same is true if $v_r=0$.

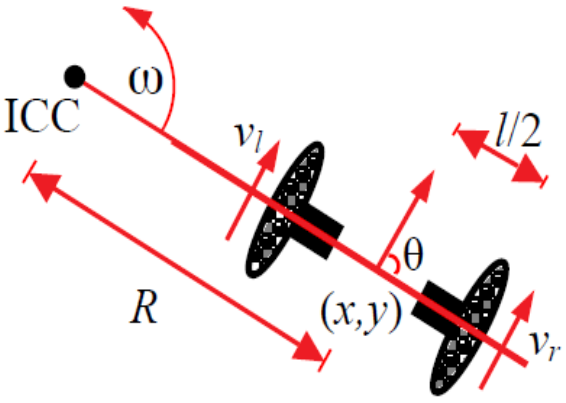

Fig.1. Differential drive

Forward kinematic solution is typically used for automatic control of a robot such that it follows the desired trajectory [8]. At certain given wheel speeds forward kinematic equations provide an updated pose. If the starting position (x, y, θ) is known, the new position (x', y', θ') can be obtained by using 3D rotation matrix:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} \cos(\omega\delta t) & -\sin(\omega\delta t) & 0 \\ \sin(\omega\delta t) & \cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x\text{-}ICC_x \\ y\text{-}ICC_y \\ \theta \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{pmatrix} \quad (1)$$

To solve inverse kinematic, there are two special cases of control:

1. $v_r=v_l$ =>$R=\infty$=> The robot moves in straight line and θ remains the same;

2. $v_r=-v_l$ =>$R=0$ => The robot rotates in place about ICC (any θ is reachable while (x,y) is unchanged).

This problem most often has no solution, in the sense that the robot cannot reach an arbitrary pose by simply setting appropriate values for wheels speeds v_l, v_r and let the motors run for a while. In real situations there are some differences in the model, therefore there is some deviation from the predicted motion.

IV. Proposed method

For differential drive there are two control parameters of the motion, yet there are various factors, which can affect performance of a robot. Vehicles of this drive are very sensitive to slight changes in velocity in each of the wheels. Small errors in relative velocities of the wheels can affect the robot trajectory significantly. They are also very sensitive to roughness of the ground plane, and may need extra wheels for support. There is a number of difficult to estimate parameters that represent different imperfections of the manufacturing process resulting in slight deviations of the wheel speeds from the set of expected values.

Therefore, we introduce two parameters - coefficients k_r, k_l (2), (3), which represents all possible deviations from the ideal, like disposition of mass center, reducer's mechanical resistance, etc.

$$v_r = k_r \cdot \omega\left(R + \frac{l}{2}\right) \quad (2)$$

$$v_l = k_l \cdot \omega\left(R - \frac{l}{2}\right) \quad (3)$$

From the expressions (2) and (3) we define our problem statement as finding coefficients k_r and k_l, which for above mentioned reasons are specific to particular robotic platform. Mathematically the task can be expressed and solved as rather typical optimization task. Unfortunately the task have no analytical solution because of the optimization criteria, which in our case is a set of experimental data specific to particular robotic platform. Therefore, we propose to use machine learning that usually has good performance in solving the problems involving experimental data. Machine learning algorithms are organized into taxonomy, based on the desired outcome of the algorithm. Common algorithm types include:

1. Supervised learning, where the algorithm generates a function that maps inputs to desired outputs;
2. Unsupervised learning, which models a set of inputs: labeled examples are not available;
3. Reinforcement learning, where the algorithm learns the policy of how to act given an observation of the world.

When the desired task and the environment for mobile robot platform are known, it is appropriate to use supervised learning, because it is possible to experimentally measure the necessary parameters and respective desired output values. Most of the supervised learning algorithms analyze the training data and produce an approximated function. The approximated function should predict the correct output value for any valid input object with the acceptable accuracy. In this manner it leads to proposed method, which is shown in Fig.2.

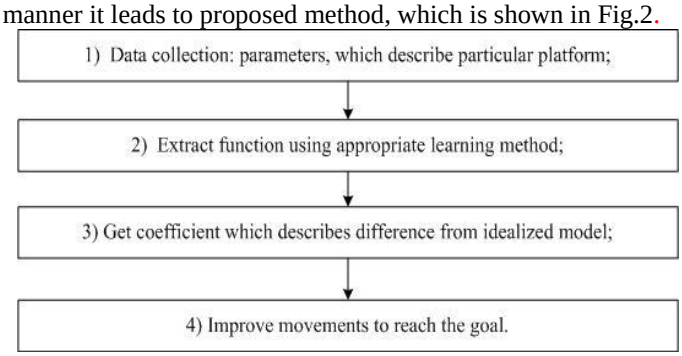

Fig.2.Proposed method.

Our particular step implementation is discussed in the following sections.

A. Data collection

Training data must contain the main parameters, which describe a robot motion. In this case, control parameters and the distance R from ICC to the midpoint between the wheels specifies the character of motion. Distance R is important, especially when linear motion needs to be achieved ($R \to \infty$). It can be done by using the global coordinate system or doing it manually, in this case use the drawing tool to get the necessary parameters. Control parameters need to be randomly chosen.

B. Function extraction using learning method

Collected data (training examples) are used to obtain regression function. Training examples are spitted into two subsets, yet there is no common way to do that, therefore it is important to vary the length and examples of subsets. In supervised learning each example is a pair consisting of an input object and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a regression function. The inferred function should predict the correct output value for any valid input object.

C. Integration of function in prediction equations for a particular mobile platform

A lot of parameters, which affect performance of a robot, are difficult to detect, thus variation from the idealized model can be expressed in the form of a coefficient, which describes different factor influence. When the function of velocity is obtained, it can be integrated into the robot control system.

D. Prediction or planning according to task

Obtained coefficients k_r (2) and k_l (3) are used to improve position estimate in the global coordinate system or robot control system. In this way, it would improve prediction and planning of motion to reach the desired goal for a particular task.

V. Learning method

It is very important to choose the appropriate learning method, which is the best to apply for a particular task. The choice was between neural networks and genetic programming, which might be used to solve regression problem [9].

A. Artificial neural networks

An artificial neural network is a mathematical model or computational model that is inspired by the structure and functional aspects of biological neural networks. A neural network is composed of a number of nodes, or units, connected by links. Fig. 3 shows a typical unit. Each link has a numeric weight associated with it $w_{j,i}$. Weights are primary means of long-term storage in neural networks, and learning usually takes place by updating the weights.

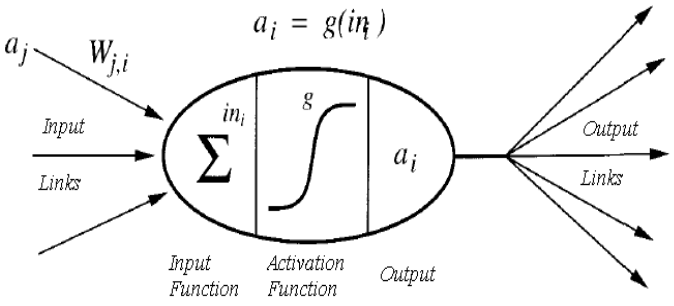

Fig.3. A unit.

The computation is split into two components. The first is the linear component that is called the input function, in_i, that computes the weighted sum of the unit input values. Second is a nonlinear component called the activation function, g, that transforms the weighted sum into the final value, which serves as the unit activation value, a_i. Usually, all units in a network use the same activation function.

Each unit has a set of input links to other units, a current activation level, and a means of computing the activation level at the next step in time, given its inputs and weights. The idea is that each unit does a local computation based on the inputs from its neighbors, but without the need for any global control over the set of units as a whole. In practice, most neural network implementations are in software and use synchronous control to update all the units in fixed sequence.

To build a neural network to perform a certain task, one must first decide how many units need to be used, what kind of units are appropriate, and how the units will be connected to form a network. One then initializes the weights of the network, and trains the weights by using the learning algorithm applied to a set of training examples for the task. The use of examples also implies that one must decide how to encode the examples in terms of inputs and outputs of the network.

Advantages:

Adaptive learning: the ability to learn how to do tasks based on the data given for training or initial experience [10]. Self-Organization: An artificial neural network can create its own organization or representation of the information it receives during the learning time. Real time operation: artificial neural network computations may be carried out in parallel, and special hardware devices are being designed and manufactured, which takes advantage of this capability. Fault tolerance via redundant information coding: partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even in case of a major network damage.

Disadvantages:

Minimizing over the fitting requires a great deal of computational effort. Individual relations between the input variables and the output variables are not developed by engineering judgment so that the model tends to become a black box or some input/output table without analytical basis. The sample size has to be large.

B. Genetic programming

Genetic programming is a powerful method for automatically generating computer programs via the process of natural selection [11]. Genetic programming is a domain-independent method that genetically breeds a population of computer programs to solve a problem. Specifically, genetic programming iteratively transforms a population of computer programs into a new generation of programs by applying analogs of naturally occurring genetic operations.

Programs are expressed in genetic programming as syntax trees rather than as lines of code as shown in Fig. 4. The tree includes nodes and links. The nodes indicate the instructions to execute (functions). The links indicate the arguments for each instruction (terminals).

In more advanced forms of genetic programming, programs can be composed of multiple components. In this case the representation used in genetic programming is set of trees (one for each component) grouped together under a special node called root. These (sub) trees are branches.

The genetic operations include mutation, crossover (sexual recombination), reproduction, gene duplication, and gene deletion. Two most common operators are mutation and crossover. Mutation involves the replacement of an arbitrarily chosen subprogram with a newly generated random subprogram. Crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With a tree-based representation, replacing a node means replacing the whole branch. This adds greater effectiveness to the crossover operator. The expressions resulting from crossover are very much different from their initial parents.

Genetic programming starts with an initial population of randomly generated computer programs composed of the functions and terminals appropriate to the problem domain. The functions may be standard arithmetic operations, standard programming operations, standard mathematical functions, etc.

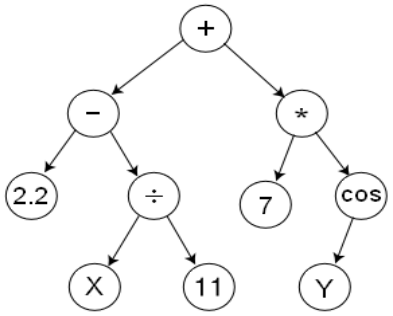

Fig.4. A function represented as a tree structure.

Each individual computer program in population is measured in terms of how well it performs in the particular problem environment. This measure is called fitness measure. The nature of the fitness measure varies with the problem. Typically, each computer program in the population is run over a number of different fitness cases so that its fitness is measured as a sum or an average over a variety of representative different situations. These fitness cases sometimes represent a sampling of different values of an independent variable or a sampling of different initial conditions of a system. For example, the fitness of an individual computer program in the population may be measured in terms of the sum of the absolute value of the differences between the output produced by the program and the correct answer to the problem.

Typically, the best individual that appeared in any generation of a run (i.e., the best-so-far individual) is designed as the result produced by genetic programming. In symbolic regression we have no a priori knowledge of the form of the solution and we expect the genetic programming system to find both the form and the details of the solution equation.

Advantages [10],[11]:

Genetic programming does not impose any fixed length of the solution. Computer programs evolved by genetic programming have variable length, so that the algorithm can find the size necessary for a solution itself. The solution of genetic programming is the equation, which can be used for different applications and it is human readable.

Disadvantages:

Genetic programs that can be constructed by the algorithm are numerous. This is one of the reasons why people thought that it would be impossible to find programs that would be good solutions to a given problem.

The process of finding a good solution can take a very long time.

Most of researchers use neural networks to improve kinematic models for fixed base robots, yet genetic programming seems to be more applicable for this application. Its main advantage is that the result is a symbolic expression, which can be integrated in the control system to improve position estimate or control parameters. At the same time,

neural network model tends to be a black box therefore more complicated to integrate or analyze .

Genetic programming does not require as much knowledge about the problem and the possible solutions as neural networks. Besides, the functions which can be used for result gathering can be selected manually. It can be more useful in the situations when the required function is with some predictable characteristics.

VI. Experiments

The goal of the experiments is to obtain specific kinematic model, which would correspond to particular platform. First of all, the training data needs to be collected and after that, the function of the wheel velocities can be obtained.

The experiments have been conducted in the indoor environment. The platform used was a simple differential drive with controllable wheel velocities. The training data contains the parameters from which the trajectory of the robot depends: including wheel velocities and radius of ICC. For getting these parameters it is important to obtain the exact trajectory of the robot, because almost always it will be a curvature (Fig. 5). The drawing tool attached to the midpoint of the wheel axis was used for draw it on the motion surface. This provides experimental trajectory with high precision.

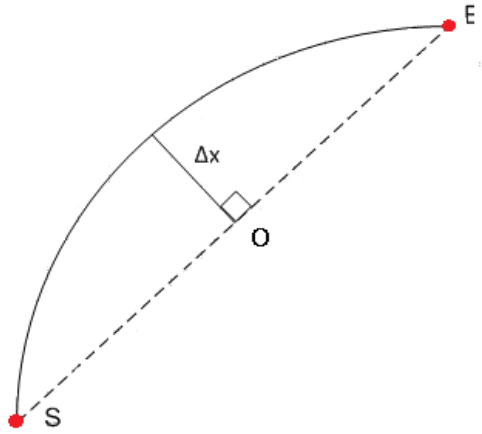

Fig.5. Trajectory of differential drive.

The motion curve radius is obtained by expression (4), where parameters from the obtained trajectory are used.

$$R = \frac{\Delta x}{2} + \frac{SE^2}{8 * \Delta x} \quad (4)$$

In these experiments one wheel velocity was constant. The speed was controlled in one direction with the pulse width modulation duty cycle (0 bit = 0%, 255 bits = 100%). To ensure precision of measurements, for each of the applied control parameters 4 experiments were made, and training data contained only mean values of them.

Experiments showed that there is a small deviation from the linear trajectory when control parameters were the same.

To obtain symbolic expression, which describes the relation of the parameters, *GPdotNET* was applied. It is a tree based genetic programming application for solving problems based on symbolic regression. This is an open source project and can be applied in various engineering problems of modeling and optimization. The project contains a C# library with Genetic programming implementation algorithms and Windows Forms application for graphical and visual presentation of the results. GPdotNET also supports parallel processing for multicore processors based on ParallelFx library [12].

The preparatory steps to apply the technique for a problem includes the selection of appropriate functions and terminals out of which programs will be constructed, the definition of a problem specific fitness function. For this model, 6 randomly generated constants in the interval from 0 to 10 were used. As one of wheel velocity was constant, the other one was used as the terminal. Predicted character of the target function has to be analyzed to choose the functions, which are appropriate for this task. It is predictable that nonlinearity is not cyclic, therefore periodical functions are not suitable for this application. The functions used here were: addition, subtraction, multiplication, division, squaring, cube, square root, exponent, module, inverse proportion. Fitness value was root mean square error. Visual representation of genetic programming function fitting training data was used to keep track of genetic programming evolution. Value difference from real and predicted function was evaluated to find out which is the best of all the obtained results.

VII. Results and Discussion

As a result, a specific kinematic model was obtained. Fig.6. shows the graph demonstrating how the radius of ICC is changing depending on the left wheel velocity. These results clearly indicate that there are differences from idealized model. In this situation the obtained coefficient is 2% of the full wheel velocity. The largest absolute prediction errors occur in situations when R→∞, however this does not affect performance of the model.

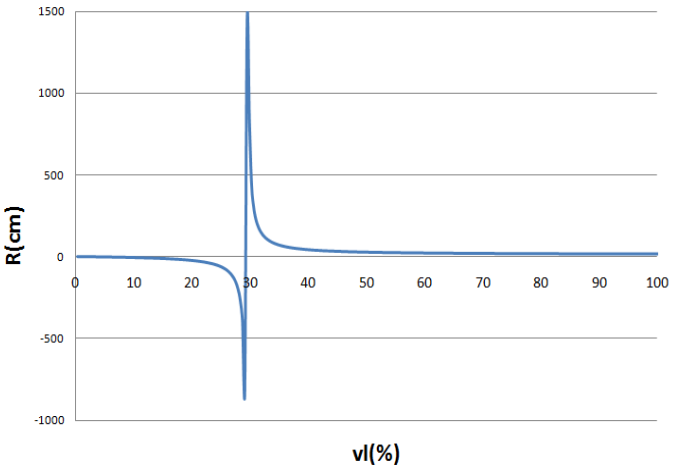

Fig.6. Graph of obtained function (vl- velocity of right wheel in %).

To evaluate precision of the obtained function, the error, which is the difference between predicted and real values, was analyzed statistically. As a result, 78.58% of all training set examples errors fall within the interval [-5.76 cm; 6.2 cm]. Predicted values, which are calculated by using the obtained equation, are widely dispersed around the mean value. This

happens because with the increase of the radius of curvature the error also increases. Precision of the test set examples is high. All examples are located in the interval [-5.16 cm; 3.58 cm].

The obtained model is very precise, because these errors do not affect the choice of control parameters for obtaining linear trajectory when the choice is based on the radius of curvature. When pulse width modulation duty cycle rises up, the changes in the radius of curvature are very small (in mm), which means that even if there are some deviations, they are very small.

Robots are made to reach some goals and usually it is done by using some plan based on some pre-generated kinematic model of the robot,. Sometimes it is impossible to detect all factors, which affect this model during its operation, therefore, it is difficult to find the way how to improve it. The proposed method proved that mobile base kinematic models can be improved with the help of artificial intelligence methods, yet one needs to be clear about task specifications and corresponding environment. The main advantage here is that it shows this model not in the common context but in a specific one. This provides the opportunities to use it in various applications.

Linear motion for idealized differential drive model can be obtained when wheel velocities are equal, thus it is the simplest way to verify precision of the model. If there are certain factors, which affect the motion, the platform trajectory will not be a straight line, and the robot would deviate from the target position. It is more advisable to use the drawing tool to estimate deviations, because if the global coordinate frame is used for this purpose, there may be situations when the trajectory is linear yet the robot deviates from the target only because of the direction of motion of its initial position.

In future we are interested in applying the learning methods to improve kinematic model of a more difficult mobile robotized platform, such as a 8-wheeled mobile platform [13].

ACKNOWLEDGEMENT

The research described in this paper is supported by funding of the ERDF Project «Development of technology for multiagent robotic intelligent system», Contract Nr. 2010/0258/2DP/2.1.1.1.0/10/APIA/VIAA/005.

REFERENCES

[1] H. Sadjadian, H.D. Taghirad, "Numerical Methods for Computing the Forward Kinematics of a Redundant Parallel Manipulator", *IEEE Conference on Mechatronics and Robotics*, Aachen, Germany,2004, pp.557-562.

[2] E. Oyama, S. Tachi „Modular neural net system for inverse kinematics learning", *Proceedings of the 2000 IEEE Int. Conf. on Robotics and Automation*, San Francisco CA, USA, 2000, pp 3239-3246.

[3] E. Oyama, N.C. Young "Inverse Kinematics Learning by Modular Architecture Neural Networks with Performance Prediction Networks", *IEEE International Conference on Robotics & Automation*, 2001, pp. 1006-1012.

[4] J. Martin, J. Lope, M. Santos "A Method to Learn the Inverse Kinematics of multi-link robots by evolving neuro-controllers", *Neurocomputing*, vol. 72 , Amsterdam, The Netherlands: Elsevier, 2009, pp. 2806-2814.

[5] B. Bosci, D. Nguen-Tuong "Learning Inverse Kinematics with Structured Prediction", *Intelligent Robots and Systems (IROS)*, 2011, pp. 698-703.

[6] V. R. de Angelo and C. Torras "Learning Inverse Kinematics: Reduced Sampling Through Decomposition into Virtual Robots", *IEEE Transactions on Systems, Man, and Cybernetics, Part B(Cybernetics)*, Vol. 38, 2008, pp. 1571-1577.

[7] G. Dutek and M. Jenkin *Computational Principles of Mobile Robotics*, Cambridge: Cambridge University Press, 2000.

[8] S. Kucuk, Z. Bingul "Robot Kinematics: Forward and Inverse Kinematics", *Industrial Robotics: Theory, Modeling and Control*, S.Cubero, Germany: Pro Literatur Verlag, 2006, pp. 117-148.

[9] J. S. Russel, P. Norwig *Artificial Intelligence: A modern approach*, Englewood Cliffs, United States of America: Prentice Hall, 1995.

[10] Y. Singh, P.K. Bhatia, O.Sangwan "A review of studies on machine learning techniques", *International Journal of Computer Science and Security, Vol.1 Issue: 1*, 2007, pp. 70-84.

[11] K.R Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

[12] GPdotNET.CodePlex, Documentation, 2010 [Online]. Available: http://gpdotnet.codeplex.com/documentation.[Accessed: May 2, 2012]

[13] A. Nikitenko, G.Kulikovskis "Eight wheel robotic platform and its Fuzzy control system", International Conference on automation, robotics and c control systems, Orlando, USA, 2010, pp. 16-23.

Kintija Priedniece has received her Bc.sc.ing degree in computer control of electrical technologies in 2010. In 2012 she received a Mg.Sc.ing. degree in intellectual robotic systems. Both degrees were acquired at Riga Technical University, Riga, Latvia.

She is working as a research assistant at Riga Technical University. Her research interests include robotic, artificial intelligence.

Agris Nikitenko has received his Dr.sc.ing. in 2006 from Riga Technical University as well as Bc.sc.ing. and Mg.sc.ing. focusing on artificial intelligence in his thesis.

Currently he is docent in the Department of Systems Theory and Design of Riga Technical University and vice dean of study affairs in the Faculty of Computer Science and Information Technology. His scientific interests cover artificial intelligence and autonomy as well as their application in robotics.

He has received award of Verner fon Siemens for his doctoral thesis in 2006. At present he is member of IEEE and ACM as well as represents Latvia in NATO RTO AVT panel.

Aleksis Liekna has received his Bc.sc.ing degree in 2008 and his Mg.sc.ing. degree in 2010 from Riga Technical University. At the moment he is a PhD student at Riga Technical University. His major field of study is computer science.

He is working as a Research Assistant at Riga Technical University. His research interests include artificial intelligence and multi-agent systems.

He was awarded by the Latvian Foundation for Education for his bachelor's thesis "Development and Implementation of Reinforcement Learning Model".

Guntis Kulikovskis has received his Mg.sc.ing. in Riga Technical University focusing on driving mechanisms of underwater vehicles. At the moment he is a PhD student in the Faculty of Transport and Mechanical Engineering.

Currently he is researcher in the Division of Autonomous Robotic Systems of Riga Technical University. His main scientific focus areas are highly mobile ground vehicles and underwater robotic systems.