The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Intuitive and Model-based On-line Programming of Industrial Robots: New Input Devices

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

***Abstract*— This paper focuses on the simplification of the on-line programming process of industrial robots. It presents in detail the input part of a modular on-line programming environment presented as overview in [1]. Main concept of this programming environment is an intuitive way of moving and teaching robots, while supporting the user with assisting algorithms like collision avoidance and automatic path planning. Goal is the combination of different approaches from tele-operation, programming by demonstration, Virtual Reality and off-line programming, and to reuse them in a new fashion on-line on the shop-floor. This paper presents some ideas and concepts, how input devices and strategies for robot programming could look like and how to use them to set up an intuitive manual motion control of the robot.**

I. INTRODUCTION

A. Motivation

As stated in [1] the main purpose of the proposed on-line programming environment is the simplification of jogging and programming industrial robots. 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. Untrained people getting frustrated when trying to program a robot using the pendant, coming along with a standard industrial robot. 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. Following the idea also mentioned in [2], [3] the combination of a human, being one of the most versatile sensors, and a intuitive way of jogging a robot could greatly improve efficiency in production and could create new applications for industrial robots. In this paper we focus on the input part of the proposed on-line programming environment in [1] (s. Fig. 1). A compact overview of usability in human-machine-cooperation can be found in [4] and a good motivation for the need of intuitive user-interface for robot programming can be found in [5]. To summarize the above: There is a need for new input devices and input strategies for robot programming.

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 [6]. Moving an industrial robot with six degree of freedom is a mental challenge Normally the user has to adapt himself and always think in different coordinate systems (world/robot/user centric). Therefore in [7] was investigated, how jogging in certain coordinate systems (user centric vs. robot centric) influences the number of wrong user actions. In [8] the *matrix of confusion* was introduced. It describes which input action causes which robot movement depending on the relative position of user and robot. In [9] the effect of this matrix on user input using teach pendants was investigated. Additionally to the "standard mode" the "compatible mode" was introduced. 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. The time and the numbers of errors for accomplishing a given task was evaluated and the "compatible mode" drastically reduces errors and programming time.

Besides the coordinate systems the input device/panel itself is under investigation. Instead of one universal pendant, smaller and smarter devices, that are dedicated to special applications, seem to be an appropriate way to reduce complexity in robot programming. Already in the early 80ties [10] jogging with joystick or buttons was compared.

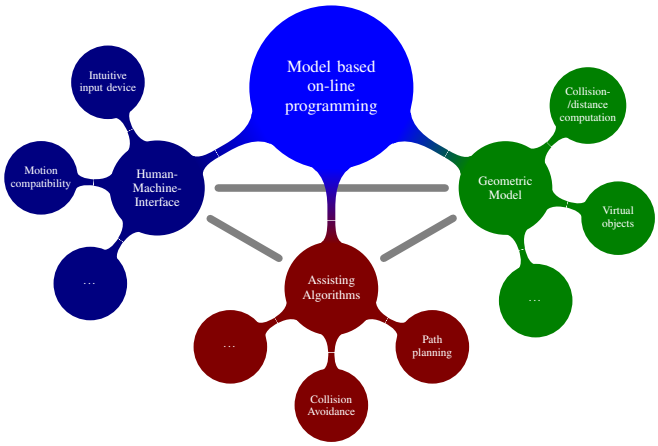

Fig. 1. Concept of the developed "Modular On-line Programming Environment" - It is based on the idea of combining an intuitive input device, a geometric model representing the environment and assisting algorithms that are based on the geometric model and react on the given input. In this paper we focus on the Human-Machine-Interface: input devices, motion compatibility, ...

Based on a force-torque-sensor, a programming system was presented, that creates robot programs out of inaccurate user input data through compliant motion generation [11], [12]. A joystick with force feed-back was proposed [13] to program polishing tasks. Instead of transforming user input 1:1 to robot movements in [14], [15] a digital pen is used to describe and program a robot task by hand-writing. It is mainly 2D but allows recording of writing speed and pen pressure to be used as variables and transformed e.g. to moving speed, acceleration, force,.... For recording hand-writing an inertial sensor based pen was proposed in [16], which could be used to record smooth robot trajectories. A 3D approach is shown in [17], with an tracked hand held input device, an industrial robot can be directly controlled in a master-slave manner. Combining VR and robot programming a free movable device is proposed in [18]. A virtual paint gun is connected to the tracked input device simulating the painting process. Then the robot program can directly downloaded to the robot. Other research uses force-torque-sensors to interact directly with the robot [19]. Additionally in [20] the geometric model (space without obstacles) is cleverly generated on the fly during user interaction. Based on this geometric data in a second step the user generated path and can be automatically optimized. Due to the model based approach the latter directly relates to our approach [1]. But instead of direct interaction, we focus on free movable 6D input devices to be in a certain safety distance to the robot and to create programs and motions without the robot itself moving. Regarding robot input devices the research on usability of input device in VR [21], can be a source for inspiration for input devices in robot programming.

C. Outline

The paper is organized as follows: In section INPUT DEVICES the developed input devices will be shown and explained. Afterwards the needed and therefore implemented modules for an intuitive motion control based on these devices will be discussed. The following sections LABORATORY SETUP is giving a brief overview about the test environment. The paper closes with conclusions and an outlook on future steps.

II. INPUT DEVICES

A. Hardware

Some of the developed input/guiding devices are shown in Fig. 2 and Fig. 3. They all are equipped with reflective sphere markers that allow position and orientation (6D) tracking by an infrared-camera system. In Fig. 4 the laboratory of the IPR and the volume that is captured by 4 infrared cameras is shown. The common idea behind all proposed input devices in this paper, is either to transform user movements with the input device directly to the robot or to use the input device as direction indicator, pointing in the direction the robot should drive to.

Our "design" decisions for these input devices are based on simplicity and flexibility. First of all, the input device should be suitable for *one hand use*. Therefore it has to be

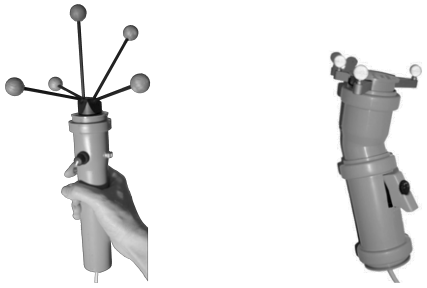

Fig. 2. Input devices - They are tracked via a tracking system by the sphere markers. Left: device with a trigger button controlled by the forefinger, a control dial and a sliders button controlled by the thumb.; Right: device with a trigger button on the back controlled by the thumb (invisible on this picture) and a drill-machine-like slider button.

Fig. 3. 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.

light-weighted and should easily fit in a hand, while all dials, sliders and buttons are reachable with the fingers of the same hand. Second, we are looking for a *one button motion control* meaning that all actions that affects motion control of the robot (e.g. starting, moving, stopping the robot and including the needed calibration) should be possible with only one button. For investigation and to test different kind of setups we firstly build up the two hand held devices shown in Fig. 2. On the left side it is demonstrated how it fits in one hand. The forefinger is on a pushbutton, which triggers actions as long it is pressed based on the current state of the robotic system (e.g. move robot). Furthermore there are two analog buttons - a control dial and a slider button - reachable by the thumb. The control dial stays in its adjusted position, the slider button automatically resets to zero position. These two buttons can be mapped to different control parameters (e.g. speed of the robot, scaling between user and robot motion).

The input device on the right side of Fig. 2 consists only of two buttons. In contrast to the other device the trigger button is not pushed by the forefinger but by the thumb. A big analog button like the one of a drilling machine can be operated by the fore- and middle finger and is in our tests typically mapped to robots acceleration.

The input devices in Fig. 3 are designed for more dedicated applications. The pen-like input device on the left is mainly used for calibrating the environment, teaching points/paths the robot has to move to/along, and generating virtual objects/obstacles [1]. The pointer can also be used for simulating a virtual pen (s. YouTube [22]).

Fig. 4. Laboratory at the IPR: 2 KR16 robots, a $4 \times 4 \times 2m^3$ volume is covered by the infrared tracking systems. Devices shown in Fig. 2 and Fig. 3 can be used in this volume to control the robots.

The proposed input devices (Fig. 2 and Fig.3 on the left) are adequate for research and study. But regarding safety in industrial environments, it is useful that both hands forced to be on the control device when jogging an robot. To simulate this academically a Logitech-Controller was modified and added to the collection of input devices (s. Fig. 3 on the right). To guarantee that both hands are on the controller additionally to the motion trigger button a second buttons on the rear of the device has to be pushed simultaneously. In this case we have abandoned the *one hand use* but try to keep the *one button motioncontrol*. The controller is additionally equipped with a rumble motor, so feedback can be given to the user (collision danger or leaving work area).

B. Input processing - Frame Filter

As described above, all input devices are tracked via their sphere markers (s. Fig. 2 and 3). The local coordinate system given to the input device is arbitrarily placed by the tracking system somewhere in between these markers (s. Fig. 6). In this paper every coordinate system is defined by using the well known approach of 4×4-matrices or so called *Frames*. The following paragraphs are presenting the transformations from the first frame coming from the tracking system until the last frame that is used as output to control the robot. Every frame transformations is called a *Frame Filter* (s. Fig. 5). A Frame Filter has as input and as output a Boolean value and a frame. The Boolean value coming from a preceding filter indicates that the frame on the input port is valid and the output Boolean value marks that the output frame is valid. The complete data flow between consecutive filters is called the *Frame Pipeline*.

1) Rotation-Calibration-Filter: user - input device: First the input device and its position to the user has to be calibrated. For the input devices in Fig. 2 it turned out that the wrist is an appropriate place to locate the coordinate system. Turning the hand will lead the robot to turn around its"Tool Center Point" (TCP) correspondingly. A fast calibration technique was chosen to compute the "Wrist Center Point" (WCP). In calibration mode the operator has to turn the input device around the wrist to 4 different locations (s. Fig. 6), on which the WCP can be directly computed. For a sphere

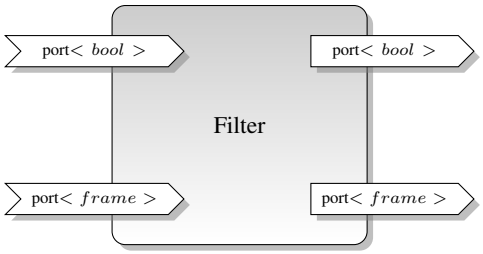

Fig. 5. Frame Filter - it has on the input side as well as on the output side a Boolean and a frame port. A "true" on the input port<bool> imply a valid frame on the input port<frame>. After transformation the frame filter sets its output ports correspondingly.

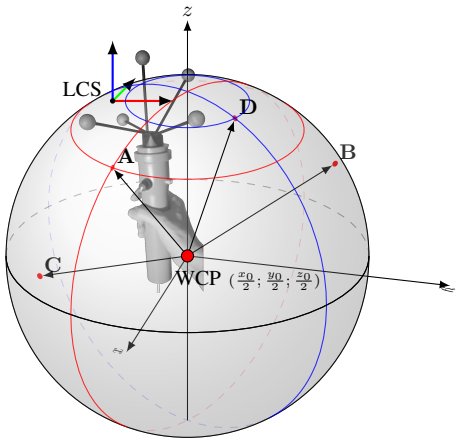

Fig. 6. Rotation-Calibration-Filter - input device: Tracking system places local coordinate system of input device arbitrarily (LCS); to identify "Wrist Center Point" (WCP) the user has to enter 4 points (A, B, C, D), while turning around the wrist.

with center $c = (\frac{x_0}{2}; \frac{y_0}{2}; \frac{z_0}{2})$ and radius r there is

$$(x - \frac{x_0}{2})^2 + (y - \frac{y_0}{2})^2 + (z - \frac{z_0}{2})^2 = r^2.$$

The radius it can also be expressed like $r^2 = x_0^2 + y_0^2 + z_0^2 - u$. This results in

$$x \cdot x_0 + y \cdot y_0 + z \cdot z_0 + u = x^2 + y^2 + z^2$$

Using the translational part of the 4 measured coordinate systems $A \rightarrow (a_x; a_y; a_z), B, C, D$ (s.Fig. 6) we get following linear system of equations

$$\underbrace{\begin{pmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{pmatrix}}_{M} \cdot \begin{pmatrix} x_0 \\ x_1 \\ z_0 \\ u \end{pmatrix} = \begin{pmatrix} a_x^2 + a_y^2 + a_z^2 \\ b_x^2 + b_y^2 + b_z^2 \\ c_x^2 + c_y^2 + c_z^2 \\ d_x^2 + d_y^2 + d_z^2 \end{pmatrix}$$

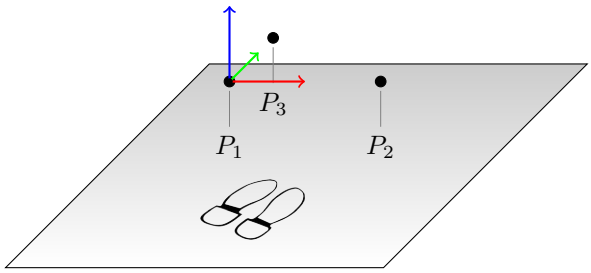

Fig. 7. Calibrate position of user with respect to robot - user has to enter 3 points: first the origin P_1 and then desired direction of x-axis with P_2 and y-axis with P_3; z-axis is computed via $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}$.

Using the Cramer's rule the system is directly solvable:

$$x_0 = \frac{\det(M_1)}{\det(M)} \quad \text{with } M_1 = \begin{pmatrix} a_x^2+a_y^2+a_z^2 & a_y & a_z & 1 \\ b_x^2+b_y^2+b_z^2 & b_y & b_z & 1 \\ c_x^2+c_y^2+c_z^2 & c_y & c_z & 1 \\ d_x^2+d_y^2+d_z^2 & d_y & d_z & 1 \end{pmatrix}$$

$$y_0 = \frac{\det(M_2)}{\det(M)}, \quad z_0 = \frac{\det(M_3)}{\det(M)}$$

After determining the center c, the transformation frame from the local coordinate system of the input device to the WCP can be calculated by normal frame calculations.

The time needed for this calibration step: $3sec$. If the equation system is not solvable the user has to try again with 4 new points. The same calibration principle is used for the pen-like input device. The tip has e.g. to be placed on surface and then turned around recording 4 points. The Logitech controller was manually calibrated in a way that the coordinate system is located in the middle of the controller and parallel to its.

2) Position-Calibration-Filter: input device - robot: The next step is to define the coordinate system the user wants to work in. It can be freely chosen to be aligned with the robot or in any other appropriate way (e.g. a work bench). It is simply done by giving 3 points (origin and direction of x and y axis; s. Fig.II-B.2). The origin is set by the first point P_1, P_2 defines direction of x-axis and P_3 direction of y-axis. In

$$T = \begin{pmatrix} \boxed{\vec{x}} & \boxed{\vec{y}} & \boxed{\vec{z}} & \boxed{\vec{t}} \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

the translational vector $\vec{t}$ is the vector of the first given point P_1. The x-axis $\vec{x}$ is defined by the normalized vector $\overrightarrow{P_1P_2}$ and y-axis $\vec{y}$ is defined by the normalized vector of $\overrightarrow{P_1P_3}$. The z-axis $\vec{z}$ is then generated by the normalized vector of $\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3}$.

Based on the inverse of T the coordinates of the tracking system can be converted to the desired coordinate system of the operator.

It takes about $3sec$ to enter the 3 points via the input device. When using the pen-like input device this calibration can be done very accurate. After the two calibration steps (rotation and position) the motion compatibility is achieved in every point in work space.

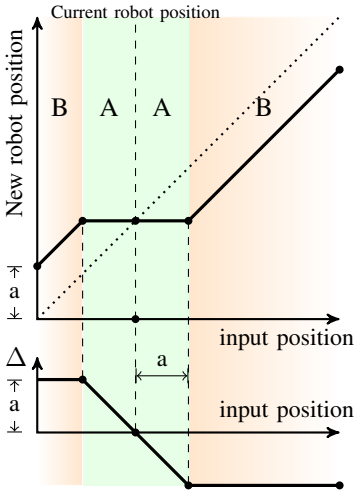

Fig. 8. Stabilisation - As long as input signal is in the A-region the output stays constant, by subtracting Δa from the input (s. lower part). When entering B-region output follows the input value ($\Delta a = const.$). If the Boolean input port (s. Fig. 5) changes from false to true, current robot position (dotted line in the middle) is updated. When choosing a appropriate, noisy data and tremor of user can be removed.

One interesting thing is to mention: when jogging the robot via an human operator an exact calibration is not crucial. A human is stunningly flexible and corrections are done unconsciously. Even when changing the coordinate system secretly during jogging, the tested human operators haven't noticed this change until a significant offset was reached.

3) Stabilisation-Filter: The noisy data given by the tracking system and the human tremor causes the robot to move, even when the input device is held steady at a fixed position. Therefore a stabilisation filter was added. It is derived from the concepts of [23] using only 2 zones A and B (s. Fig 8). Not until the input device is moved more than a given radius a (s. Fig. 8) around the current robot position, the output signal changes. With the appropriate size of a the noise and the tremor can be removed.

4) Relative-Move-Filter: This frame transformation outputs the relative displacement of the incoming frames, since the time the input port (s. Fig. 5) was set to true.

5) Scaling-Filter: One of the input device's dials or sliders adjusts the scaling between user motion and robot motion. Using a high scaling factor, the user can move the robot very fast from the given start position to the end position, using only small movements of the wrist. A low scaling factor allows very accurate movements. Using the pen-like input device (s. Fig. 3, left) it is e.g. possible to write a name with a pen attached to the robot on a sheet of paper (s. Fig. 11 right or s. accompanying video). All this can be done with none or very little amount of practice.

C. Filter Pipeline

The setup of the Filter Pipeline is done by concatenating the different filters (s. Fig. 10). If a filter is not correctly working (e.g. a calibration filter not correctly initialized), its output port<bool> is set to false. The successive filter will then discard the received frame and set its output port<bool> to false, too, and so on...

As seen in Fig.10 on the left, the input device itself is represented in the pipeline as frame filter. Every time the trigger button is pressed with the forefinger (s. Fig.2 on the left) the output port<bool> is set to true and the frame given by the tracking system is set on output port<frame>. Consequently all filters will be turned on and the robot will move and stop when the forefinger releases the button an the output port<bool> is set to false.

At the beginning, the Rotation-Calibration-Filter is not initialized. If the input device's trigger button is pushed for the first time, the Rotation-Calibration-Filter takes the given frame as first point, subsequently if the trigger button is pushed again the next frames are taken. As mentioned above after 4 points the Rotation-Calibration-Filter is calibrated. Next time the trigger button is pushed the Calibration-Rotation-Filter transforms the given frame and sends it to the output and sets the output port<bool> to true. From this moment on, the Position-Calibration-Filter gets the next frames, and so on...With this technique all filters can be set up with just one push button. Therefore we call it the *one button motion control*.

III. LABORATORY SETUP

The hardware setup is shown in Fig. 9. It consists of a KUKA KR16 Industrial Robot, a KRCed05, a Real-time-Linux PC running RTAI and RTnet, two tracking-cameras and one of the proposed guiding devices. The Real-time-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 dials, 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, it generates the necessary motion profiles and sends the joint values every $12msec$ to the robot controller. The robot controller is then driving the robot with these commanded values. The proposed frame filters are programmed using the framework OROCOS [24].

IV. RESULTS AND CONCLUSIONS

A. Results

To compare the traditional way of programming with the proposed input method, several benchmarks were build up compuaring the number of errors and the time needed to fulfill a task. An error was counted once a wrong movement was executed (s. video). Before given the task, all probands could train both of the techniques for 1 minute.

Table I contains results for 4 persons of the benchmark shown in the video. Using the new input technique the ♯ of errors and the needed time could be significantly reduced. People - untrained in classical robot programming - could quite fast adapt the new technique and made in contrast to the classical way no errors. Person 4 - experienced in robot programming - had to get used to the fact, that it is unnecessary to mentally adapt the coordinate system during jogging. In any way, he still was faster and did less errors. Other benchmarks using the simplified car frame (s. Fig. 4, background) simulating spot welding applications, e.g. moving the TCP to 20 given positions on the car frame, did show similar significant results.

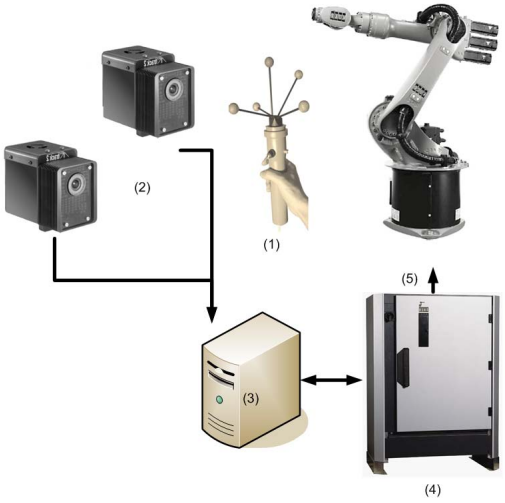

Fig. 9. Hardware setup - The guiding device (1) is tracked by cameras (2), the information about the position is sent to the Master-PC (3). Based on the position of the guiding device, it's status and the used assisting modules, joint values are generated and sent via real-time Ethernet every 12ms to the KR-C2 Controller (4), which drives the robot (5).

Fig. 11. Video scenes - Left: Jogging the robot with the input device shown in Fig. 2; right: Writing with a virtual pen - The manual movement is executed and copied at the same time by the robot.

The video (s. Fig. 11) is presenting the benchmarks of which the results are shown in Table I. Additionally writing on a sheet with the pen-like input device (s. Fig. 3 on the left) is shown. Doing this with the teach pendant is impossible.

B. Conclusions

Using the new input technique some remarkable results could be observed in benchmarks performed by students:

- Untrained people could fast and error-free solve standard programming tasks due to the intuitive way of jogging robots and the developed *one button motion control*.
  - Jogging the robot to specific points in space (s. video).
  - Writing/painting on a sheet of paper (s. video).
  - Following pre-printed patterns with a pen mounted on the robot.

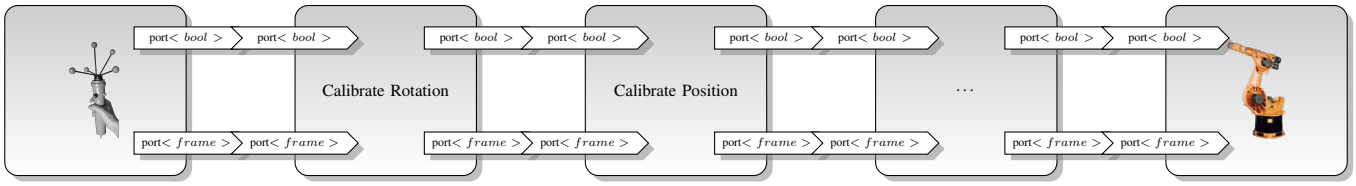

Fig. 10. Cut-out of the input part of the Frame Pipeline - The ports of the filters are connected consecutively. The precessing filter is signaling with its Boolean output port, whether its frames are valid or not. If all Boolean ports are set to "true", the robot will move. If e.g. an error occurs in one filter, its Boolean output port will change to "false",correspondingly all succeeding filter will also turn to "false" and the robot will stop.

TABLE I

COMPARING ♯ OF ERRORS AND ♯ OF TIME NEEDED FOR CLASSICAL AND NEW WAY FOR PROGRAMMING BENCHMARK SHOWN IN THE VIDEO

person	class.♯err	class.♯sec	new ♯err	new ♯sec
1	6	78	0	49
2	11	86	0	59
3	7	73	0	51
4	3	72	2	55

- The system requires only a small amount of brain work (Natural Hand-Eye-Coordination)
- Complex trajectories are generated intuitively compared to other input method (s. video).

The proposed tests did focus mainly on moving and placing the robot. At the moment some investigation are done adding a PDA or mobile phone mounted on the forearm of the user keeping his mobility but allowing to enter technological parameters via a touch screen and giving textual feedback. Further steps will be the investigation how continuous movements can be programmed (e.g. gluing, arc welding, milling,...) using the proposed input techniques. Especially regarding path quality, speed and accuracy.

REFERENCES

[1] B. Hein, M. Hensel, and H. Worn, "Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 3952–3957.

[2] P. Akella, M. Peshkin, E. Colgate, W. Wannasuphoprasit, N. Nagesh, J. Wells, S. Holland, T. Pearson, and B. Peacock, "Cobots for the automobile assembly line," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 728–733 vol.1.

[3] R. Gillespie, J. Colgate, and M. Peshkin, "A general framework for cobot control," *Robotics and Automation, IEEE Transactions on*, vol. 17, pp. 391–401, 2001.

[4] S. Thrun, "Toward a framework for human-robot interaction," *Human-Computer Interaction*, vol. 19, p. 9, 2004.

[5] R. Schraft and C. Meyer, "The need for an intuitive teaching method for small and medium enterprises," in *ISR 2006 - ROBOTIK 2006*, ser. VDI-Berichte, vol. 1956. VDI-Wissensforum, 2006, p. 95.

[6] H. Denkena, Wörn, R. Apitz, R. Bischoff, B. Hein, P. Kowalski, D. Mages, and H. Schuler, *Roboterprogrammierung in der Fertigung*. (German): Springer, 2005, vol. 9, pp. 656–660.

[7] S. Gray, J. Wilson, and C. Syan, *Human control of robot motion: orientation, perception and compatibility*. UK: Routledge, 1992, vol. 2, pp. 48–64.

[8] 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, 1993.

[9] E. C. Morley and C. S. Syan, "Teach pendants: how are they for you?" *Industrial Robot: An International Journal*, vol. 22, pp. 18 – 22, 1995.

[10] H. Brantmark and U. Norefors, "Man/machine communication in asea's new robot controller," *Asea Journal*, vol. 55, pp. 145–150, 1982.

[11] Q. Wang, J. D. Schutter, W. Witvrouw, and S. Graves, "Derivation of compliant motion programs based on human demonstration," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3, 1996, pp. 2616–2621 vol.3.

[12] Q. Wang and J. D. Schutter, "Towards real-time robot programming by human demonstration for 6d force controlled actions," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 3, 1998, pp. 2256–2261 vol.3.

[13] F. Nagata, K. Watanabe, K. Kiguchi, K. Tsuda, S. Kawaguchi, Y. Noda, and M. Komino, "Joystick teaching system for polishing robots using fuzzy compliance control," in *Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on*, Aug. 2001, pp. 362–367.

[14] 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," Aug. 2007. [Online]. Available: http://www.i-joe.org/ojs/viewarticle.php?id=148

[15] J. Pires, T. Godinho, and R. Araújo, "Using digital pens to program welding tasks," *Industrial Robot: An International Journal*, vol. 34, pp. 476 – 486, 2007.

[16] E.-S. Choi, W. Chang, W.-C. Bang, J. Yang, S.-J. Cho, J.-K. Oh, J.-K. Cho, and D.-Y. Kim, "Development of the gyro-free handwriting input device based on inertial navigation system (ins) theory," in *SICE 2004 Annual Conference*, vol. 2, 2004, pp. 1176–1181 vol. 2.

[17] J. Rutgeerts, P. Slaets, F. Schillebeeckx, W. Meeussen, W. Verdonck, B. Stallaert, P. Princen, T. Lefebvre, H. Bruyninckx, and J. D. Schutter, "A demonstration tool with kalman filter data processing for robot programming by human demonstration," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 3592–3597.

[18] K. Nissum, T. Larsen, O. Madsen, and H. Nielsen, "Virtual 3d environment for planning robotic paint routes," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 4776–4781.

[19] M. H. Choi and W. W. Lee, "A force/moment sensor for intuitive robot teaching application," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4, 2001, pp. 4011–4016 vol.4.

[20] Y. Maeda, T. Ushioda, and S. Makita, "Easy robot programming for industrial manipulators by manual volume sweeping," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, 2008, pp. 2234–2239.

[21] M. Doulis, V. Zwimpfer, J. Pfluger, A. Simon, C. Stern, T. Haldimann, and C. Jenni, "Spaceactor - interface prototypes for virtual environments," in *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on*, 2006, pp. 171–174.

[22] B. Hein, "Me and the robot," Dec. 2007. [Online]. Available: http://www.youtube.com/watch?v=V0vBIuEgprg

[23] A. Sears and B. Shneiderman, "High precision touchscreens: Design strategies and comparisons with a mouse," *International Journal of Man-Machine Studies*, vol. 34, pp. 593–613, 1991. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.30.8594

[24] H. Bruyninckx, "The orocos project — smarter control in robotics & automation!" [Online]. Available: http://www.orocos.org/