Automatic Programming and Control for Robotic Deburring

Julian Ricardo Diaz Posada*, Shivaram Kumar, Alexander Kuss, Ulrich Schneider, Manuel Drust, Thomas Dietz, Alexander Verl†

*Fraunhofer Institute for Manufacturing Engineering and Automation (IPA), Nobelstr. 12, 70569 Stuttgart, Germany,
E-Mail: Julian.Diaz.Posada@ipa.fraunhofer.de

†Institute for Control Engineering of Machine Tools and Manufacturing Systems (ISW), University of Stuttgart, Germany

Abstract

In the current industrial scenario, robots are rarely used in contact operations such as machining and finishing as they entail complex programming and control methods. Further, the disparity between accuracy specifications, communication technologies and control methods required for such operations calls for greater efforts in robot programming and control. This paper presents a novel approach to automatically program an industrial robot-based on the CAD model of the product variants and to enable online control to minimize errors during a deburring process. The paper starts with the modeling of the product, process and resource (PPR model) which is used to generate robot motion trajectories taking into account the constraints and the free degrees of freedom (DoFs) of the robotic deburring process. The operator selects the edge of the workpiece to be machined, and an automatic program generation system is designed which programs the robot for the deburring process and enables online compensation. A laser scanner sensor device is used for localizing the workpiece in the robot cell and in the online compensation to perform fine corrections of the robot's movement during the process. Experimental results are used to validate the robotic program generation and control mechanism for a deburring process, and to outline the future potential of this work.

1 Introduction

Robots are low cost, reliable and flexible manufacturing solutions for industrial automation, where the lot size is small and the variation of profile geometries is high. While recent statistical data from the International Federation of Robotics [1] shows a definite increase in the total number of produced robots, the contribution of robots in machining and finishing applications is still at a minimum of 2 percent. Robots in manufacturing possess advantages over conventional tool machines when comparing dexterity over extended workspaces and the changeability to new processes and products [2] but are not adopted due to the many challenges faced. A study [3] on challenges and obstacles in robot-machining identifies complex programming methods, poor accuracy and insufficient rigidity as the three main obstacles and discusses some of the research efforts being taken to overcome them. The focus of the work presented in this paper is on minimizing the complexity of robot programming and online compensation of path errors for a robotic deburring process.

Existing robot programming trends [4] can be roughly classified into online and offline programming methods. Online programming methods such as Lead-Through programming or Teach-In programming are simple and easy to perform for repetitive tasks, but they are often time consuming and not easily adaptable to new or additional tasks. With higher complexity of the robot tasks, online programming becomes infeasible as the overall production cost increases. Offline programming [5] offers a solution to this problem by simulating the process and robot cell using software tools and generating robot programs offline without any loss in production time. A user with expert knowledge in CAD and CAM software tools and the process can design the desired robot task offline, simulate it and then use the tools to generate robot ready trajectories. Current research in robot programming methods try to reduce the complexity in offline programming methods, to enable the process experts with less or no experience in offline CAD/-CAM tools to easily reconfigure the robot system to suit the process requirements [6]. Key research in this direction includes Programming by demonstration (PbD) [7] where the process is demonstrated by the operator using gestures and movements which are processed by offline software tools for program generation. Commercially available offline programming tools are expensive and are aimed at robot applications which do not require frequent reconfigurations. Recent research [8] finds that an offline program generation system which incorporates human inputs, sensor information and CAD models can effectively reduce the overall programming effort. An offline programming system which uses these inputs to easily reconfigure the robot program is discussed in this paper.

For robot-based manufacturing applications, meeting high accuracy standards is one of the major obstacles. Robot calibration methods are extensively used in order to im-

prove the robot accuracy [9]. Co-ordinate measurement machines, touch probes and measuring arms [9] are some of the commonly used tools to perform robot calibration. The calibration process is inherently tedious and repetitive and the accuracy of the calibration is completely reliant on the equipment and calibration method employed. Apart from the inherent manufacturing inaccuracies, geometric and non-geometric errors also contribute to the poor accuracy of robots [10]. Due to a number of factors contributing to the inaccuracies and the high cost involved in complete cell calibration and robot modeling, end-effector pose measurements based on calibration are effective for a reconfigurable workcell [11].

The calibration process can be used to generate a static kinematic model of the robot to reduce the position errors in static positions. In a contact operation such as deburring, in order to compensate errors due to robot interaction with tool and workpiece, one can use either offline or online compensation strategies or a combination of both. While offline compensation requires an accurate process model, the online compensation method requires integrating sensor systems to measure errors in real-time.

Offline compensation methods use a combination of robot model and a machining process model to modify the robot tool path offline for machining [12]. On the other hand, online compensation methods compensate the robot path errors or interaction forces during the process, based on whether position or force is measured [13]. To minimize the modeling efforts, a visual servo control based approach using a 2D laser measurement system is implemented in this research for an online compensation.

The paper starts with a brief description of the robot cell in **Section 2**. The calibration of the tool and sensor devices is explained in **Section 3**. **Section 4** describes the localization of the workpiece in the robot cell. The design of the program generation system and the individual components, deburring process model, user desired deburring path and sensor model measurements as inputs are introduced in **Section 5**. Further, in **Section 6**, the control and compensation scheme is designed and implemented. The program generation system and the online control mechanism with the control loop is validated with an exemplary workpiece.

2 Description of the Robot Deburring System

The architecture of the system is designed to reduce programming and set-up times by modeling the task and the process model. The robotic deburring process is semantically modeled into a system software using the AML (Automation Markup Language) [14] and the XML (eXtensible Markup Language). Based on the task description, the robotic deburring process is interpreted by the motion generation software component to generate deburring paths constraining the motions based in end-user inputs. The motion generation component is explained later in **Section 5.2**. The software architecture of the robot system facilitates the visualization of the robot cell and the interaction with the workpiece. This software serves also as an interface for allowing manipulation of the robot while automatically programming the manufacturing task. The system architecture of the robot machining system introducing the individual components and their interactions is shown in the figure below.

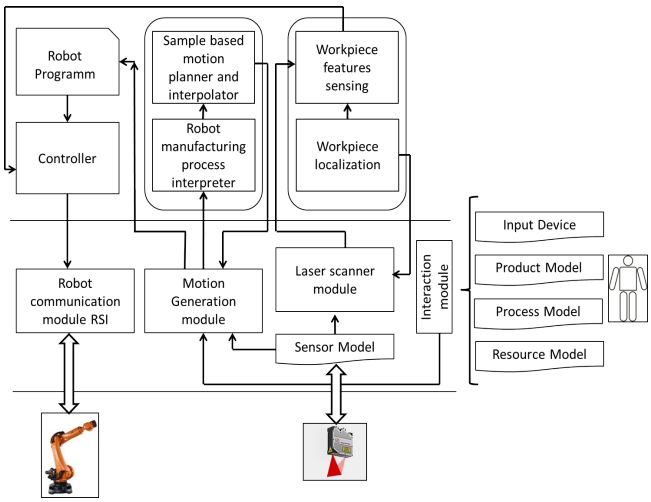

Figure 1 System Architecture of the Deburring System

The robotic deburring system consists of a Kuka Quantec KR270 industrial robot manipulator, a Biax pneumatic spindle deburring tool with adjustable deflection and a 2-D laser scanner, scanControl 2900-50 from the company Micro-Epsilon mounted on the robot flange (**Figure 2**). The deburring process is completed in two stages. In the first stage, a robot program is automatically generated from a combination of the three input components: task specifications given by the human, sensor information and CAD models. In the second stage, an online compensation algorithm is executed during the deburring process. The spindle allows the configuration of different stiffness and deflections providing the system with a range of tolerance when deburring the workpiece. The laser scanner provides the robot system with measurements of the workpiece to compensate the robot using a control system.

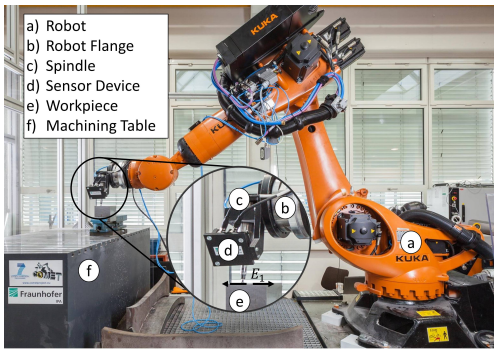

Figure 2 Robot deburring system

3 Tool and Sensor Calibration

The calibration procedure developed to determine the Tool Centre Point (TCP) frame and the sensor frame are described in this section.

3.1 TCP Calibration

The TCP calibration is carried out using a Leica AT901 laser tracker measurement system (its coordinate system is notated as TRACKER). The tracker system follows reflective targets giving position measurements along the X, Y and Z-axes. The tracker targets are fixed to the robot end-effector and repeated position and orientation measurements along the axes, taken one at a time, are used to record circle and plane features of the flange (FLANGE) and the TCP illustrated in the figure below.

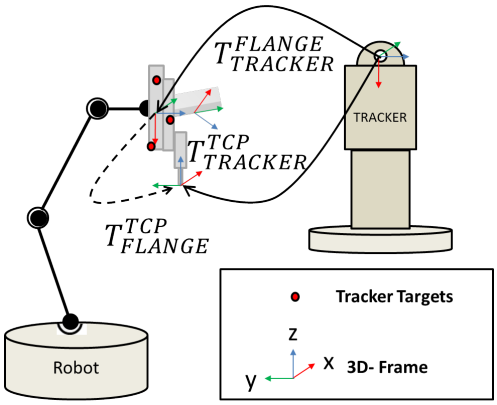

Figure 3 Transformations between TCP and Flange of the robot

The measured geometrical features are used to determine the position and orientation of the frame at the flange and the TCP. The flange position and orientation in reference to the tracker are given by $T^{FLANGE}_{TRACKER}$, while tool position and orientation in the tracker frame are given by $T^{TCP}_{TRACKER}$. The required calibrated transformation of the tool in the flange frame T^{TCP}_{FLANGE} is given by the following equation,

$$T^{TCP}_{FLANGE} = T^{FLANGE\ -1}_{TRACKER} T^{TCP}_{TRACKER} \tag{1}$$

3.2 Sensor Calibration

The calibration of the sensor frame (SENSOR) for the 2-dimensional laser scanner with respect to the flange co-ordinate system denoted as T^{SENSOR}_{FLANGE} is performed using a novel approach. In this approach, multiple laser scanner measurements of a calibration structure along with the corresponding robot flange position are measured to compute this transformation. Each laser scanner measurement, composed of seven sensed lines of the calibration structure (refer **Figure 4**), is used to define the reference coordinate system of the calibration structure, denoted as (CS). The co-ordinate system CS is required to obtain the transformation from the sensor frame to the calibration structure T^{CS}_{SENSOR}. The robot sensor recording the consecutive line segments from the calibration structure and a sample recorded measurement is shown in the figure below.

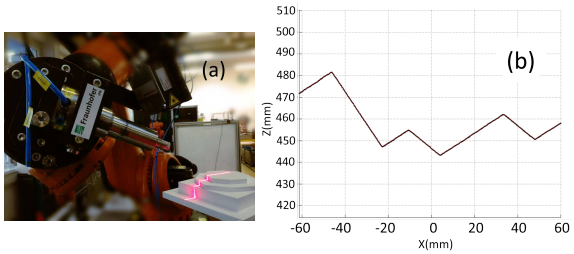

Figure 4 Left: Measurement of calibration structure features and Right: recorded feature measurement

Moreover, in each measurement, the transformation robot base (ROBOTBASE) to flange $T^{FLANGE}_{ROBOTBASE}$ is recorded. A closed kinematic equation from robot base to calibration structure and back to robot base is illustrated as follows.

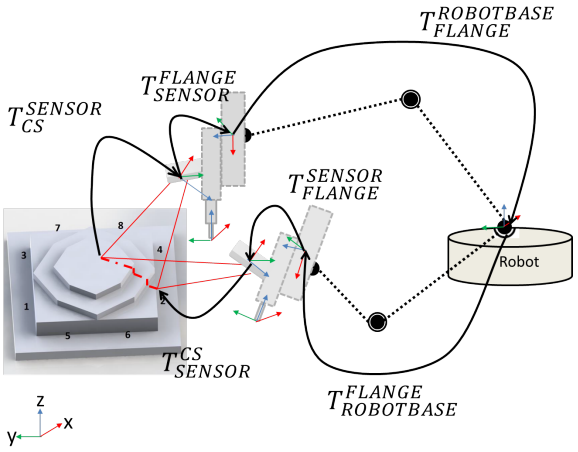

Figure 5 Transformations involved in the sensor calibration method

The closed kinematic equation obtained from these measurements is used to compute the flange to sensor frame transformation T^{SENSOR}_{FLANGE} with i and j as parameters for indexing different measurements as follows,

$$\begin{aligned} TF(i,j) = [T^{SENSOR}_{CS_i} \cdot T^{FLANGE}_{SENSOR}(\beta) \cdot T^{ROBOTBASE}_{FLANGE_i}] \cdot \\ [T^{FLANGE_j}_{ROBOTBASE} \cdot T^{SENSOR}_{FLANGE}(\beta) \cdot T^{CS_j}_{SENSOR}] \\ where, \quad \beta = \begin{bmatrix} x & y & z & a & b & c \end{bmatrix} \end{aligned} \tag{2}$$

The kinematic equation is optimized by using the Levenberg-Marquardt algorithm within different combinations of measurements. Five optimization loops are programmed. The first loop, to optimize weight factor selection of the Levenberg-Marquardt algorithm, in the second, to select the initial parameters of vector β, in the third loop, to optimize translational components, in the fourth loop, to optimize rotational components and finally to optimize all the parameters of the flange to sensor frame transformation

T_{FLANGE}^{SENSOR}. The Levenberg-Marquardt optimization problem is defined for the obtained set of measurements, optimization loops and dependent values $[TF(i,j),\emptyset]$ optimizing the parameters of β from the model curve $f(TF(i,j),\beta)$ so that the sum of the deviations becomes minimal in the following equation,

$$S(\beta) = \sum_{k=1}^{m} [\emptyset - f(TF(i,j),\beta)]^2. \quad (3)$$

4 Localization of workpieces

A basis for the proposed online sensor compensation is the knowledge about the coarse workpiece location. This information can be obtained by online teaching with the robot end effector. However, in the case of changing product variants teaching can be a time consuming task. To overcome this drawback, we integrate an automatic workpiece localization based on optical sensor measurements. The 2D sensor mounted on the robot end effector (see **Section 2**) is moved over the workpiece geometry as shown in the figure (refer **Figure 6 (a)**). Information about the robot positions $T_{FLANGE}^{ROBOTBASE}$ retrieved from the robot controller and the sensor transformation T_{SENSOR}^{FLANGE} are used to transform the 2D sensor data in 3D space resulting in a 3D point cloud of the measured workpiece. The measured geometry is represented by a sensor point cloud $S = \vec{i}_k$ with the points $\vec{r}_i$ for $i = 1,...,N_S$. The workpiece CAD model is used to generate a corresponding reference point cloud $R = \vec{r}_j$ of the workpiece geometry with the points $\vec{r}_j$ for $j = 1,...,N_R$. Finally, a point cloud matching is performed between the S and R using the generalized ICP algorithm by [16]. It minimizes the sum of squared distances between corresponding points of two point clouds in an iterative process taking into account locally planar structure. The result of the point cloud matching is shown in the (**Figure 6 (b)**). The resulting transformation $T_{ROBOTBASE}^{WP}$ can then be used for coarse workpiece localization in the offline planning environment. As the reference point cloud R is derived from the nominal CAD model of the workpiece, it does not account for geometric deviations of the real workpiece geometry. This results in inaccuracies of the workpiece localization. However, in combination with the online sensor measurements proposed in this paper, these inaccuracies can be compensated.

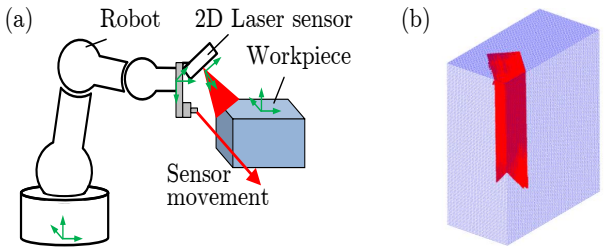

Figure 6 (a) Sensor movement for 3D workpiece localization and (b) Localization result between reference point cloud R (blue) and sensor point cloud S (red).

5 Automatic Program Generation System

In this section, the motion generation system used to generate robot path points is explained followed by the design and implementation of the automatic program generation system.

5.1 State Machine Based Deburring Process Model

The kinematic arrangement of the sensor with the tool in the current setup makes the sensor traverse a path ahead of the tool. The kinematic relationship between the tool and sensor frame is used to generate an offset path and a mapping distance (k) between the sensor measurement frame and the tool frame for workpiece variants. Subsequently, the entire path is modeled into four process states (Sensing, Deburring, Stop Sensing, Stop Deburring) as shown in the figure below.

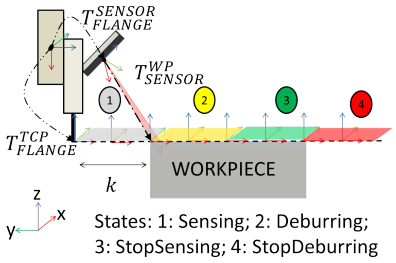

Figure 7 Deburring process states

A state machine model ensures a reliable system by restricting the system to be at only one of the four states at any point of time. The individual states and their descriptions are elaborated in the table below.

Table 1 Summary of States

States	Description
Sensing	The tool is not in contact with the workpiece. The robot system is ready to receive sensor measurements and compute the tool position
Deburring	The tool is in contact with the workpiece. The compensation is active and simultaneously sensor measurements are recorded
StopSensing	The tool is in contact with the workpiece. The sensor measurements are not recorded but the compensation mechanism is active
StopDeburring	The tool is not in contact with the workpiece and the sensor system is also in a deactivated state

An initial path over the workpiece is computed by interpreting the DoFs and constraints of the process by optimal

motion generation explained in **Section 5.2**. The four process states and the robot system behavior is then assigned to the generated robot path points.

5.2 Motion Generation in Workpiece for Robotic Deburring

A path over the workpiece without taking into account the sensor model is initially computed by interpreting the DoFs and constraints of the deburring manufacturing process using the motion generation methods proposed by the authors [15] taking into account the user specified product, process and resource models (**Figure 1**). The robot deburring cell is simulated with the CAD files using the PPR model described using the Automation Markup Language (AML) [14]. The process is modeled into an eXtensible Markup Language (XML) file also linked to the AML. It describes the geometrical transformation from workpiece to process and specifies the DoFs and constraints of the process. For the deburring process, the rotation around the tool axis (rotation around Z-axis) is defined as a DoF as specified in the following listing,

Listing 1 Semantic-modeling of the deburring process in XML

```
<Process name="Deburring">
    <FrameConstraintsDoF>
        <GeoPar name="x" unit="mm" type="Fixed" value="0"/>
        <GeoPar name="y" unit="mm" type="Fixed" value="0"/>
        <GeoPar name="z" unit="mm" type="Fixed" value="0"/>
        <GeoPar name="a" unit="deg" type="Fixed" value="0"/>
        <GeoPar name="b" unit="deg" type="Fixed" value="0"/>
        <GeoPar name="c" unit="deg" type="Range" min="-45" max="45" value="0"/>
    </FrameConstraintsDoF>
</Process>
```

Based on user specifications over the simulated workpiece, desired start and end positions are defined as $\vec{P}_{S_n}$, $\vec{P}_{E_n}$ respectively, where n denotes the number of selected edges. Orientations are defined taking into account a defined product convention denoted with quaternions $\underline{q}_n$ which are given by the interaction mode (refer **Figure 1**) to the motion planner. The workpiece edge positions are then parameterized with the parameter l and are specified in the homogeneous matrix transformation from robot base to workpiece notated in (**Equation 4**). In order to generalize motion generation for multiple edges, the rotation between them is calculated by finding the required transformation between the quaternions and evaluating the axis-angle convention $[\alpha_n, \vec{v}_n] = Q2AA((\underline{q}_n)^{-1} \cdot \underline{q}_{[n+1]})$ and parameterizing it taking into account a rotation factor over l notated ζ as in (**Equation 5**).

$$T^{WP}_{ROBOTBASE}(l) = \begin{bmatrix} R_q(\underline{q}_n) & \vec{P}_{S_n} + (\frac{l}{100})(\vec{P}_{E_n} - \vec{P}_{S_n}) \\ 0 & 1 \end{bmatrix}, \quad l \in [100n, 100(n+1)]. \tag{4}$$

$$\beta(l,n) = \alpha_n \cdot \left(\frac{l - (100n - \zeta)}{2\zeta} \right). \tag{5}$$

The process model is mathematically denoted using the semantical description (refer **Listing 1**) in the transformation from workpiece to manufactuing process as denoted in (**Equation 6**). The process and product description are embedded into the Open Motion Planning library [17]. The parameters l and δc (rotation around Z-axis, represented with the variable c) are specified as the DoF of the sample based motion planner [18] and an Optimal Rapidly exploring Random Tree algorithm (RRT*) [17] is used for finding optimal way to rotate the end-effector for the deburring process. **Figure 8** depicts the sampled states in the Robot Manufacturing Process space in which the DoFs are configured, the Cartesian space positions depicting the path σ^*_m (where m represent the number of poses in this path defined by the end user) and the robot joint movements for reaching the deburring of two orthogonal edges. Moreover, the robot sequences obtained from the simulation are also depicted.

$$T^{PROCESS}_{WP}(\delta c) = \begin{bmatrix} R_{eul} \begin{pmatrix} a \\ b \\ c + \delta c \\ 0 \end{pmatrix} & \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \end{bmatrix} \tag{6}$$

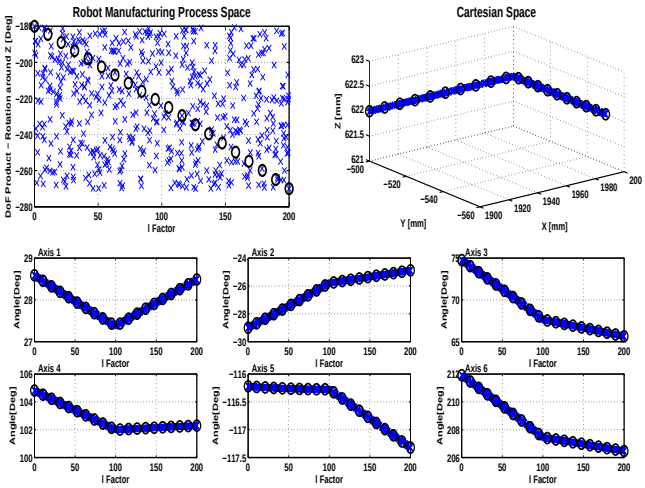

Figure 8 Evaluated samples and interconnected solution in the Robot Manufacturing Process space, Cartesian and joint spaces for robotic deburring using the rotation around the tool vector with the found motion

5.3 Sensor Model

Each of the m poses of σ^*_m calculated in **Section 5.2** is related to the sensor model by further interpreting the constraints of the workpiece and the manufacturing process. To achieve this for linear edges, the mapping distance k is mathematically calculated constrained with the parameter l, to map the tool position information with that of the workpiece edge for sensing and later compensating the robot during the deburring process. In order to determine the transformation matrix for the expected sensor measurements, a closed kinematic chain between base-tool-sensor-workpiece is taken. The end transformation from robot

base to expected sensor measurements on the workpiece is denoted as $T_{ROBOTBASE}^{WP_{EXP\_MEAS}}$, the transformation from robot base to TCP is denoted as $T_{ROBOTBASE}^{TCP}$, the transformation from tool to sensor is denoted as T_{TCP}^{SENSOR} and the transformation from sensor to expected measurements is denoted as $T_{SENSOR}^{WP_{EXP\_MEAS}}$. The equation describing the kinematic chain with these transformations is derived below,

$$[T_{ROBOTBASE}^{WP_{EXP\_MEAS}}] = [T_{ROBOTBASE}^{TCP}] \cdot [T_{TCP}^{SENSOR}] \cdot [T_{SENSOR}^{WP_{EXP\_MEAS}}] \tag{7}$$

From the calibration process (see **Section 3**), the transformation to the tool and the sensor with respect to the robot flange is determined. Thus the transformation matrix from tool to the sensor frame T_{TCP}^{SENSOR} is calculated as,

$$T_{TCP}^{SENSOR} = {T_{FLANGE}^{TCP}}^{-1}\, T_{FLANGE}^{SENSOR} \tag{8}$$

The TCP pose in robotbase is then constrained using the parameterization of the edge using the defined parameter l in (**Equation 4**). With this procedure it is assured that the the robot moves during the sensing state over the edge. The rotational component $\underline{q}_n$ is known in advance from the computed path σ_m^*. In the equation below this constraint is expressed for each of the translational components,

$$T_{ROBOTBASE}^{TCP} = \begin{bmatrix} R_q(\underline{q}_n) & \begin{pmatrix} x_{TCP}(l) \\ y_{TCP}(l) \\ z_{TCP}(l) \end{pmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{9}$$

The transformation sensor to expected measurement is also defined and the translation components for x and z components are also unknowns (y is null because it is a 2 dimensional sensor) as follows,

$$T_{SENSOR}^{WP_{EXP\_MEAS}} = \begin{bmatrix} R^{wp_{exp\_smeas}} & \begin{pmatrix} x_{SENSOR} \\ 0 \\ z_{SENSOR} \end{pmatrix} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{10}$$

By substituting (**Equation 8**), (**Equation 9**) and (**Equation 10**) into (**Equation 7**) and equaling this to the first computed pose in the workpiece (σ_1^*) a linear system of three equations eq_1, eq_2, eq_3 with three variables (l, x_{SENSOR} and z_{SENSOR}) is automatically built and solved by using a Python embedded tool into the CAM software as follows,

$$\begin{bmatrix} x^{\sigma_1^*} \\ y^{\sigma_1^*} \\ z^{\sigma_1^*} \end{bmatrix} = \begin{bmatrix} eq_1(x_{SENSOR}, z_{SENSOR}, x_{TCP}(l)) \\ eq_2(x_{SENSOR}, z_{SENSOR}, y_{TCP}(l)) \\ eq_3(x_{SENSOR}, z_{SENSOR}, z_{TCP}(l)) \end{bmatrix} \tag{11}$$

The tool position is offset along the machining edge with the solved l and in this way its relation to the Cartesian space is solved for determining the mapping distance k.
Then using k, a sensor path is calculated for each of the poses of σ_m^* obtaining in this way the sensing path denotated as σ_m^{Sens} which relates the machining points with the sensing points. In order to map the tool position with the expected measurements, a 6-D frame containing position and orientation information of the workpiece is constructed by taking into account the workpiece and the sensor CAD models using a triangular mesh. The intersection points of the workpiece edge and projected sensor scanned lines are identified using a triangle intersection algorithm. This algorithm based on the CAD objects of the workpiece is used to calculated the expected sensor measurements over the sensor path. An illustration of the scan lines onto the workpiece edge and the intersection points on the workpiece edge is shown below.

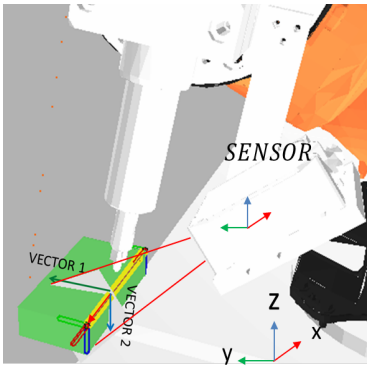

Figure 9 Expected Measurement calculation

From the points lying on the surface of the workpiece along the scan lines, the edge points of the workpiece are determined. The Z and Y-axes vectors are defined on the workpiece surface with the edge point and other surface points along the scan line on the two workpiece faces as shown in **Figure 9**. The frame on the edge of the workpiece is constructed using these two vectors and its normal vector calculated as X-axis depicted by red arrow. A similar algorithm is used to interpret the real measurements of the sensor.

5.4 Algorithm for Program Generation

The automatic program generation system is initialized with the localization of the workpiece as in **Section 4**. With the obtained workpiece edge points, a set of robot poses σ_m^*, defined in tool frame, for the deburring process are generated as in, **Section 5.2**. Based on the sensor model, a list of poses defined in sensor frame σ_m^{Sens} is computed in **Section 5.3**. The deburring process is modeled in the program generation system into the four states introduced in **Section 5.1**. Based on the generated robot poses over the workpiece (σ_m^*), the sensor modeled pose (σ_m^{Sens}) with respect to the workpiece (refer **Figure 7**), each robot pose is assigned to one of the four states.
The robot behaviour described during the different states is generated in the form of a robot program code as shown in the figure below.

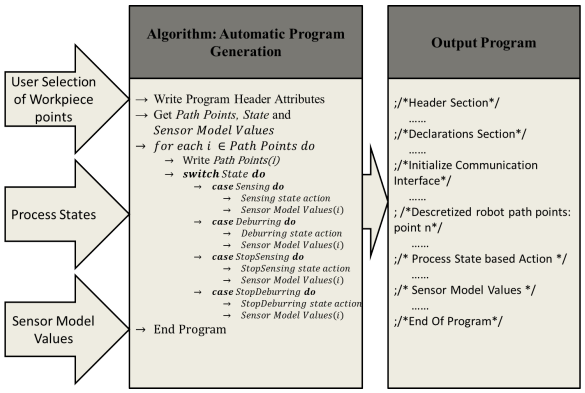

Figure 10 Automatic program generation algorithm

6 Compensation and Control

In this section the design and implementation of the controller and the compensation scheme is explained. Moreover, results from experimental compensation are depicted.

6.1 Design and Implementation

A control loop with a manually tuned Proportional-Integral (PI) controller is designed in a Programmable Logic Controller (PLC) to compensate the robot path errors online. The information flow between the sensor device, robot system and the controller is illustrated in (**Figure 11**). The disparity in the communication cycletimes between the different components (robot controller, PLC, sensor device) is overcome by imposing hard real-time constraints into the PLC and the robot controller, with a cycle time of 12 ms.

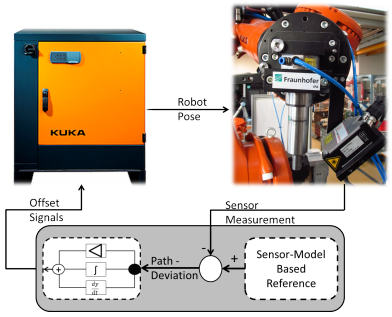

Figure 11 Online path compensation scheme

The model of the robot manipulator is simulated in the control loop (refer **Figure 11**) using a PT1 delay element. A PT1 is a single order delay element with its transfer function $G(s)$ defined as,

$$G(s) = \frac{K_1}{(1+T_1 s)}. \tag{12}$$

A controller to trace the references is designed using the PI components with a transfer function $C(s)$ defined as,

$$C(s) = K_p\left[1 + \frac{1}{(1+T_n s)}\right], \tag{13}$$

The PT1 model system is parameterized with the robot time constant $T_1(ms)$, controller time constant T_n, robot gain K_n and proportional gain of controller K_p. These values are further tuned to get the optimal system response. The PI controller with an integral time T_n =15 ms and a having a proportional gain of $K_p = 0.8$ is chosen for further experimentation. The controller is tuned to perform finer compensations (in the order of 1 mm), assuming that the real workpiece geometry matches approximately with that of the CAD geometry. The errors given to the controllers, in this case deltas in Y and Z-axes in the TCP frame, are calculated into the PLC by computing the transformation between the expected and desired sensor measurements in the TCP frame as follows,

$$T^{WP_{MEAS}}_{WP_{EXP\_MEAS}} = (T^{SENSOR}_{TCP} \cdot T^{WP_{EXP\_MEAS}}_{SENSOR})^{-}1 \cdot (T^{SENSOR}_{TCP} \cdot T^{WP\_MEAS}_{SENSOR}). \tag{14}$$

6.2 Experimental Compensation

The automatic program generation system is validated and the control system performance is analyzed for an edge of a cuboid workpiece. The workpiece localization is performed as described in **Section 4**. A program is automatically generated using the motion planning algorithms, state machine and sensor model as explained in **Section 5**. The errors are computed as in **Equation 14** and compensated by the PI controllers in TCP frame. The compensation action at the different control points of the deburring path is illustrated in the figure below.

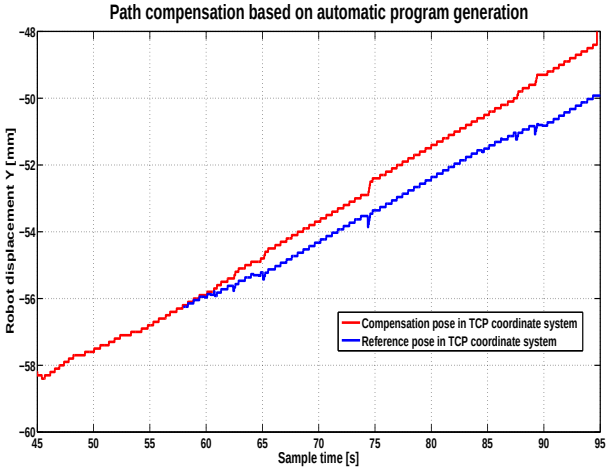

Figure 12 Path compensation based on automatic program generation

The reference values are set at the control points using the expected sensor measurements of the workpiece. Small variations on these measurements are due to the sample based nature of the motion planner. The start of stage two can be noticed when both paths start. Close matching at this point is due to the localization algorithm. Proper response of the system is noticed. Future work is testing the control response when the process dynamics is also involved.

7 Conclusions

In SMEs where the lot size is low and the variation between components is high, robot systems which can be easily reconfigured to new processes and products will play a key role. To make up for this deficiency, an intuitive programming system for faster reconfiguration of robots by generating robot ready programs with minimal steps is presented in this paper.

The novelty of this approach lies in encapsulating firstly, the motion generation with its constraints and free degrees of freedom derived from the product, process and resource models. Secondly, the process model states. Thirdly, the control information for online path compensation within the robot program, obtained from CAM computations in which the sensor model is embedded. And finally, the user input. By performing an online compensation, the requirements of repetitive calibrations, tedious robot modeling process and teach-in processes can be reduced.

Future work of the presented approach is related to the extension of the sensor models in order to allow computation of expected measurements in CAM and sensing of more complex geometries in workpieces. Moreover, the implementation and evaluation of different sensors by changing the sensor model into the architecture and adapting them into the robot cell could be approached. Furthermore, the cognitive abilities of robot systems could be increased by optimizing robot motions by using the free degrees of freedom interpreted from the manufacturing process and the sensor constraints when measurement of complex workpiece features is required.

8 Acknowledgments

The authors would like to acknowledge Mr. Arjun Sridhar, Mr. Pedro Rosales and Mr. Tae Hun Lee for their contribution on the implementation of some functionalities described in this paper and also Mrs. Luzia Schuhmacher for proofreading this paper. The research leading to these results was supported and financed by Baden-Württemberg Stiftung (ROB-8).

References

[1] International Federation of Robotics - Statistical Department, “World robotics 2014 industrial robots,” 2014. [Online]. Available: http://www.worldrobotics.org/

[2] Jayaweera, N.: Webb, P.: Robotic edge profiling of complex components, Industrial Robot, Vol. 38, No. 1, pp. 38-47, 2011.

[3] Karim, A.: Verl,A.: Challenges and Obstacles in robotic-machining, 44th International Symposium on Robotics (ISR), pp. 1-4, 2013.

[4] Pan, Z.: Joseph, P.: Larkin, N.: van Duin, S.; Norrish, J.: Recent Progress in Programming Methods for Industrial Robots, 41th International Symposium on Robotics (ISR), pp. 1-8, 2010.

[5] Vuong, N.D.: Lim, T. M. L.: Yang, G.: Simulation and Offline Programming for Contact Operations, Springer London, 2015.

[6] Leali,F.: Pini, F.: Ansaloni, M.: Integration of CAM off-line programming in robot high accuracy machining, International symposium on System integration IEEE/SICE, pp. 580-585, 2013.

[7] Billard, A.: Calinon,S.: Dillman, Rüdiger: Schaal, S.: Robot programming by demonstration, Springer handbook of robotics, Springer, pp. 1371-1394, 2008.

[8] Dietz, T.: Schneider, U.: Barho, M.: Oberer-Treitz, S.: Drust, M.: Hollmann, R.: Hägele, M.: Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments, ROBOTIK 2012, VDE-Verlag., pp. 428-433, 2012.

[9] Elatta Y, A.: Gen, Li Pei: Zhi, Fang Liang: Daoyuan, Yu: Fei, Luo: An Overview of Robotic Calibration, Information Technology J., Vol. 3, pp. 74-78, 2004.

[10] Schneider, U.: Drust, M.: Ansaloni, M.: Lechmann, C.: Pelliciari, M.: Leali, F.: Gunnink, J.W.: Verl, A.: Improving robotic machining accuracy through experimental error investigation and modular compensation, Advanced Manufacturing Technology J., pp. 1-13, 2014.

[11] Leali, F.: Vergnano, A.: Pini, F.: Pellicciari, M.: Berselli, G.: A workcell calibration method for enhancing accuracy in robot machining of aerospace parts, Advanced Manufacturing Technology J., pp. 1-9, 2014.

[12] Slavkovic, N. R.: Milutinovic, D.S.: Glavonjic, M. M.: A method for off-line compensation of cutting force-induced errors in robotic machining by tool path modification, Advanced Manufacturing Technology, Springer, Vol. 70, pp. 2083-2096, 2014.

[13] Samad, T.: Annaswamy, A.: "The impact of control technology" [Online]. Available: http://ieeecss.org/general/impact-control-technology

[14] Drath, R.: Lüder, A.: Peschke, J.: Hundt, L.: AutomationML - the glue for seamless automation engineering: Conference on emerging Technologies and Factory Automation ETFA, IEEE, pp. 616-623, 2008.

[15] Diaz Posada, J. R.: Dietz, T.: Ockert, P.: Kuss, A.: Hägele, M.: Verl, A.: Automatic Optimal Motion Generation for Robotic Manufacturing Processes: Optimal Collision Avoidance in Robotic Welding, 12th IEEE Conference on Automation Science and Engineering CASE, "submitted to peer review", 2016.

[16] Segal, A.: Haehnel, D.: Thrun, S.: Generalized-ICP, Robotics: Science and Systems, Vol. 2, No. 4, 2009.

[17] Karaman, S.: Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning, Int. J. Rob. Res., Vol. 30, pp. 846-894, 2011.

[18] LaValle, S.M.: Motion Planning: The Essentials, IEEE Robotics and Automation Society Magazine, Vol. 18, pp. 79-89, 2011.