Robots Collision Avoidance Using Learning through Imitation

Aurel Fratu, Jean-Paul Becar

Robots Collision Avoidance Using Learning through Imitation

Aurel Fratu
Dept. of ATI
Transilvania University of Brasov
Brasov, Romania
e-mail: fratu@unitbv.ro

Jean-Paul Becar
IUT- GEII
University of Valenciennes, LAMAV FR-CNRS 2956
Valenciennes, France
e-mail: jean-paul.becar@univ-valenciennes.fr

***Abstract*—This paper deals with the collision avoidance of the cooperative robots using the learning through imitation. Each physical robot acts fully independently, communicating with corresponding virtual prototype and imitating her behavior. Each physical robot reproduces the motion of her virtual prototype. The estimation of the collision-free actions of the virtual cooperative robots and the transfer of the virtual joint trajectories to the physical robots who imitate there virtual prototypes, are the original ideas. We tested the present strategy on several simulation scenarios; involving two virtual robots that each must cooperate with other and estimating collision-free actions. The effectiveness of the proposed strategy is discussed by theoretical considerations and illustrated by simulation of the motion of two cooperative manipulators. It is shown that the proposed collision-free strategy, while tracking the end-effector trajectory, is efficient and practical.**

Keywords— collision avoidance; virtual prototypes; cooperative tasks; learning through imitation

I. INTRODUCTION

A key requirement for cooperative efficient operation is good coordination and reciprocal collision avoidance. Cooperative robots are permanently in danger to be in collision. Therefore installations with cooperative robots in real world, require collision avoidance methods, which take into account the mutual constraints of the robots [3]. Generating motions for real or virtual robots, which are coupled to a task, is usually a complicated problem.

The contact of the robot with an obstacle must be detected and it will cause the robot to stop quickly and thereafter back off to reduce forces between the robot and environment [1]. The problem of the contact with obstacle imposes the null velocity in the moment of the impact and to obtain the zero-velocity points on the pathway. The collision detection simply determines if two geometric objects are intersecting or not. The intersecting of two objects is possible in the virtual world, where the virtual objects can be even intersected and there no exist the risk to be destroyed.

978-1-4799-2442-4/13/$31.00 ©2013 IEEE.

The contact detection between the virtual objects is responsible for determining the distance between two points in a space and substitutes the distance metrics devices [2].

This strategy may use simple CAD methods or may be extended to be more complex and take into account invalid areas of the space.

Using this strategy one detects collisions in all directions, protecting not only the physical end-effectors but also the work pieces and the physical robot itself. The intersecting or mutually penetrating of two objects is possible in the virtual world, where the intersecting of virtual objects is possible, and where there is no risk of destruction.

The ability of predicting the behaviour of cooperative manipulators is important for several reasons: for example in design, designers want to know whether the manipulator will be able to perform a certain typical task in a given time frame; in creating feedback control schemes, where stability is a major problem, the control engineer cannot risk a valuable piece of equipment by exposing it to untested control strategies. Therefore, a facile strategy for collision avoidance, capable of predicting the behaviour of a robotic manipulator or of the system at whole becomes imperative.

In the real world, like collision detection, where robots need to interact with their surrounding, it is important that the computer can simulate the interactions of the cooperative participants with the passive or active changing environment, using virtual prototyping.

When the robots need to interact with their surrounding, it is important that the computer can simulate the interactions of the cooperative participants, with the passive or active changing environment in the graphics field, using virtual prototyping. In this paper, one propose a fast method that simultaneously determines actions for two virtual robots that each must cooperate with other. The actions for the cooperative tasks are computed for each virtual robot and are transferred, with a central coordination to corresponding physical robot which must imitate her virtual homonym. Thus, one proves that our method guarantees the collision-free motion for each of the cooperative robots.

The planning package communicates primarily with simulation. A planning module can send messages to the simulation such as computed plans for the robots. The planning module can further send trajectory and planning structure information to visualization so users can see the results of an algorithm. The planning module also receives control signals from the simulation module, such as when to start planning joint trajectories.

The visualization module is responsible for visualizing any aspect needed by the programmer. Users interact with the simulation environment through the visualization [4]. This includes, but not limited to, computer screen and cameras video. The visualization provides an interface to develop interactive implementations based on imitation strategy. Optimization of the real robots behavior is performed in the low dimensional virtual space using the virtual robot prototypes. This paper is focused on the collision avoidance through transfer the motion mapping from virtual space, in 3-D dimensional real space.

II. Overview of Learning by Imitation

Imitation is an important learning mechanism in many intelligent systems including robots. It is easy to recuperate kinematic information from virtual robot motion, using for example motion capture. Imitating the motion with stable robot dynamics is a challenging research problem [8, 10].

In this framework a physical robot arm can be a collection of rigid bodies, subject to the influence of various forces in the workspace, and restricted by various motion constraints.

One propose a control strategy for two physical cooperative robots that uses capture data from their virtual prototypes and imitate them to track the motion in the real space avoiding the collision. One present an model, for which robust controller can be easily designed. A typical example is a linear quadratic regulator (LQR) [5, 7], which one use for our examples.

Joint trajectory tracking is enabled by commanding desired joint accelerations based on joint angle and velocity errors as well as supply forward joint accelerations. One applies the controller to tracking motion capture clips of two cooperative robots who accomplish a collaborative task. The resulting robot motion clearly preserves the original behavior of each virtual robot.

In addition, the controller does not require intensive pre-processing of motion capture data, which makes it potentially applicable to real time applications.

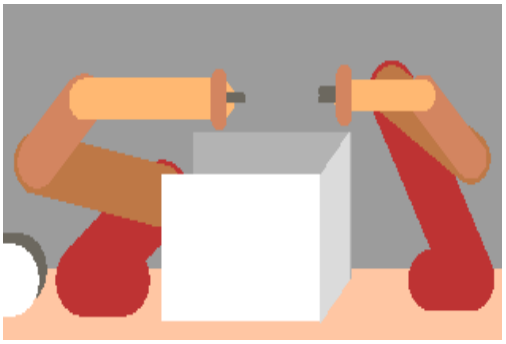
A characteristic feature of robot programming is that usually it is dealing with two different worlds; the real physical world to be manipulated, and the abstract models representing this world in a functional or descriptive manner by programs and data. The basic idea behind these approaches is to relieve the programmer from knowing all specific robot details and free him from coding every small motion /action.

Rather, he is specifying his application on a high abstraction level, telling the robot in an intuitive way what has to be done and not how this has to be done.

In this paper, one propose an approach to achieving pathway acquisition in robots programming, using imitation strategy. The framework for our method is shown in Fig. 1. First, a motion capture system transforms Cartesian position of virtual robot structure to virtual joint angles based on kinematic model.

Then, the joint angles are converted in binary words and transferred to real robot joint controllers via intelligent interface. After this one use the control loops structure to establish relationships between the virtual and real robot control systems. One employs dimensionality reduction to represent posture information in a virtual low-dimensional space [6,]. One presents result demonstrating that the proposed approach allows a real robot to learn move, based exclusively on virtual robot motion capture without the need for a detailed physical model of the robot.

In this paper one use the virtual robot prototypes and the motion capture systems to obtain the reference motion data, which typically consist of a set of trajectories in the Cartesian space.



a)

b)

Fig. 1. A framework for robot learning by imitation.
Virtual cooperative robots (a) - Real cooperative robots (b)

The paper is focused on tracking joint angle trajectories, although some cooperative tasks may require tracking other quantities such as end-effectors trajectories which will be addressed in future work.

III. Processing of the Motion Captured Data

The robots' motion should be animated with the highest degree of realism possible using motion capture data or accurate full-body simulation, while the multitudes secondary

details to the auxiliary elements (scene, cameras, etc.) can be simulated at much lower fidelity.

One transfers, via intelligent interface, the joint angle data from a motion capture system to a kinematic model for an anthropomorphic robot. To generate the desired motion sequence for the real robot, one captures the motions from a virtual robot model and maps these to the joint settings of the physical robot.

Initially, a set of virtual postures is created to the virtual robot and the pictures' positions are recorded for each posture, during motion. These recorded pictures' positions provide a set of Cartesian points in the 3D capture volume for each posture.

To obtain the final real robot posture, the virtual pictures' positions are assigned as positional constraints on the physical robot. To derive the joint angles one use standard inverse kinematics (IK) routines. The IK routine then directly generates the desired joint angles on the robot for each posture.

The data is obtained using a motion capture channel taking into account the joint motion range [2]. Due to the joint limits and the difference between the kinematics of the virtual and real robot, the joint angle data are pre-processed. In the pre-processing stage, one assumes that both virtual and physical robots are on the scene at the same time and estimate the correct arms position and orientation.

One then compute the inverse kinematics for new posture to obtain the cleaned joint angles and retain the difference from reference joint angles. At each frame during control, one adds the difference to the reference data to obtain the cleaned reference joint angles. This correction is extremely simple and the controller does not require supplementary cleanup.

IV. CONTROLLER COMPOSITION

Fig. 2 shows the overview of the controller. The two main components are a situation controller and a tracking controller.

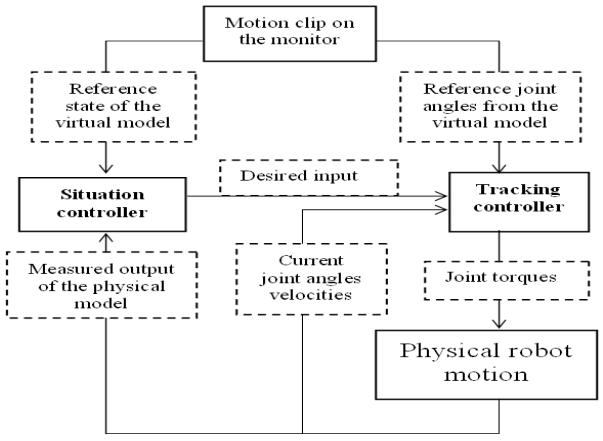


Fig. 2. Overview of the controllers

The situation controller is responsible for keeping the whole physical structure in equilibrium, usually using a controller designed for a simplified dynamics model, such as Linear Quadratic Regulator (LQR). The output of the situation controller is the desired input to the track controller.

The tracking controller is responsible for making every joint track the desired trajectory. It solves an optimization problem that respects both joint tracking and desired inputs to the simplified model and obtains the joint torques to be commanded to the real robot.

V. SIMPLIFIED VIRTUAL MODEL

The basic essence of this framework is to describe each rigid object in the planning scene as a dynamical system, which is characterized by its state variables (i.e. position, orientation, linear and angular velocity).

One supposes that the simplified virtual model is linear and represented by the following state-space differential equation:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u, \tag{1}$$

$$y = \boldsymbol{C}\boldsymbol{x}, \tag{2}$$

where $\boldsymbol{x}$ is the state vector, u is the input, and y is the output of the simplified virtual model. Also one assumes that is used a state feedback controller for equilibrium:

$$u = \boldsymbol{K}(\boldsymbol{x}_{ref} - \boldsymbol{x}), \tag{3}$$

where $\boldsymbol{K}$ is a constant gain matrix and $\boldsymbol{x}_{\text{ref}}$ is a reference state, typically computed from the reference motion.

An observer compares the estimated and actual outputs to update the state estimate $\hat{\boldsymbol{x}}$ as:

$$\dot{\hat{\boldsymbol{x}}} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{B}u + \boldsymbol{F}(\hat{y} - y), \tag{4}$$

where $\boldsymbol{F}$ is the observer gain and $\hat{y} = \boldsymbol{C}\hat{\boldsymbol{x}}$ is the estimated output. Because do not have access to real state, one replaces the state $\boldsymbol{x}$ with its estimate $\hat{\boldsymbol{x}}$ in Eq. (3):

$$u = \boldsymbol{K}(\boldsymbol{x}_{ref} - \hat{\boldsymbol{x}}). \tag{5}$$

Using Eqs. (1), (2), (4) and (5), one obtains the following system of the estimated state and new input:

$$\begin{aligned} \dot{\hat{\boldsymbol{x}}} &= \boldsymbol{A}_b\hat{\boldsymbol{x}} + \boldsymbol{B}_b u_b \\ u_b &= (\boldsymbol{x}_{ref}^T y^T)^T \end{aligned} \tag{6}$$

where:

$$\boldsymbol{A}_b = \boldsymbol{A} - \boldsymbol{B}\boldsymbol{K} - \boldsymbol{F}\boldsymbol{C}$$

$$\boldsymbol{B}_b = \boldsymbol{B} - \boldsymbol{F}.$$

Equation (6) describes how to estimate the current state of the simplified virtual model based on a reference state and measured output.

The estimated state and input to the simplified virtual model computed by Eq. (5) will be used as the input to the tracking controller. As well an inverse mapping from the real joint space, back to the original virtual joint space is then used to generate optimized motion.

VI. JOINT CONTROLLERS

The articulated robot configuration is uniquely defined by the generalized coordinate $\boldsymbol{q}$. One also denotes the generalized force by $\boldsymbol{\tau}_j$.

One supposes that virtual robots usually move with some of their links in contact with the virtual objects, in potentially state of collision in the virtual space. More of that, one can intersect any virtual structure without risk to be destroyed.

Let N_c denote the number of links which will be in contact. One represents the linear and angular velocities of the i-th contact link by a 6-dimensional vector $\dot{\boldsymbol{X}}_{ci}$. The relationship between the generalized velocity $\dot{\boldsymbol{q}}$ and $\dot{\boldsymbol{X}}_{ci}$ is written as:

$$\dot{\boldsymbol{X}}_{ci} = \boldsymbol{J}_{ci}\dot{\boldsymbol{q}} \tag{7}$$

where $\boldsymbol{J}_{ci}$ is the Jacobian matrix of the i-th contact link's position and orientation with respect to the generalized coordinates.

Differentiating Eq. (7), one obtains the relationship of the accelerations:

$$\ddot{\boldsymbol{X}}_{ci} = \boldsymbol{J}_{ci}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_{ci}\dot{\boldsymbol{q}} \tag{8}$$

One defines the compound contact Jacobian matrix $\boldsymbol{J}_c$, by:

$$\boldsymbol{J}_c = \begin{pmatrix} \boldsymbol{J}_{c1} \\ \boldsymbol{J}_{c2} \\ \boldsymbol{J}_{c3} \\ \\ \boldsymbol{J}_{cNc} \end{pmatrix} \tag{9}$$

Because the source joint is not actuated, one may only control the joint generalized force vector of the physical robot. In addition, each of the N_c links in contact receives contact force $\boldsymbol{f}_{ci}$ and moment, around the link local frame n_{ci} (i = 1, 2, …, N_c).

One also defines the compound contact force / moment vector by:

$$\boldsymbol{f}_c = \boldsymbol{f}_{c1}^T \; n_{c1}^T \quad \cdots \quad \boldsymbol{f}_{cNc}^T \; n_{cNc}^T \tag{10}$$

The equation of motion of the robot is written as:

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{G} = \boldsymbol{N}^T\boldsymbol{\tau}_j + \boldsymbol{J}_c^T\boldsymbol{f}_c \tag{11}$$

where $\boldsymbol{M}$ is the joint-space inertia matrix and $\boldsymbol{G}$ is the sum of Coriolis, centrifugal and gravity forces. Matrix $\boldsymbol{N}$ is used to map the joint torques into the generalized forces.

One can use any simplified model as long as it represents the dynamics of the virtual robot and a situation controller can be designed. A typical example is a linear system, for which a regulator can be easily designed by optimal control.

The local controllers compute the desired accelerations of joint and contact links based on the reference and current position and velocity as well as the reference accelerations [5].

In the joint controller, the desired acceleration $\hat{\ddot{\boldsymbol{q}}}$ is computed as follows at each joint:

$$\hat{\ddot{\boldsymbol{q}}} = \ddot{\boldsymbol{q}}_{ref} + k_d(\dot{\boldsymbol{q}}_{ref} - \dot{\boldsymbol{q}}) + k_p(\boldsymbol{q}_{ref} - \boldsymbol{q}) \tag{12}$$

where $\boldsymbol{q}$ is the current joint position, $\boldsymbol{q}_{ref}$ is the reference joint position in the captured data, and k_p and k_d are constant position and velocity gains that may be different for each joint.

One assumes that the position and orientation of the virtual joints is available by computing the kinematics. One can therefore compute the desired linear and angular accelerations of the virtual joints, and combine them with all desired joint accelerations to form the desired acceleration vector $\hat{\ddot{\boldsymbol{q}}}$.

Control law (12) is the same as the one used in resolved acceleration control except that the real robot joint is not actuated and the desired acceleration may be altered by the optimization stage.

The models proposed arise naturally and provide a means of verifying the plausibility of the motion in the real environment. With further work it should be possible to experimentally obtain more accurate robot dynamically models who require finding good imitation [11].

VI. ARCHITECTURE FOR COMPOSING CONTROLLERS AND PLANNERS

The general architecture includes simulation, visualization, and planning modules. The simulation module is the primary location for the development and testing of new controllers, and contains a set of features which are useful for developers. The visualization module provides an interface to develop alternative implementations.

The task planner contains multiple motion planners. The task planner also contains a world model and communicates with the simulator. The task planners are responsible for defining the objective of the planning process, while the motion planners actually generate the plan and allows planning until it has computed a path to a specified goal.

The motion planners are the individual motion planning algorithms which compute controls. The planning system is divided among several modules in order to accomplish these

tasks. The task coordination module contains motion planners for generating sequences of controls.

The planning package communicates primarily with simulation. A planning module can send messages to the simulation such as computed plans for the robots. The planning module can further send trajectory and planning structure information to visualization, so users can see the results of an algorithm. Task planner follows the planed motion in the simulated model.

World models can be used to reduce dimensions of the state space, from the motion planners. It is useful for planning because the planning process has complexity which depends on the dimensionality of the space. This reduction will make the planning process more efficient, and is related to being able to remove some systems from collision checking. These two functions together allow a full simulation to be loaded, while allowing the planning of the robots individually in a decoupled manner, for greater efficiency.

All of the abstract notions described in this section interact according to Fig. 3.

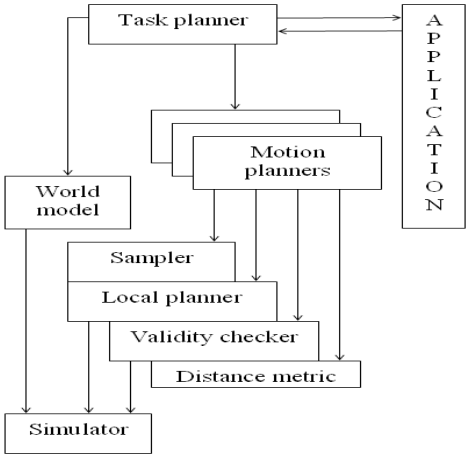


Fig. 3. The general structure of the planning modules

The collision checker describes which pairs of geometries objects should be interacting in the simulation. The collision checker is actually responsible for performing the checks and reporting when geometries have come into contact.

Validity checkers provide a way to determine if a given state is valid. The most basic implementation of a validity checker takes a state, translates it into its geometrical configuration, and checks if there is a collision between the geometry of the robots and the environment [9].

Samplers are able to generate samples within the bounds of an abstract space. Distance metrics are responsible for determining the distance of points in a space. These modules may use simple interpolating methods or may be extended to be more complex and take into account invalid areas of the space.

Users interact with the simulation environment through the visualization. This includes, but is not limited to, camera interaction and computer screen. The visualization provides an interface to develop alternative implementations.

The approach was implemented using a simulation software platform called Robot Motion Imitation Platform (RMIP) developed at University of Brasov [11], who's precision has been demonstrated in some simulation settings for cooperative work, using Virtual Reality.

The proposed architecture, composed of several modules, has separate nodes launched as executables which communicate via message passing and are organized into packages, stacks, and nodes.

A package is a collection of files, while a stack is a collection of such packages. Nodes are executables. RMIP also allows developers to integrate additional plug-in into the architecture. There are three packages which run as nodes: the simulation, planning, and visualization packages.

Motion planners are responsible for coming up with trajectories for individual robots or for groups of robots. Motion planners employ the use of modules, used as a black box by the motion planner, but may have a wide variety of underlying implementations available to accomplish the task. Furthermore, because of the flexibility of the system class, planners can plan over controllers as well. This allows for interesting new planning opportunities,

This current method only deals with static data sets. Even if this is not an issue for off-line computations, it prevents many uses in real-world applications since very small time steps are required to ensure stability.

For real-world applications various ways to overcome this problem have been used. Task planners are responsible for coordinating the high-level planning efforts of the joints. Task planners contain at least one instance of a motion planner and use planners to accomplish a given task.

Ultimately, the goal of planning is to come up with valid trajectories for one or many systems, which bring them from some initial state to a goal state, but the task planner may be attempting to accomplish a higher-level task such as motion coordination. In this sense, the task planners are responsible for defining the objective of the planning process, while the motion planners actually generate the plan.

Local planners program the robots according to their dynamics. RMIP offers two basic types of local planners, an approach local planner which uses a basic approach technique to extend trajectories toward desired states, and a kinematic local planner which connects two states exactly, but only works for robots which have a two-point boundary problem solver and kinematic agents.

Validity checkers provide a way to determine if a given state is valid. The most basic implementation of a validity checker, which is provided with RMIP simply takes a state, translates it into its geometrical configuration, and checks if there is a collision between the geometry of the robots and the environment. The RMIP is an architecture that provides libraries and tools to help software developers in programming. RMIP is focused on 3D simulation of the dynamics systems

and on a control and planning interface that provides primitives for motion planning by imitation.

Also is a object-oriented infrastructure for the integration of controllers, as well as the integration of planners with controllers to achieve feedback based planning, In particular, RMIP is used to provide concrete implementations of motion planners. The proposed infrastructure, however, allows the definition of more complex problems such as planning among moving obstacles and collision avoidance in the cooperative tasks. The joint kinematics and inertial parameters are derived from the CAD model.

The second contribution of this work involves the development of a platform for composing and evaluating controllers and motion planners. This platform, called RMIP utilizes a framework which allows both high level and low level controllers to be composed in a way that provides complex interactions between many robots.

The experiments were organized in the following manner. First, experiments were conducted using a single robot moving in an environment with obstacles. Then, experiments with two cooperative robots were run both in an obstacle-free world and in an environment with moving obstacles.

VII. CONCLUSION

Collision detection strategy is based on identifying the zero-velocity impact points, between virtual objects, in the moment of the impact. The null velocity in the moment of the impact requires a highly accurate model of robot dynamics and the environment in order to achieve the collision avoidance.

So, the problem of the contact detection is better analyzed on the virtual prototypes in the virtual environment where one predict there behavior. The problem of the contact detection in the virtual environment on the virtual robots is important for the reason that this built-in function is proven superior to other collision detection devices. The contact of two objects is possible in the virtual world, where the virtual objects can be intersected and no exist the risk to be destroyed.

Our vision of this dynamic simulator is the ability to simulate small mechanical parts of a robot arm. It reduces the frequency of the checks significantly, so as to help speed up the calculations. We revealed the potential of the Reciprocal Velocity Obstacle approach by applying it to scenarios in which two virtual robots accomplish their tasks, independently or in cooperation, in a complex environment. We would like to extend the current method, allowing it to handle various types of time-varying data sets used in animation process. Furthermore, we would like to apply our collision detection framework to several applications including the motion planning of physical robots while passing near each other.

In our formulation, the real robots must have exactly the same dynamics model as virtual robots in order to be able to imitate the behaviour of the latter Virtual robots could be handled using an abstraction of the dynamics model of their real homologue. Real robots will imitate the virtual robot's behaviour and will move according to it.

Programming approach such as learning by imitation is more flexible and can adapt to environmental change. This method is typically directly applicable to cooperative robots due the possibility to transfer the virtual joint trajectories from virtual space to the real space of the physical robots. The experimental results were obtained from evaluating the controller in a variety of scenarios concerning the reciprocal collision avoidance.

Programming real robots, especially to perform the behavior of the virtual robots is accomplished by imitation, using virtual robots motion data capture. The real (physical) robot will imitate her virtual prototype; it has no supplementary devices for collision avoidance, and gives it higher reliability and more cost efficiency. Also, since there is no device attached to the real robot tool, one no extends the tool offset distance, which allows bigger maximum tool weight and better reorientation performance.

Actually, the method and installation described in [11] are currently under testing to be eventually integrated into a real collision environment, developed at Transilvania University of Brasov. The authors expect fully automated robot programming by imitation based on this method, using robust enough system to be applied in industrial applications, will not become true before the end of this decade.

REFERENCES

[1] Weinstein, R., Teran, J., Fedkiw, R. (2005), "Dynamic simulation of articulated rigid bodies with contact and collision," in Journal IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 3, p. 365-374

[2] Zordan, V., Hodgins, J. (2002), " Motion Capture-Driven Simulations that Hit and React," in Proceedings of ACM SIGGRAPH Symposium on Computer Animation, San Antonio, TX, July 2002, p. 89-96

[3] Silver, D. (2005), "Cooperative path finding," The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'05), p. 23-28

[4] Pettre, J., Ondrej, J., Olivier, A.-H., Cretual, A., Donikian, S.," Experiment based modeling, simulation and validation of interactions between virtual walkers," Symposium on Computer Animation, Association for Computing Machinery (ACM), 2009

[5] Reist, P., Tedrake, R. (2010), " Simulation-based LQR-trees with input and state constraints," in Proceeding of the IEEE International Conference on Robotics and Automation (ICRA), p. 5504 -5510

[6] Gold, K. (2009), "An information pipeline model of human-robot interaction," in Proceedings of the 4th ACM/IEEE international conference on Human robot interaction, ISBN 978-1-60558-404-1, p. 85-92, New York, NY, USA, doi.acm.org / 10.1145/1514095

[7] Powers, A., Kiesler, S., Fussell, S., Torrey, C. (2007), "Comparing a computer agent with a humanoid robot," in Proceedings of the ACM/IEEE international conference on Human-robot interaction (HRI '07), ACM, New York, NY, USA, p. 145-152

[8] Price, B., Boutilier, C. (2003), "Accelerating reinforcement learning through implicit imitation," in Journal of Artificial Intelligence Research, vol. 19, p. 569-629

[9] Cheng, F. S., "The Method of Recovering TCP Positions in Industrial Robot Production Programs," in Proceedings of 2007 IEEE International Conference on Mechatronics and Automation, August 2007, p. 805-810.

[10] Asfour, T. ,Azad, P, Gyarfas, F., "Dillmann, R., Imitation learning of dual-arm manipulation tasks," in International Journal of Humanoid Robotics, 2008, vol. 5(2), p.183-202

[11] Fratu, A., "Method and installation for joints trajectory planning of a physical robot arm" (Proposal patent: A 00482 / 28. 06. 2013) unpublished.