Bulletin of the *Transilvania* University of Braşov
Series I: Engineering Sciences • Vol. 6 (55) No. 2 - 2013

PHYSICAL ROBOTS PROGRAMMING BY IMITATION USING VIRTUAL ROBOT PROTOTYPES

A. FRATU[1] M. FRATU[2]

Abstract: *This paper deals with the programming through imitation. In this paper the author give a brief overview of a general programming concept, describing the primary tasks which a robot control system needs to implement. Based on original idea the author proposes a new strategy to robot programming, using virtual robot prototypes. 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. 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.*

Key words: *virtual prototype, behavioral simulation, programming by imitation, desired path.*

1. Introduction

The development of robot programming concepts is almost as old as the development of robot manipulators itself.

Creating accurate robot path points for a robot application is an important programming task. It requires a robot programmer to have the knowledge of the robot's reference frames, positions, software operations, and the actual programming language.

In the conventional "lead-through" method, the robot programmer uses the robot teach pendant accessory to position the robot joints and end-effector and record the satisfied robot pose as a robot situation.

Although the programmer's visual observations can make the taught robot path points accurate, the required teaching task has to be conducted with the real robot online and the taught path points can be inaccurate if the positions of the robot's end-effector and work piece are slightly changed during the robot operations.

Today's robot simulation software provides the robot programmer with the functions of creating virtual robot path points in an interactive and virtual 3D design environment [1].

By the time a robot simulation design is completed, the simulation robot program is able to move the virtual robot and end-effector to all desired virtual robot path points for performing the specified operations to the virtual work-piece without collisions in the simulated work-cell. However, because of the inevitable dimensional differences of the components between the real robot work-cell and the

[1] Dept. of Automatics and Information Technology, *Transilvania* University of Braşov.
[2] Dept. of Installations for Constructions, *Transilvania* University of Braşov.

simulated robot work-cell, the virtual robot path points, created in the simulated work-cell, must be adjusted relative to the actual position of the components in the real robot work-cell, before they can be transferred to the real robot system. This task involves the techniques of calibrating the position coordinates of the simulation device models with respect to the user-defined real robot path points.

2. Overview of Learning/Programming by Imitation

Imitation is a learning mechanism in many intelligent systems including robots. It is easy to recuperate kinematics information from virtual robot motion using for example motion capture. Imitating the motion with stable robot dynamics is a challenging research problem [5].

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

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.

In the simplest case, these models are pure imagination of the programmers; in high level programming languages, e.g. it may consist of CAD 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. This concept is illustrated in Figure 1.

Automatic programming systems provide little or no direct control over the program code the robot will run.

Instead, robot code is generated from information entered into the system in a variety of indirect ways. Often a robot system must be running while automatic programming is performed, and these systems have been referred to as ”online” programming systems. However, automatic programming may also be performed on simulated or virtual robots, for example in industrial robotic CAD systems. In this case the real robot is offline but the virtual robot is online.

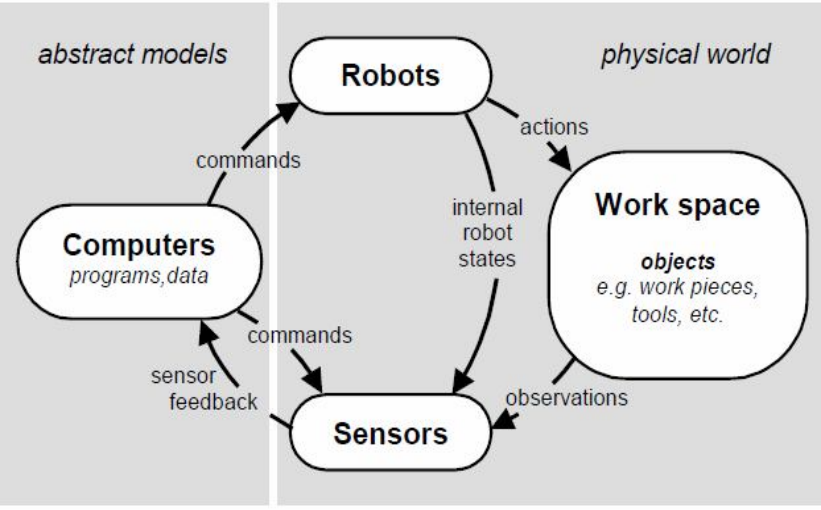

Fig.1. *General robots programming paradigm*

In any case, commands based on some model are causing robots to change the state of the real model as well as the virtual model itself. During a sequence of actions both models have to be kept consistent to each other.

This implicit programming concept implies many complex modules leading to automated robot programming. For example, there is a need for user-friendly human interfaces for specifying robot applications; this may range from graphical specifications/annotations within a CAD environment, till to spoken commands or gestures, interpreted by some speech understanding or vision system respectively.

These commands have to be converted automatically into a sequence of actions/motions by a task planning system. 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 [4].

Robot learning by imitation technique is based on motion-oriented robot programming languages. The motion-oriented robot programming languages nowadays are indispensable in robot applications; in research they often constitute the basis of higher level robot programming concepts [5].

One of the essential ingredients of modern robot programming languages is the thorough usage of the frame concept. Explicitly, all robot's poses and object locations as well as motions are expressed in accordance with human spatial intuition in terms of Cartesian coordinates.

By using homogeneous coordinates, translations and rotations can be computed by multiplying points or coordinate systems in 3D Euclidian space with one single (4 x 4) transform matrix.

As a matter of course, languages using the frame concept should supply programmers with a multitude of built-in functions to specify such transforms.

A motion-oriented robot programming language provides many functions to convert *x, y, z* - coordinates and/or Euler angles into transform matrices and vice versa.

Operator overloading and robotics specific math-functions allow a simple notation of transform matrix equations etc.

For the application programmer, who at least in the industrial world usually is not an expert of robotics, the details of the robot hardware have to be hidden behind well-defined easy-to-use software interfaces.

The robot programming implications of this, led to the development of the so-called programming by imitation concept.

Learning by imitation programs, closely communicate with the users' applications and the motion channels of robot control systems.

Explicitly, the paths of the virtual robot are reading in specified time intervals. In dependency of the virtual joints variables values, the real joint variables are modified, generating the robots' paths "guided motion" [2].

Usually, the virtual model requires from the robot programmer an in-depth understanding of the robots' functionality. Thus, it is very important to supply programmers with powerful programming language constructs to relieve such difficult tasks.

The programmer simply defines "start" and "goal" positions. After moving the robot to the "start" position in path imitating mode, it is moved in Cartesian interpolation mode to the "goal" position while an "imitating program" has been activated.

3. Creating Robot Reference Path Points through Robot Simulation Technique

With the today's robot simulation technology a robot programmer may also utilize the robot simulation software to program the motions and actions of a real robot offline, in a virtual and interactive 3D design environment.

Many robot simulation software packages provides the robot programmers with the most comprehensive and generic simulation functions, robot models, CAD data translators, and robot program translators [7].

A robot simulation design starts with building the 3D robot device models based on the geometry, joints, kinematics of the corresponding real devices such as a robot and its peripheral equipment.

The base frame $B[i](x, y, z)$ of a retrieved device defines its position in the simulation work-cell. With all required devices in the work-cell, the robot programmer is able to create virtual robot path points, called tag points and program the desired motions and actions of the robot device and end-effector device in robot simulation language.

The device simulation program allows the robot programmer to verify the performance of the robot device in the work-cell. The robot behavioral simulation in the virtual environment enables us to predict the behavior of a given real manipulator into real environment [8], [9].

After the tag points are adjusted relative to the position of the corresponding virtual robot, the robot path planner program will command the real robot controller for task execution in the real robot work-cell.

Comparing to the conventional online robot programming, the true robot offline programming by imitation provides several advantages in terms of the improved robot work-cell performance and reduced robot downtime [10].

For robot learning by imitation an intuitive and easy to use software tools are necessary. The designer engineer must know to create the virtual robot and virtual work-cell prototypes.

The Figure 2 shows a programming by imitation system, with face - to - face virtual and corresponding real robot system.

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.

Joint trajectory tracking is enabled by commanding desired joint accelerations based on joint angle errors. The resulting real robot motion clearly preserves the original behavior of the virtual robot. This task involves the techniques of calibrating the position coordinates of the simulation device models with respect to the user-defined real robot points, before they can be transferred to the real robot system.

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

Using imitation strategy, one proposes to achieve pathway acquisition from virtual world. 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 closed loops structure to establish relationships between the virtual and real robot control systems.

The imitate program will transfer the virtual joint variables values in the real world, which are used as reference values in the individuals closed control loop. Based on the position error, the control system compute the actuators' torques to continuously modify the real joints values between "start" and "goal", such the real joints pursuit the virtual joints. In similar ways any functional dependencies of some motion properties (speed, distance etc.) can be specified in a textual programming manner. Unfortunately, up to now there is a lack of off-line tools supporting robot programmers to specify robot path properties comfortably.

4. Accuracy Improvement of Virtual Robot Path Points

In robot applications, there are often the cases in which the robot programmer must be able to quickly and reliably change the existing robot points in the robot program so that they can be accurate to the slight changes of components in the existing or identical robot work-cell.

It is obvious that inevitable differences exist between the real robot work-cell and the simulated robot work-cell because of the modeling tolerance and dimension variation of the corresponding components.

Therefore, it is not feasible to directly transfer tag path points to the actual robot controller for execution. Instead, the robot programmer must apply the prototype calibration functions to adjust the tag path points with respect to a number of robot points imposed from the real robot work-cell.

Different methods have been developed for measuring the dimensional difference of the similar components in the robot work-cell and using it to convert the robot points in the existing robot programs.

The robot programmer must "measure" the positional variations of two similar points in the real robot work-cell and compensate the pre-taught robot points with either the robot system utility function or the robot program instruction.

However, if the dimensional difference exists between two identical robots, an external calibration system must be used for identifying the robots' difference so that the taught robot path points for one virtual robot system can be transferred to the real identical one. The process is called the robot calibration [3].

Prior to the robot calibration, the robot programmer needs to conduct the calibration experiment in which a developed robot calibration program moves the virtual robot frames to a set of taught robot calibration points. Depending on the required accuracy, a set calibration points is required.

It is also important to select robot calibration points that are able to move each robot joint as much as possible in order to "excite" its calibration parameters. The dimensional difference of the robot joint parameters is then determined through a specific mathematical solution such as the standard non-linear least squares optimization.

Theoretically, the existing robot kinematics model can be modified with the identified robot parameters.

However, due to the difficulties in directly modifying the kinematic parameters of an actual robot controller, the external calibration system compensates the corresponding joint values of all robot points in the existing robot program by solving the robot's inverse kinematics equations with the identified robot joint parameters.

A robot calibration system must be able to identify the parameters of robot joint frames, in two "identical" robot work-cells, and compensate the existing robot points so that they can be transferred to the identical robot system for execution.

With the calibrated prototype frames and the assumption that the virtual robot prototype is exactly the same as the real robot, the positions relative to the robot base frame $R(x, y, z)$ in the simulation work-cell is exactly the same as the corresponding one in the real robot work-cell.

5. Experimental Configuration for Robot Programming by Imitation

Figure 2 also displays (left image) the user interface of a virtual anthropomorphic robot arm, which has been created by the motion simulation system.

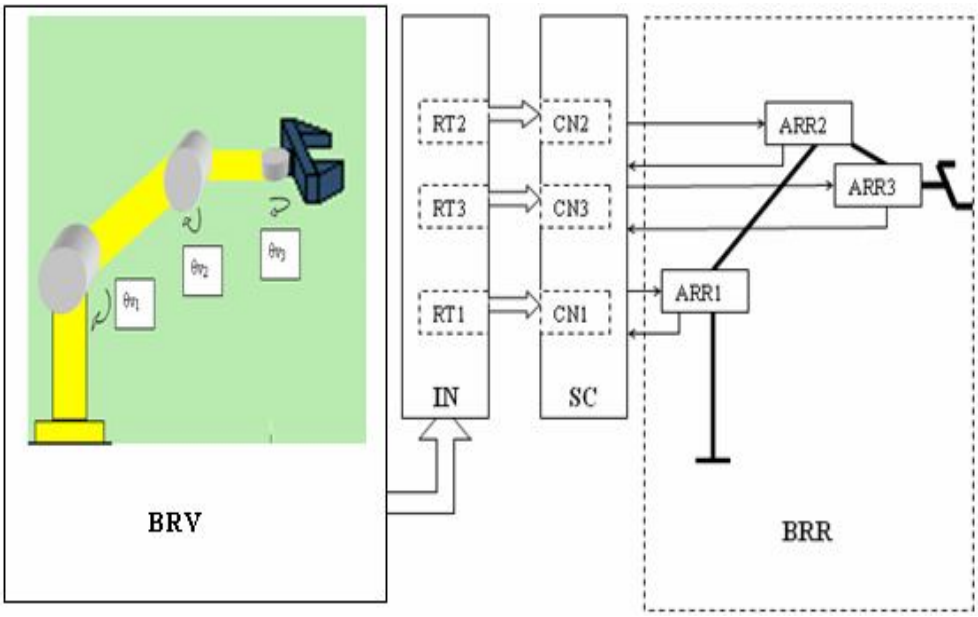

Fig. 2. *Imitation programming system with corresponding virtual and real robot* (*face-to-face*)

One transfers via intelligent interface the joint angles data from a motion capture system to a kinematic model for an anthropomorphic robot.

To generate the desired motion sequence for the real robot, we capture the motions from a virtual robot model and map these to the joint settings of the physical robot.

Initially, a set of virtual postures is created to the virtual robot arm BRV and the pictures' positions are recorded for each posture, during motion [6].

These recorded pictures' positions provide a set of Cartesian points in the 3D capture volume for each posture.

To obtain the final 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.

We assume to 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.

The data is obtained using a motion capture channel taking into account the joint motion range. The symbolic spatial relations specifying the virtual environment can be used for the automatic pursuit of possible virtual path as well as for planning of appropriate behavior of the real robot arm BRR, which may guide the motion process during execution.

The easiest way to generate the spatial relations explicitly is the interactively programming of the behavior of the virtual prototype in his virtual environment in order to specify suitable positions θ_{v1}, θ_{v2}, θ_{v3}.

This kind of specification provides an easy to use interactive graphical tool to define any kind of robot path; the user has to deal only with a limited and manageable amount of spatial information in a very comfortable manner.

An automatic robot programming system has to recognize the correct robot task type and should map it to a sequence of robot

operations [11]. The desired pathways are automatically transferred and parameterized in the numerical interface IN, using the path planner.

The applicable robot tasks are designed and the desired pathways are programmed off-line and stored in the functional modules RT1, RT2, RT3.

The pursuit controllers compute the estimate output of the comparative modules CN1, CN2, CN3 the future state of the virtual robot prototype and the measured state of the physical robot.

While motion execution is in progress, the real robot joints ARR1, ARR2, ARR3 are activates into the real environment. Each time, a skill primitive is executed by the robot control system SC; it changes the robot joints state. As no time limit for the motion is specified, the real robot imitates the behavior of the virtual robot.

In our laboratory currently we are developing Cartesian control architecture able to interpret the physical robot commands in the above given form. The basis of our implementation is a flexible and modular system for robot programming by imitation.

In our experimental configuration in order to prove the correctness of the robot programming by imitation we have chosen an anthropomorphic robot arm, with 3 DOF equipped with electrical actuators, mounted on the real robot's joints.

The robot's control unit is connected via TCP/IP to a PC equipped with the interface card; the PC is running the simulation and control process. The robot control system receives and executes each 16 ms, an elementary move operation.

6. Conclusion

In this paper the author has reformulated the motion planning problem into a behavioral simulation problem associated with behavioral imitating techniques; where the virtual robot path guides the real robot, from its starting configuration to its target.

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

Programming real robots, especially to perform the behavior of the virtual robots is accomplished by imitation, using virtual robots motion data capture.

The actions for virtual robot are transferred, with a central coordination to corresponding physical robot which must imitate her virtual homonym.

The virtual robot path points, created in the simulated work-cell, must be adjusted because of the dimensional differences components between the physical robot and virtual robot.

In this paper one assume that our strategy is able to deduce the exact shape, position and velocity of the virtual robots and of the virtual obstacles, in the virtual environment. One transfers the behavior of the virtual robots, in the real world to the physical robots.

This paper is focused on the programming by imitation, transferring of the motion mapping from virtual space in 3-D dimensional real space.

The author expect fully automated robot programming by imitation, using robust enough system to be applied in industrial applications, will not become true before the end of this decade.

References

1. Asfour, T., Azad, P, Gyarfas, F., Dillmann, R.: *Imitation learning of dual-arm manipulation tasks.* In: International Journal of Humanoid Robotics **5** (2008) No. 2, p. 183-202.
2. Barzel, R., et al.: *Plausible motion*

simulation for computer graphics animation. In: Proceedings of the Eurographics Workshop on Computer Animation and Simulation, New York, December 1996, Springer-Verlag Publisher, p. 183-197.
3. 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.
4. Cheng, F.S.: *Programming Vision-Guided Industrial Robot Operations*. In: Journal of Engineering Technology **26** (2009) No. 1, p. 10-15.
5. Connolly, C.: *Artificial Intelligence and Robotic Hand-Eye Coordination*. In: International Journal of Industrial Robots **35** (2008) No. 6, p. 496-503.
6. Fratu, A.: *Method and installation for joints trajectory planning of a physical robot arm*. In: (proposal patent) unpublished.
7. Sugihara, T., Nakamura, Y.: *A Fast Online Gait Planning with Boundary Condition Relaxation for Humanoid Robots.* In: Proceedings of IEEE International Conference on Robotics and Automation, Barcelona, Spain, April 18-22, 2005, p. 305-310.
8. Venture, G., Ripert, P.J. Khalil, W., Gautier, M., Bodson, P.: *Modeling and identification of passenger car dynamics using robotics formalism*. In: Journal IEEE Trans. on Intelligent Transportation Systems **7** (2006) No. 3, p. 349-359.
9. Weinstein, R., Teran, J., Fedkiw, R.: *Dynamic simulation of articulated rigid bodies with contact and collision*. In: Journal IEEE Transactions on Visualization and Computer Graphics **12** (2005) No. 3, p. 365-374.
10. Yamane, K., Nakamura, Y.: *Natural motion animation through constraining and de-constraining at will.* In: Journal IEEE Transactions on Visualization and Computer Graphics **9** (2003) No. 3, p. 352-360.
11. Zhang, L., Young, J., Kim, Y.J., Manocha, D: *A hybrid approach for complete motion planning.* In: Proceedings of IEEE International Conference on Intelligent Robots and Systems, San Diego, California, October 29 - Nov. 2, 2007, p. 7-14.