2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Intuitive and Model-based On-line Programming of Industrial Robots: A Modular On-line Programming Environment

Björn Hein and Martin Hensel and Heinz Wörn
Institute for Process Control and Robotics (IPR)
Universität Karlsruhe (TH)
[hein|hensel|woern]@ira.uka.de

Abstract— **This paper focuses on the simplification of the on-line programming process of industrial robots. It presents a modular on-line programming environment (software and hardware), which supports an intuitive way of moving and teaching robots, while supporting the user with assisting algorithms like collision avoidance and automatic path planning. Main idea is the combination of different approaches from tele-operation, programming by demonstration and off-line programming, and reuse them in a new fashion on-line on the shopfloor. The proposed programming environment is designed to evaluate the usability of different combination of assisting techniques.**

I. INTRODUCTION

A. Motivation

Programming of industrial robots is nowadays done in simulation systems like em-Workplace, IGRIP, Robot Studio, Solid-Works, KUKA-Sim, etc. Herein the complete robot cell is modelled in 3D. The user can test the reachability, fine-tune the motion properties of robot movements and handle process related information. In principle, out of these information, complete robot programs could be automatically generated and transferred to the robot controller and PLC. Of course, no available simulation system on the market has already completely implemented this programming chain, but a lot of parts exist already separately. Until now a lot of effort was spent in algorithms for automating the programming process in such simulation systems (e.g. algorithms for collision free path generation). Thanks to efficient automatic path-planning algorithms developed in the last ten years [1], [2], [3], [4] manual work in simulation systems could be significantly reduced for special applications (e.g. spot welding, laser welding).

In the industrial environment a lot of energy was put in off-line algorithms and only few efforts were spent in supporting on-line programming/jogging with appropriate user interfaces. But on-line programming is still necessary for a number of reasons: currently the "optimal programming chain" from simulation system directly down to the robot is only working in special applications (e.g. spot welding). Moreover, there are movements that can not be programmed beforehand due to unknown CAD-data or there are tasks that can only be on-line programmed by a human being - one of the most flexible "sensors" existing.

However, from the ergonomic point of view manual jogging of industrial robots is still not very satisfying:

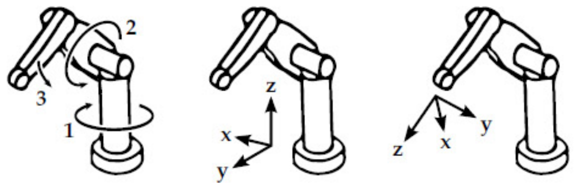

Fig. 1. Typical motion modes - left: Motion is with respect to the axis of the robot, mid: with respect to world coordinate system, right: with respect to tool center point (TCP) coordinate system

"…. In the past, user interfaces for industrial robots have sometimes been quite rudimentary. Still today, the personnel working with robots is mostly very experienced and has often learned to cope with whatever is available… [5]". A compact overview of usability in human-machine-cooperation can be found in [6] and a good motivation for the need of intuitive user-interface for on-line programming can be found in [7]. Some recent works deal with new intuitive ways of programming and describing the task a robot has to fulfill [8], but that is not the focus of our study. The purpose of our work is concentrating on the intuitive on-line jogging of an industrial robot.

Common robot input devices are teach pendants like the ones from KUKA, ABB or Reis. Since these pendants have to give access to all functionalities provided by the robot and the robot controller, they have some drawbacks: big, heavy and not very intuitive to use. Indeed apart from the buttons, most of the panels have special input devices (Joystick, or 6D-Mouse) and different moving modes (s. Fig. 1). But jogging the robot with them requires a lot of experience. There is especially one major drawback: The position of the user with respect to the robot is not taken into account while jogging the robot. Therefore, the user has to adapt himself and always think in different coordinate systems (s. Fig. 1).

B. Related Work

Investigations were done concerning usability of teach pendants or input devices to identify, which are the appropriate ways for jogging industrial robots for untrained users. In [9] was investigated, how jogging in certain coordinate systems influences the number of wrong user actions (e.g. pressing the button for -X instead of +X). As result it was found, that solving the given task in a robot centric

	Standard mode	Compatible Mode
Time	615s	540s
Number of errors: Buttons	9.3	2.8
Number of errors: Joystick	4.8	2.0

TABLE I

COMPARISON TIME/#ERROR WITH DIFFERENT INPUT DEVICES AND MODES [11]

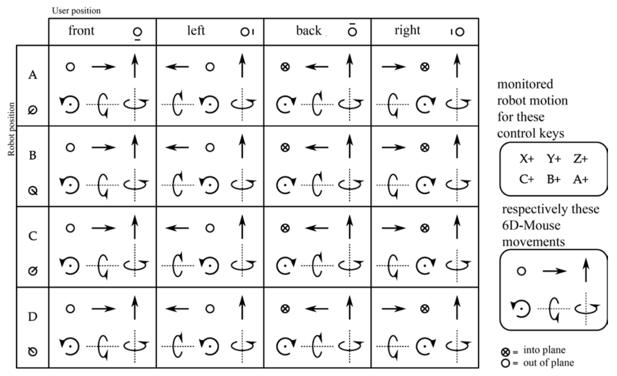

Fig. 2. Matrix of confusion jogging in world mode[11] - On the left are the given inputs using the buttons or 6D-Mouse. The matrix shows the resulting movement of the robot depending on the position of the user.

coordinate system took 24,5s instead of 12,8s in a user centric coordinate system. Comparing buttons and joystick, 73% of the users preferred buttons. The joystick provoked 7% more errors.

In [10] jogging with joystick and buttons in world coordinate system and controlling velocity was investigated. It was recorded, how long a user needs for moving the robot to a desired position. With joystick people were 25% faster. Especially the learning curve with joystick was significantly better.

In [11] the *matrix of confusion* was introduced (s. Fig 2). It describes which input action causes which robot movement depending on the relative position of user and robot. In [12] the effect of this matrix on user input using teach pendants was investigated. Besides the "standard mode" a new mode was introduced: the "compatible mode". In the compatible mode the coordinate system is adapted in 90° degree steps depending on the position of the user and therefore simplifies the matrix of confusion drastically (s. Fig. 3). The time and the numbers of errors for accomplishing a given task was evaluated. As seen in Table I jogging in "compatible mode" reduced time and number of erroneous actions. The compatible mode decouples the coordinate systems of user and robot. The user has only to "think" in his own coordinate system and therefore can act safer and faster.

This idea is one of the basic concepts of the presented "Modular On-line Programming Environment".

C. Outline

The paper is organized as follows: In section SYSTEM COMPONENTS the core parts of the developed on-line programming environment are presented. The following sections HARDWARE SETUP and MODULAR SYSTEM ARCHITECTURE are giving a brief overview about the used and developed hardware and the software architecture. The paper closes with conclusions and an outlook on future developments.

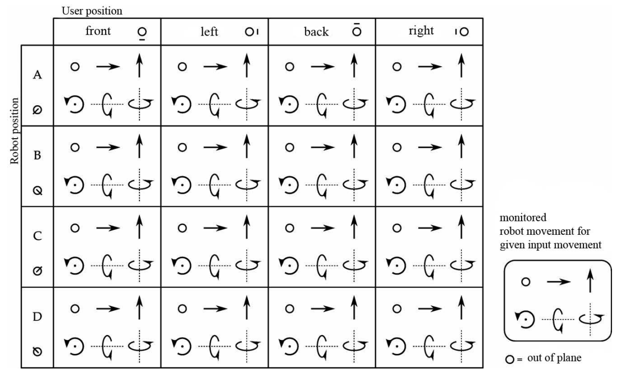

Fig. 3. Motion compatibility - input motion of the user is reflected by a compatible robot motion: every matrix element exactly represents the input values.

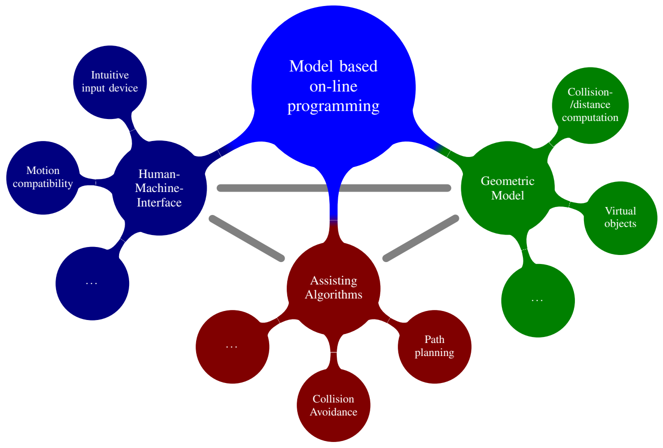

Fig. 4. Concept of the developed "Modular On-line Programming Environment" - It is based on the idea of combining an intuitive input device, a geometric model representing the environment and assisting algorithms that are based on the geometric model and react on the given input.

II. SYSTEM COMPONENTS

The proposed *Modular On-line Programming Environment* is basically build on three main components (s. Fig. 4): an intuitive *Human-Machine-Interface* to control the movement of the robot using a mobile guiding device, a *geometric model* representing the environment given by CAD or sensor data and *assisting algorithms* working on the geometric model supporting the user while moving the robot.

A. Human-Machine-Interface

1) Intuitive input device: Some of the developed input/guiding devices are shown in Fig. 5. They are very simple prototypes for testing different setups (number of buttons, kind of analog knobs or sliders) and can be held in one hand. The sphere markers are used to track the position and

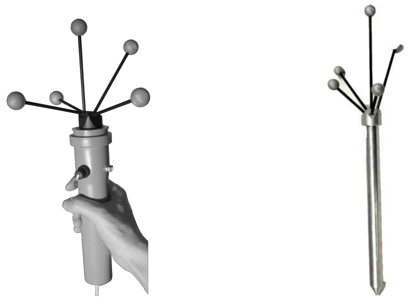

Fig. 5. Input devices - They are tracked via a tracking system by the sphere markers. Left: General input device with different kind of knobs, sliders and buttons installed; right: pen-like input device.

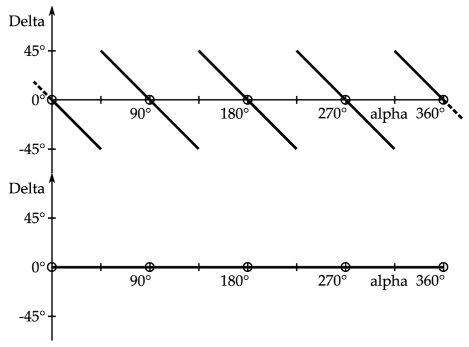

Fig. 6. Extended motion compatibility - (above) The matrix of confusion shown in Fig. 3 is in [12] only valid in the middle of every 90° sector. At every other position there is a certain amount of error (Delta), which forces the user to adapt his input motions, (bottom) In our setup we benefit from the 6D tracking system which gives us the absolute position and orientation of the guiding device. With this information we have a compatible motion in any position of the user.

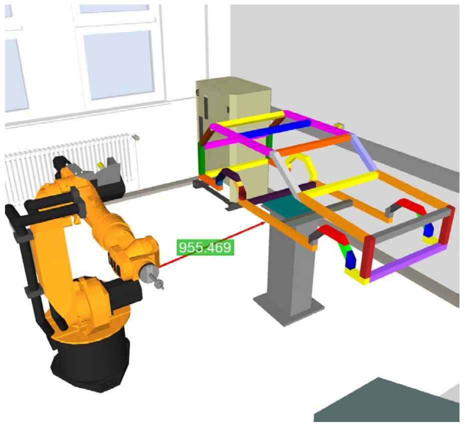

Fig. 7. Geometric Model of the laboratory at the IPR in Karlsruhe - A KR16 and a miniature car body for testing and education. Distance information are computed in real-time and used for assisting algorithms (e.g. adjust jogging speed depending on the distance to obstacles).

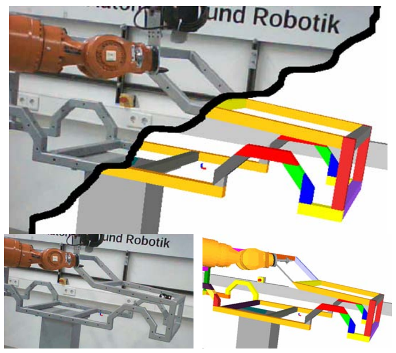

Fig. 8. Calibrated model - Simulation model and reality match exactly.

orientation of this device using a multiple infrared-camera system. Every movement of this device is transferred to the robot and executed in real-time. The tracking system is at the moment mainly used to demonstrate the possibilities when using a freely movable input device. In the future it is planned to replace the stationary tracking system by inertial sensors.

2) Extended motion compatibility: Initially the user has to do a calibration step. He defines his desired coordinate system by pointing three points in space using the guiding device. From that moment on motion compatibility is independent of changes of the position and orientation of the user. In [12] the space around the robot was split in 4 sectors (90°). So real motion compatibility was only achieved at the mid of every sector (s. Fig. 6 above) We can directly benefit from the 6D information of the tracking system and achieve motion compatibility in every point in work space (s. Fig. 6 bottom).

One of the input device's knobs adjusts the scaling between user motion and robot motion. This kind of jogging is so effective, that even untrained users could move the robot intuitively. Using a high scaling factor, the users can move the robot very fast, using only small movements of their wrist, from the given start position to the end position. A low scaling factor allows very accurate movements. Using the pen-like input device (s. Fig. 5, right) it is e.g. possible to write a name with a pen attached to the robot on a sheet of paper (s. Fig. 10, left). All this could be done with none or very little amount of practise.

B. Geometric Model

Besides supporting the user with motion compatible jogging additional ways of assisting are planned to be investigated. Therefore a geometric model was included in the developed programming environment. It contains the geometric data of all objects in the working cell (e.g. obstacles and robots) (s. Fig. 7). The position of the robot is updated in real-time every 12ms. The model of the car is calibrated so that it exactly match the real car body (s. Fig. 8).

1) Fast collision detection/distance computation: Based on the geometric data a fast an efficient way of computing distances between objects was integrated [13]. It is capable of calculating distances in the same cycle time as the robot is updated (12ms) and is used by the assisting algorithms explained later.

2) Virtual objects: Virtual objects are not really existing but describing some geometrical extension or properties used by assisting algorithms (e.g. a region, where the tool center

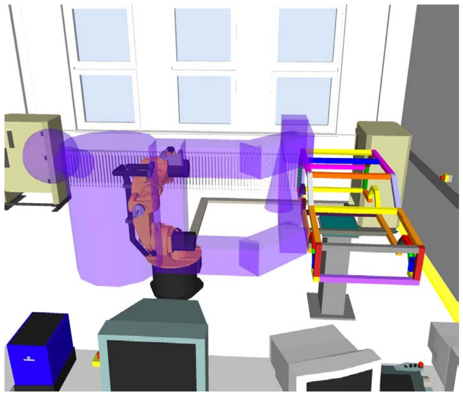

Fig. 9. TCP restriction - the user can only move the TCP inside the lucent object.

point (TCP) of the robot is not allowed to leave or to enter). It is possible to define paths and points with an additional radius. Based on this radius, virtual forces can be integrated influencing the robot movements ("snap paths/points").

C. Assisting Algorithms

The "Assisting Algorithms" support the user during its manual jogging. Either to simplify the jogging or to restrict and protect the user or the equipment.

1) Real-time Collision Detection/Avoidance: Based on this module, different supporting strategies were implemented:

- **Collision Avoidance**: It prevents the user of navigating the robot into self-collision or in collision with the environment.
- **Adapting jogging speed**: Depending on the obstacle distance or distance to the user, the jogging speed is automatically adapted (s. Fig. 7). E.g. If the robot is near the guiding device so is the user. To avoid fast and harming movement the scaling factor has to be automatically reduced.
- **Extended user notifications**: According to the distance to any obstacle, an alarm (like the parking distance control for automobiles) is sounding.

2) TCP restriction: This module was implemented to restrict the possible movements of the TCP:

Geometrical restriction: An object of arbitrary shape can be defined which the TCP cannot leave or enter. In Fig. 9, an example of a "working place combined with transfer tube" is shown. It has one huge region on the left in which the user can do complex tasks with the robot whereas when moving to the right he is forced to keep the TCP in a specified corridor, avoiding collisions with the car. The geometrical restriction was also used to protect the user while writing with the virtual pen (s. Fig. 10, right).

Based on the same algorithms a sliding along arbitrary planes in given orientation (e.g. perpendicular) is achieved. A typical use-case for this could be painting with an airbrush, which has to be kept in a certain distance.

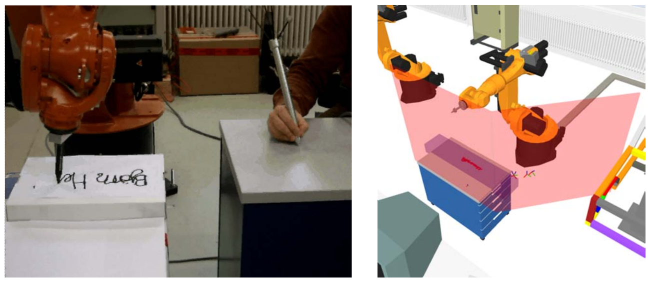

Fig. 10. Left: Writing with a virtual pen - The manual movement is executed and copied at the same time by the robot; right: TCP restriction and virtual walls protect the user.

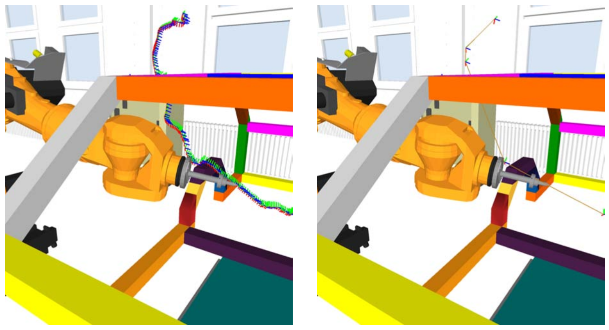

Fig. 11. Automatic smoothing - left: Path generated manually by the user with input device; right: Online optimized path.

Path restriction: Similar to the geometrical restriction mentioned above, the TCP can be restricted to paths. But paths have additional information (e.g. length, orientation along path,...). For this reason two moving modes were implemented. One mode moves the robot forward and backward along the path by pulling the guiding device from left to right. In the second mode one can use the analog slider on the guiding device to determine the forward or backward speed along the path. The two modes are mainly used in conjunction with the automatic path planning algorithms described in the following.

3) Automatic path planning: As we have a geometric model, we could make use of the implemented path planning algorithms [14], [15] to support the user. We added to the programming framework a module that enables the user to move the robot by just pointing out the start and the goal position. A collision free path is then generated automatically. Then, it is very easy to move the robot, using the module *path restriction*. Our research will focusing in the future especially on combining manual jogging and automatic path planning.

An approach for combining manual input and automated smoothing algorithms supporting programming in simulation systems was presented in [16]. This approach could be directly applied to the on-line programming environment and was integrated as module. In Fig. 11 on the left the manually recorded path is shown. Whereas on the right the automatically smoothed path is shown.

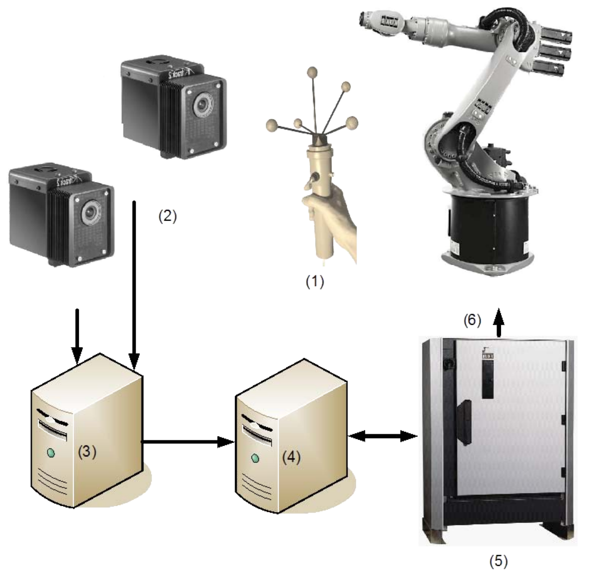

Fig. 12. Hardware setup - The guiding device (1) is tracked by cameras (2), the Tracking-PC (3) sends the information about the position to the Master-PC (4). Based on the position of the guiding device, it's status and the used assisting modules, joint values are generated and sent via real-time Ethernet every 12ms to the KR-C2 Controller (5), which drives the robot (6).

III. HARDWARE SETUP

The hardware setup is shown in Fig. 12. It consists of:

- KUKA KR16 Industrial Robot
- KR-C2 Edition2005 Controller
- Real-time-Linux PC running RTAI and RTnet
- Tracking-cameras
- Tracking-PC
- Guiding Device

The Real-time-Linux PC is the master. It gets the information about the position of the guiding device via Ethernet from the Tracking-PC, and via USB the status of the knobs, sliders and buttons of the guiding device. The entire on-line programming environment is running on the master PC. Based on the given user input and used assisting modules, it generates the necessary motion profiles and sends/gets the joint values every 12ms to/from the robot controller. The robot controller is then driving the robot with these commanded values.

IV. MODULAR SYSTEM ARCHITECTURE

The *On-line Programming Environment* is implemented in C++ and has a flexible, modular and component-based design. As mentioned in the abstract, this programming environment should be used for usability studies of different combination of the proposed assisting algorithms. Therefore control and data flow have to be freely configurable. A component has input and output ports (s. Fig. 13), which can be freely interconnected between other components as long as their data types match. Simple and complex data types such as: double, float, int, bool, vectors,. . . are supported.

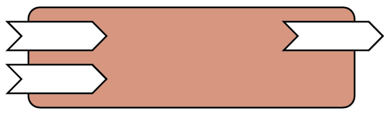

Fig. 13. Example for a component of the programming environment with two input and one output ports.

Fig. 14 gives an example for the buildup for the basic jogging functionality comprising things like calibration of wrist rotation, user orientation, stabilization,. . . and so on.

V. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

In this paper we presented a modular on-line programming environment for industrial robots. The main goal of this programming environment is to evaluate the usability of of different assisting algorithms for on-line programming of industrial robots. In this study, the numerical analysis of the system is not included. But remarkable results can be observed with the satisfactory experimentations performed by students:

- Untrained people could solve immediately complex tasks due to the implemented combination of the extended compatibility mode and the assisting algorithms.
  - Moving robot to different positions in presence of obstacles.
  - Writing/Painting on a sheet of paper.
  - Following pre-printed patterns with a pen mounted on the robot.
- The system requires only a small amount of brain work.
  - Natural Hand-Eye-Coordination.
- Complex trajectories are generated intuitively.
  - With buttons: impossible to be performed.
  - With joystick: control not very intuitive.
- Using the "park distance control" module mentioned above, students could move the robot out of the car model (s. Fig. 7) with closed eyes.

Some of the ideas presented are seen in tele-operated environments and systems using programming by demonstration. In contrast to these systems our system focuses on the on-line control of the robot. The user is working in the robot cell and the motion of the guiding device is immediately interpreted and executed as a robot motion (except for the automatic path planning module). So the user directly interacts with the robot and is assisted by our algorithms. For human-machine-interaction in Service Robotics or in Humanoid Robotics force-torque-sensors are often used as guiding device. Parts of the presented programming environment are of course also applicable to this kind of interaction.

B. Future Works

Regarding the usability, the next steps are to develop scenarios/benchmarks and evaluate different assisting algorithms with the traditional approach.

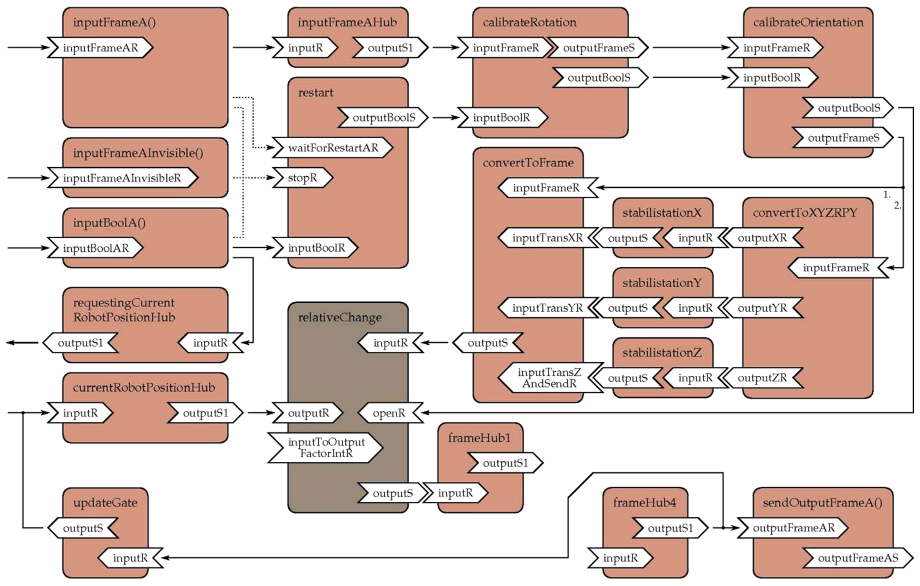

Fig. 14. Example for the basic jogging functionality build out of components

Concerning the guiding device it would be beneficial to skip the expensive tracking system using inertial sensors. Indeed, this is one goal for the future. For most use-cases it is not necessary to know the absolute position of the guiding device but the relative motion. Nowadays sensors can keep calibration at least for about 15-20 seconds. In combination with the human as flexible compensating sensor, such an inertial sensor based guiding device could be used for 30-50 seconds before drift is getting to worse. This amount of time would be completely sufficient for movement control. Especially calibration of such a guiding device could be easily achieved: just pointing to the robot (≈1 sec) and again 30-50 seconds of programming.

There is currently one special drawback regarding the automatic path planning module: the geometric model has to be calibrated to match exactly with reality. This is of course the same drawback when programming in off-line simulation systems. One possibility to overcome this would be to record and incorporate sensor data (e.g. Laser-scanner, PMD-Sensors,...) in the geometric model. But this is not in focus of our investigations.

REFERENCES

[1] G. Sánchez and J. C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Proceedings International Symposium on Robotics Research*, 2001.

[2] J. P. v. D. Berg and M. Overmars, "Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners," in *International Journal of Robotics Research, The*, 2005, vol. 24, ch. 12, pp. 1055–1071, http://ijr.sagepub.com/cgi/content/refs/24/12/1055.

[3] K. E. Bekris, B. Y. Chen, A. Ladd, E. Plaku, and L. E. Kavraki, "Multiple query probabilistic roadmap planning using single query primitives," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[5] A. Kazi, J. Bunsendal, D. Haag, B. Raphael, and R. Bischoff, *Advances in Human-Robot Interaction*. Springer-Verlag, 2005, ch. Next Generation Teach Pendants for Industrial Robots, pp. 47–66.

[6] S. Thrun, "Towards a framework for human-robot interaction," *Human-Computer Interaction*, vol. 19, no. 1-2, pp. 9–24, 2004.

[7] R. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," in *SR 2006 - ROBOTIK 2006 : Proceedings of the Joint Conference on Robotics*, ser. VDI-Berichte, no. 1956. VDI-Wissensforum, May 2006, p. 95.

[8] J. Pires, T. Godinho, K. Nilsson, M. Haage, and C. Meyer, "Programming industrial robots using advanced input-output devices: test-case example using a cad package and a digital pen based on the anoto technology," *International Journal of Online Engineering (iJOE)*, vol. 3, no. 3, 2007.

[9] S. V. Gray, J. R. Wilson, and C. S. Syan, *Human Robot Interaction*. Taylor Fancis, 1992, ch. Human control of robot motion: orientation, perception and compatibility, pp. 48–46.

[10] H. Brantmark, L. A., and U. Norefors, "Man/machine communication in asea's new robot controller," *Asea Journal*, vol. 55, no. 6, pp. 145–150, 1982.

[11] E. C. Morley, C. Syan, and J. Wilson, "Robot control and the matrix of confusion," in *9th International Conference on CAD/CAM, Robotics and Factories of the Future*, Newark, NJ, August 1993.

[12] E. C. Morley and C. S. Syan, "Teach pendants: how are they for you?" in *Industrial Robot*. MCB University Press, 1995, vol. 22, no. 4, pp. 18–22.

[13] M. Salonia, B. Hein, and H. Wörn, "Fast approximated conversion of workspace distances into free regions in configuration space of robots," in *Proc. of 37th International Symposium on Robotics ISR 2006 and 4th German Conference on Robotics*. VDI Verlag, 2006.

[14] D. Mages, B. Hein, and H. Wörn, "A deterministic hierarchical local planner for probabilistic roadmap construction," in *Proc. of 37th International Symposium on Robotics ISR 2006 and 4th German Conference on Robotics*, 2006, Munich.

[15] B. Hein and H. Wörn, "Fast hierarchical a* path planning for industrial robots based on efficient use of distance computations," in *Proc. of 37th International Symposium on Robotics ISR 2006 and 4th German Conference on Robotics*, vol. 1956, 2006.

[16] H. Wörn, B. Hein, D. Mages, B. Denkena, R. Apitz, P. Kowalski, and N. Reimer, "Combining manual haptic path planning of industrial robots with automatic path smoothing," in *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics*, Barcelona, Spain, 2005.