MODELING, IDENTIFICATION AND CONTROL, 1993, VOL. 14, NO. 2, 93–105
doi:10.4173/mic.1993.2.4

Automatic programming of grinding robot

TRYGVE THOMESSEN†, OLE JAKOB ELLE, JON LUND LARSEN, TORGRIM ANDERSEN, JAHN E. PEDERSEN and TERJE K. LIEN

Keywords: *Robotic grinding, force control, speed control, automatic contour tracking, automatic path generation*

A new programming method is developed for grinding robots. Instead of using the conventional jog-and-teach method, the workpiece geometry is automatically scanned by a contour tracking system. During tracking of the workpiece contour, the robot position is continuously logged. Finally a robot program is automatically generated.

This system is implemented and tested on a MultiCraft 560 robot. Experimental results show that the programming time is considerably reduced when applying this system. Thereby the robot can quickly and easily be reconfigured for working on different products. This makes it profitable to apply grinding robots in companies with small batch sizes.

In addition, the programming is simplified. Thus, the people in industry can do the programming work without needing help from robot experts.

1. Introduction

Programming grinding and deburring robots is usually time consuming. This is because the robot's tool is in physical contact with the workpiece. In addition, the workpiece geometry is often complicated. When developing new programs, one has to do a lot of programming and testing work before the program is running satisfactorily.

Several scientists have worked on the robot programming problem. Hirzinger (1982) developed a special joy-stick for industrial robots. With this joy-stick it was possible to control all the six axes of the robot using one hand. This turned out to be an efficient way for programming robots but a lot of experience was needed to control the joy-stick.

De Schutter (1986) used automatic contour tracking in some simple deburring experiments. Programming of the robot was very simple because only the magnitude of the contact force between grinder and workpiece and desired motion speed had to be specified. The workpiece geometry did not need to be specified. Because the maximum motion speed was limited, the system could not be used in industrial applications.

Schmid (1990) developed special sensors for deburring robots. The sensors were designed with a geometry similar to the deburring tools, and they were used to improve the teach-in programming. By connecting the sensors to the robot control system, the contact forces were supervised during the programming operation.

Abele, Booley and Sturz (1985) developed a system for interactive programming of deburring robots. Some characteristic points on the workpiece first had to be programmed. Then the robot automatically tracked the contour between these points. The weakness of the system was that many characteristic points had to be programmed

Received 25 January 1993.

† SINTEF Production Engineering, N-7034 Trondheim, Norway.

Presented at the 23rd International Symposium on Industrial Robots, Barcelona 9 October 1992, Barcelona, Spain. Reprinted with permission from Asociación Española de Robótica.

if the workpiece contour was complex. In addition, tracking of the workpiece was time consuming because the tracking speed was low.

This paper presents a project at the Norwegian Institute of Technology in cooperation with the SINTEF Research Institute. A new user interface for grinding and deburring robots is being developed. The objective is to simplify the robot programming and reduce the programming time so it becomes profitable to apply industrial robots for grinding and deburring in small and medium batch sizes.

2. Conventional robot programming

The common way of programming grinding robots is the jog-and-teach approach. In Fig. 1 the robot is moved to the desired locations ($P_1, P_2, \ldots, P_N$) on the workpiece contour using joy-sticks. The coordinates to the points are stored in the robot control system. During execution of the program, the path is generated by interpolation between these points (Fig. 2).

After logging the points, the program has to be optimized. The objective is to reduce the cycle time if the final workpiece quality is satisfactory. The optimization is done by increasing the motion speed until the result is unsatisfactory. Usually the programmed points also have to be adjusted during the optimization because overshoot occurs when the acceleration forces become too large.

The advantage of the jog-and-teach approach, is that it is simple to use. Hence, the user does not need thorough knowledge of robot programming. On the other hand, the approach is time consuming. If the path is curved, it must usually be approximated by straight line segments. Hence, lots of points have to be programmed, and the operator must be experienced to determine the necessary density of points along the path.

In addition, optimization of the program often needs a lot of time due to limited tracking capabilities of the robot. This causes large set up times which implies that it is only profitable to apply the robot in grinding if lots of equal products are going to be ground.

3. Automatic robot programming

A simple and rapid way of programming a grinding robot, is to apply automatic programming. As opposed to the jog-and-teach approach, the robot automatically scans the workpiece contour to be ground. Only a few locations have to be programmed manually.

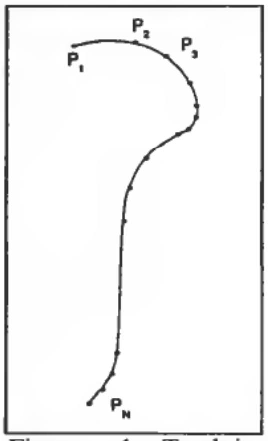

Figure 1. Teach-in points along a contour.

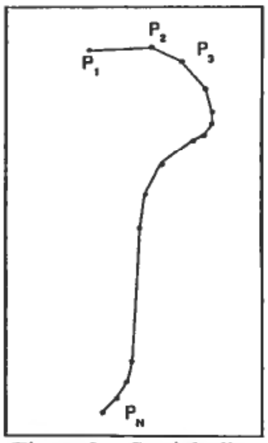

Figure 2. Straight line interpolation between teach-in points.

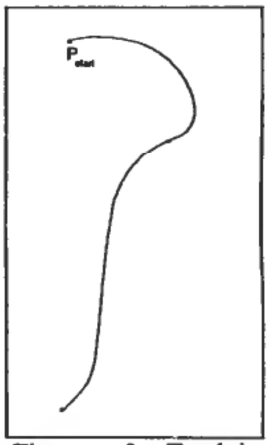

Figure 3. Teach-in point for automatic contour tracking.

3.1. *Programming strategy*

The workpiece contours to be ground are divided into contour elements. These are parts of the workpiece that are only curved about one axis. The contour of these elements is scanned using automatic contour tracking. This is shown in Fig. 3.

The robot is moved with the desired orientation to the starting point P_{start}, on the contour to be tracked.The orientation of the contour is specified with respect to either the tool coordinate system or the base coordinate system. Usually the tool coordinate system is the most convenient to use because the normal vector of the contour is parallel to one of the main axes in the tool coordinate system.

Then, the robot starts tracking the contour. The robot position is continuously stored in the control system during the tracking operation. When the whole contour has been tracked, the operator stops the robot, and a robot program on a standard format is automatically generated. A robot program also consists of motions where the robot is not in contact with the workpiece. For these motions, the conventional jog-and-teach approach can be used.

If certain locations on the workpiece are to be programmed, sensor controlled teach-in can be applied. The robot is moved towards the workpiece (like jog-and-teach). Simultaneously, the robot control system supervises the contact forces acting on the grinding tool. When physical contact comes into being, the contact force increases, and the control system stops the motion immediately. The advantage of this is that one does not have to concern oneself about damaging the grinding tool when it is close to the workpiece. The robot can be moved faster, and the risk of damaging the grinding tool is small.

The result from the teach-in operations points is stored in the control system in form of sub programs. Finally, the sub programs are linked together to a complete program for the robot.

3.2. *Automatic contour tracking*

Automatic contour tracking is a special control strategy for industrial robots. The robot tracks an unknown contour without any information about the curvature. This was developed by De Scutter (1986). The principle for automatic contour tracking can easily be explained using the following example:

If a human being closes his eyes and lets his finger follow a contour (e.g. a table edge), the following can be observed:

A certain force has to be applied against the edge to maintain physical contact.

The reaction force is approximately perpendicular to the edge. (Assuming low friction coefficient.)

The finger is moved tangential to the edge. (This is perpendicular to the contact force.)

These observations can be used as a foundation for making a control algorithm for the robot.

3.2.1. *The force control algorithm.* To apply a certain force against the edge, a force control system is necessary. The starting point to design a force control system is shown in Fig. 4. The robot is commanded to the position P_p, but because of the wall, it is forced to the position P_0.

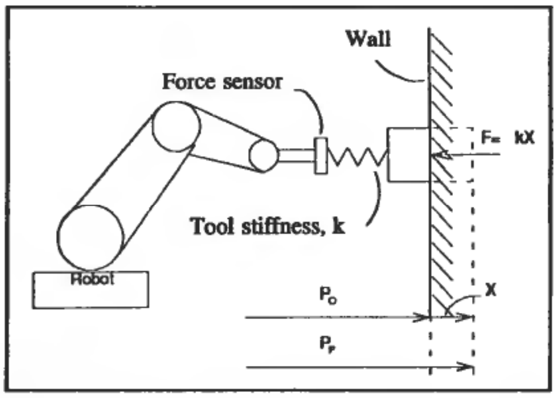

Figure 4. Robot in contact with the environment.

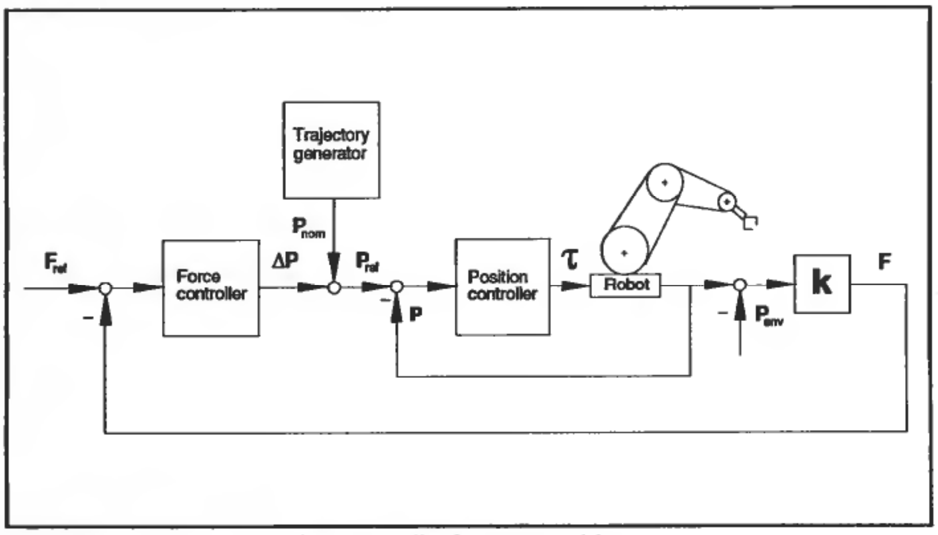

Figure 5. The force control loop.

If the tool stiffness is k, the contact force F, between the tool and the wall becomes

$$F = k \cdot x = k \cdot (P_p - P_0) \tag{1}$$

From this follows that the force control system has to regulate the robot position continuously to maintain a constant contact force. This is shown in Fig. 5. A force control loop is closed around the conventional position control loop. The difference between the reference force $\mathbf{F}_{\text{ref}}$ and the measured contact force $\mathbf{F}$ is input to the force controller. From the control deviation, the force controller which usually is an I- or PI-controller (De Scutter 1986) calculates a correction $\Delta\mathbf{P}$ to the robot's nominal position $\mathbf{P}_{\text{nom}}$. Hence, the reference to the position controller is given by

$$\boldsymbol{P}_{\text{ref}} = \boldsymbol{P}_{\text{nom}} + \Delta P \tag{2}$$

The position controller calculates the commanded motor torque $\boldsymbol{\tau}$ which moves the robot to the position **P**. Due to the position of the environment $\mathbf{P}_{\text{env}}$, the contact force **F**, comes into being. It is important to notice that the tool stiffness **k**, is included in the force feedback loop. In fact, the magnitude of **k** determines the total gain in the force control loop, and must be taken into account when designing the force controller.

3.2.2. *The automatic contour tracking algorithm.* Figure 6 shows a wheel moving along a contour. The normal vector of the contour is perpendicular to the paper plan. The object coordinate system (x_0, y_0) is fixed to the wheel with x_0 tangential and y_0 perpendicular to the contour. Assuming no friction in the wheel's bearing, the contact force F, is perpendicular to the contour and consequently, parallel to y_0. The robot has to move the wheel with a velocity v, parallel to x_0 if the wheel is going to track the contour. But the curvature of the contour is unknown which in turn implies that the orientation of the object coordinate system is also unknown.

The orientation of the object coordinate system with respect to the tool coordinate system (x_T, y_T) is given by the tracking angle α. The tool coordinate system is fixed to the robot's end effector, and its orientation is known. The contact force F, has the components F_{xT} and F_{yT} along the x_T-and the y_T-axis, respectively. Hence, the tracking angle can be calculated from

$$\alpha = \operatorname{atan} 2\left(\frac{F_{xT}}{F_{yT}}\right) \tag{3}$$

In the robot control system, the task coordinate system (x_t, y_t) is rotated the angle α, with respect to the tool coordinate system. Consequently, if α is determined accurately, the task coordinate system coincides with the object coordinate system. By giving the robot a velocity v, in the x_t-direction and a constant force F, in the force in the y_t-direction, the robot will automatically follow the contour.

The block diagram for the contour tracking system is shown in Fig. 7. Compared to the force control system (Fig. 5), the contour tracking system is extended with the transformation matrices $\mathbf{C}_t^T$ and $\mathbf{C}_T^t$. These matrices are continuously updated according to the tracking angle α, calculated by the orientation estimator. $\mathbf{C}_T^t$ transforms the force measurements from tool- to task coordinates. The measured force F_{yt} is subtracted from the reference force F_{ref}, and the force controller calculates the correction ΔP_{yt}. The desired tangential velocity v_{xt}, and the correction ΔP_{yt}, are transformed to tool coordinates (by the matrix $\mathbf{C}_t^T$) and finally fed into the position controller similar to the force control system.

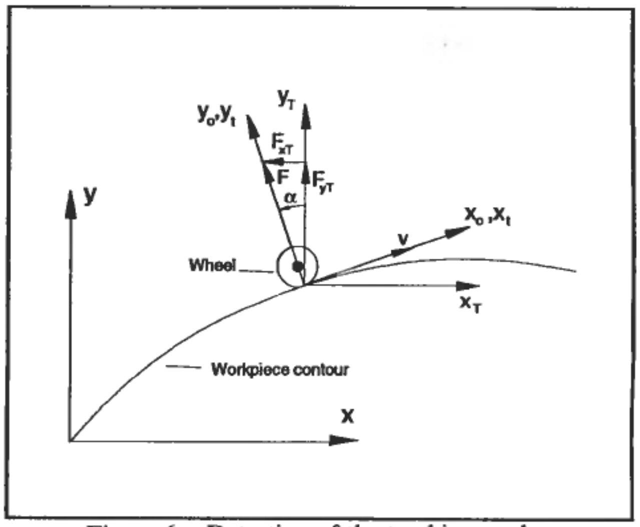

Figure 6. Detection of the tracking angle.

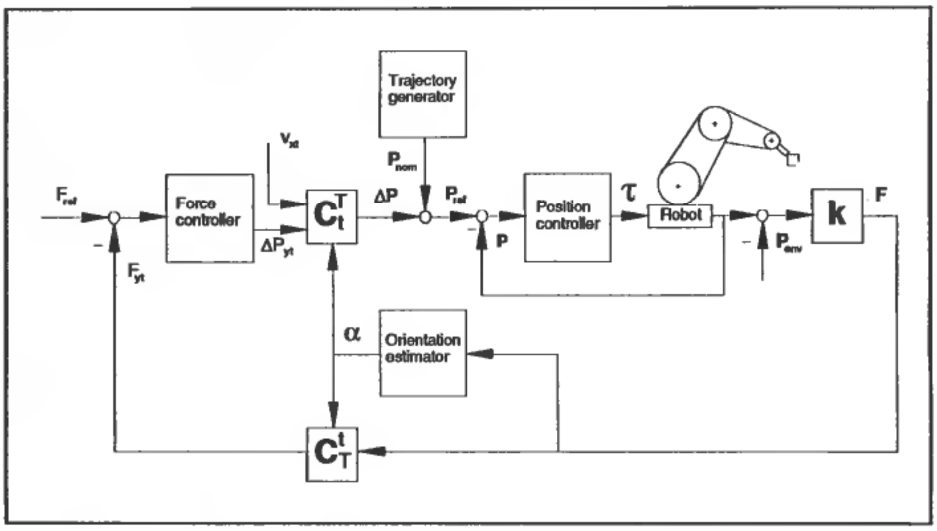

Figure 7. The automatic contour tracking system.

3.3. *Automatic path generation*

After tracking the workpiece contour, the data from logging the robot position is stored in the control system. This amount of data is disproportionately large. In addition, it is influenced by noise. Thus, the data has to be processed. The processing includes the following steps:

Noise reduction
Point reduction
Robot program generation

The objective is to reduce the amount of data as much as possible while staying within desired accuracy. From this follows that the point density is high if the contour is curved and low if the contour is approximately a straight line.

Finally, the points have to be transformed to the standard format used by the robot control system.

3.3.1. *Noise reduction.* The noise is removed from data by a two-step filtering method. The measurements around sharp corners are often influenced by noise due to high dynamic forces. This has influence on the contract force. By using a threshold for the maximum and minimum acceptable contact force, the measurements influenced by this type of noise are removed. This is called Force threshold filtering.

Due to small instabilities and inaccuracies in the control system, the measurements along a straight edge are located on both sides of the edge as a small cloud. A Gaussian window filter (Godtliebsen, Spjøtvoll, 1990) was applied to remove this kind of noise. This filter is designed on the assumption that the noise follows approximately a normal distribution. The coordinates to the current point is calculated as a function of the neighbouring points within a desired window. The window constant *n*, specifies the number of points used within the window. If e.g. *n* is equal ten, the ten previous and subsequent points are used when calculating the value to the current point. The new value of the current point is calculated as the sum of the previous and subsequent points weighted according to a Gaussian function.

On mathematical form, the new value is given by

$$\hat{p}_b = \sum_{i \in D_b} \hat{w}_i p_i \tag{4}$$

where $\hat{p}_b$ is the estimated value for p_b, p_i is the value of point i, $\hat{w}_i$ is a normalized weighting for p_i, D_b is all the points within the window, b is the index of the current point and i is the index of the point within the window. (The range of i is from b-n to $b+n$.)

The weighing w_i, is given by

$$w_i = \exp(-1/2\tau^2(p_i - p_b)^2 \tag{5}$$

where p_i and b_b are the values of point i and the current point, and τ is the filter constant.

The sum of the weightings has to be 1. Consequently, the weightings have to be normalized by the following equation

$$\hat{w}_i = \frac{w_i}{\sum_{i \in D_b} w_i} \tag{6}$$

The window is moved from the starting point to each of the measurement points along the contour. The windows for the first and the last points are necessarily smaller because there are no points before the first and after the least point. Consequently, these measurements will still be influenced by some noise after filtering, but this has usually no practical consequences.

3.3.2. *Point reduction.* The amount of data after scanning the contour is disproportionately large, and it has to be reduced. This done by approximating the contour by straight line segments. The approach is called the *Deviation height method* (Fig. 8). A straight line is made from a certain starting point on the contour to the current point. The deviation height is calculated between the line and each of the intermediate points. The deviation height is length of the normal vector between the point and the line. The current point is displaced along the contour until the deviation height exceeds a certain limit. The previous point is then used as starting point for the next line segment. This continues until the whole contour is approximated with straight line segments.

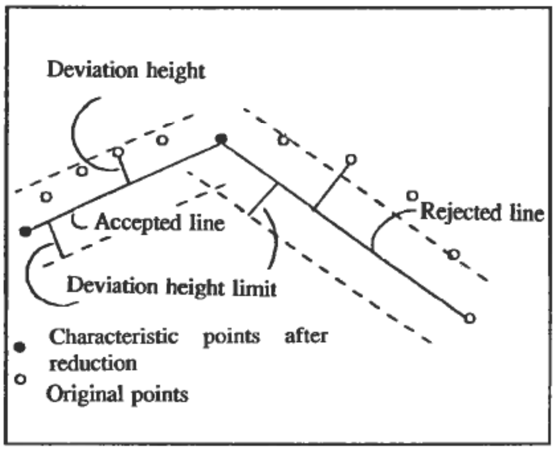

Figure 8. The deviation height method.

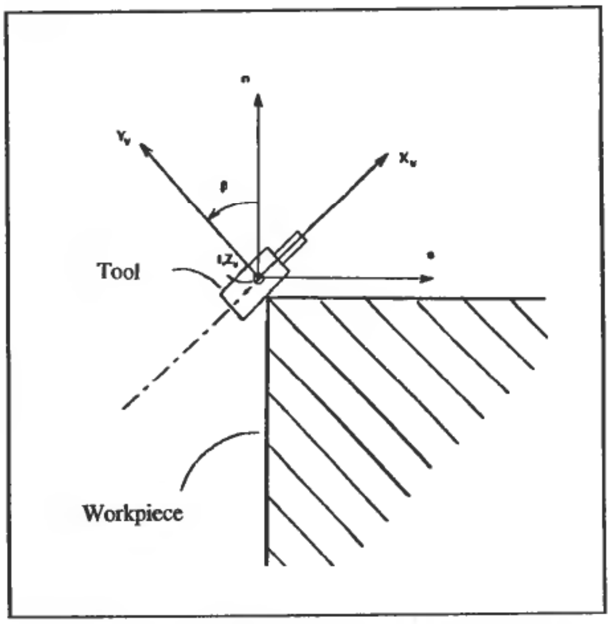

Figure 9. Specification of the tool orientation.

The deviation height limit determines the accuracy of the approximated contour. A small deviation height gives high accuracy but also a large number of straight line segments. On the other hand, a large deviation height gives a large data reduction but also a rougher approximation of the scanned contour.

3.3.3. *Robot program generation.* From the reduced data, a robot program is generated. The user must specify tool definitions, desired path velocity and orientation of the tool (Fig. 9). The orientation is specified by the angle β, between the normal vector to the contour n, and the tool axis x_v. This is equivalent to a rotation around the tangent vector t, to the contour. Consequently, the tool axis has a constant orientation with respect to the normal vector n, and it is not necessary to specify new tool orientations for each line segment.

If desired, the path velocity can automatically adapt to the path curvature (dynamic path velocity). Thereby path acceleration can be contained within desired limits. This is advantageous both for the path accuracy and the process duration.

4. Laboratory experiments

4.1. *Laboratory equipment.*

The experiments were performed using a MultiCraft 560 robot (Fig. 10). It has a very rigid structure through the use of parallel actuators. It is therefore ideal for grinding purposes. The control system is based on a VME-bus and uses the Motorola 68020/68882 processor family at 16·7 MHz for the interpolation, force control and coordinate transformations. A 125 Hz sampling frequency was obtained. The servo controller runs on a EURO-bus with a Motorola 68008 processor. The sampling frequency for the servo controller is 250 Hz. The user interface runs on a personal computer (386sx PC).

A three axis force sensor from Kistler was mounted between the tool and tool-fixture on the robot to measure contact forces. A specially constructued tool was used

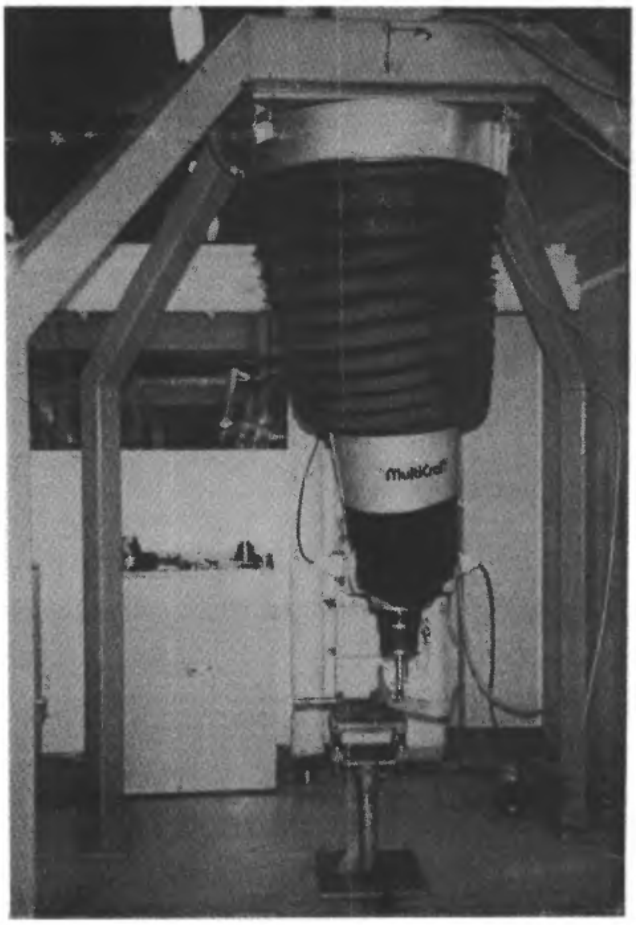

Figure 10. The Multicraft 560 robot.

during the tracking experiments. It consisted of a ball bearing mounted on a metal rod. A teach box developed for this control system, was used during the testing. Three joy-sticks were used to apply the jog-and-teach approach.

When automatic programming was used, the robot was first moved in contact with the workpiece going to be tracked. The contour tracking was activated by simply pressing a button on the teach box. The operator had to keep his finger on the button until the desired contour was tracked. When the button was released, the noise reduction, point reduction and robot program generation was automatically executed, and a robot program on the standard format for the MultiCraft 560 robot was generated.

4.2. *Experimental results.*

The performance of the force control system was tested. The force reference was momentary changed from 0 N to 30 N, and the step response was logged (Fig. 12). From the figure, the 63% rise time was found to be approximately

$$T_{0\cdot 63} = 57\,\mathrm{ms} \tag{7}$$

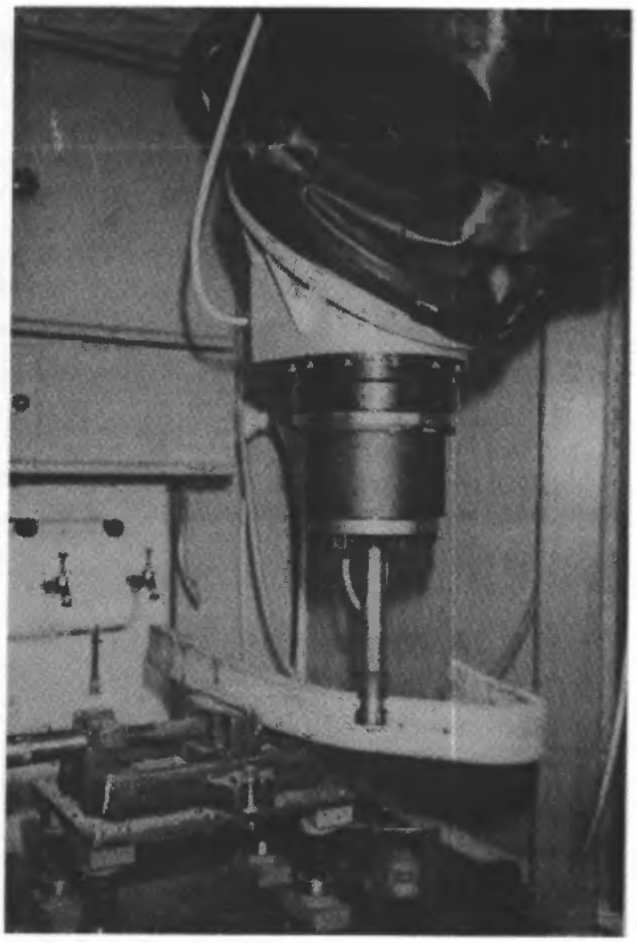

Figure 11. The piece of wood used in the tracking experiments.

Then, the bandwidth was calculated to be

$$\omega_b = \frac{1}{T_{0\cdot 63}} = \frac{1}{0\cdot 057} s^{-1} = 17\cdot 54\, s^{-1} \tag{8}$$

This is equivalent to 2·8 Hz.

Experiments were performed tracking the contours along a piece of laminated wood (Fig. 11). Along the edges there were both straight lines and acute angles. The piece was therefore ideally suited for uncovering possible errors in the system.

The robot was guided to the piece of wood using the joy sticks, and the direction of motion was indicated. From there the robot tracked the edges of the piece. A contact force normal to the edge of 20 N was used, while the velocity was 30 mm/s. The robot position was continuously stored in the control system, and is plotted in Fig. 13.

Near the sharp corners of the contour, the measurements were noisy (marked with circles on Fig. 13) due to the limited tracking capabilities of the system. This kind of noise was efficiently removed by force threshold filtering (Fig. 14) using a threshold of 13 N. It is important to notice that this filter did not smooth the contour.

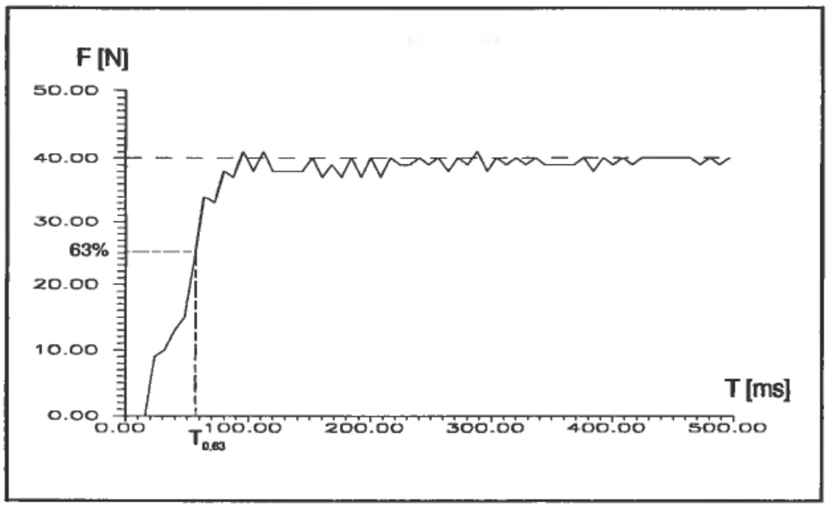

Figure 12. Step response of the force control system.

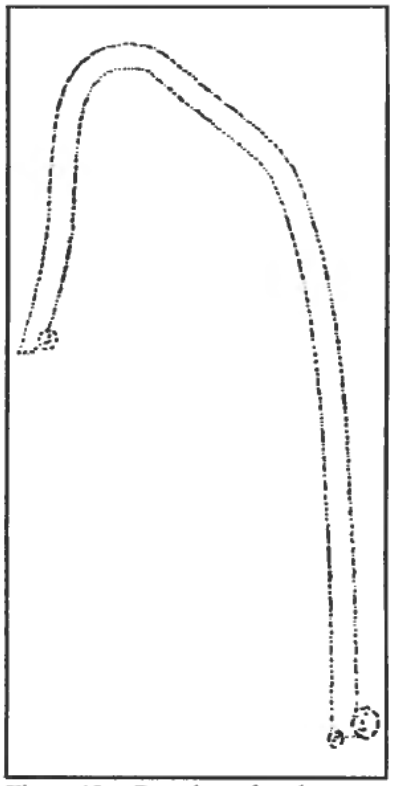

Figure 13. Raw data after the automatic contour tracking.

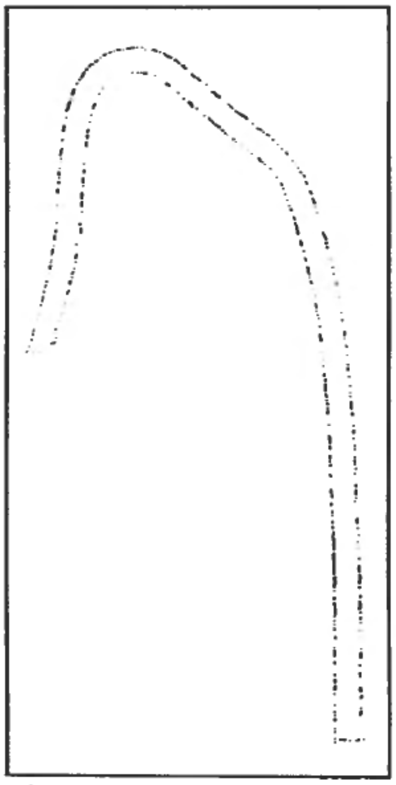

Figure 14. Removing noise from the data using the force threshold filter.

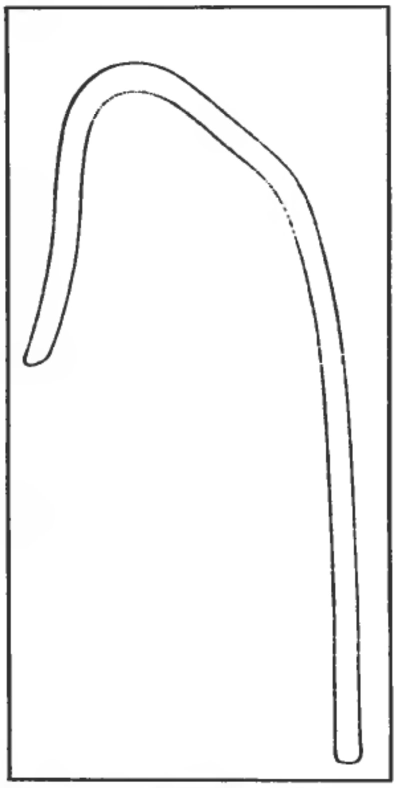
Figure 15. Removing noise from the data using the Gaussian window filter.

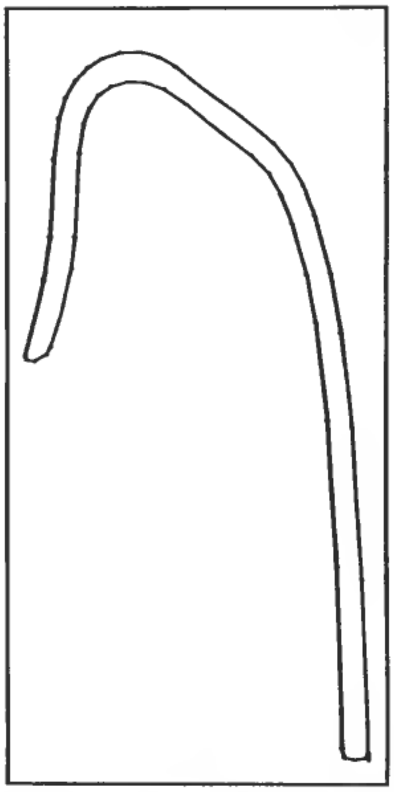
Figure 16. Point reduction using the deviation height method.

Using the Gaussian window filter on the data in Fig. 14 resulted in close approximations and a smooth surface (Fig. 15). This simplified the following point reduction to a desired degree of accuracy. Corners were rounded somewhat, but this could be countered by optimizing the filter parameters and using higher point density along the edge.

The point reduction technique was performed on the filtered measurements in Fig. 15. The result is shown in Fig. 16. The corners were identified by using the deviation-height method. A deviation height of 0·5 mm reduced the 1091 points originally recorded to 60 points. This corresponded to a reduction of more than 94%.

The reduction in programming time was also investigated using automatic programming. To track the piece of wood, the robot needed approximately 50 seconds. Similarly, a skilled operator needed about 6 minutes to program the piece of wood using the jog-and-teach approach (60 points). This corresponds to a reduction in programming time of at least 80% by applying automatic programming. In addition, automatic programming was much simpler to carry out than manual programming.

The data processing time for automatic programming (approximately 3 minutes) is not included in this comparison because it is dependent on the performance of the microprocessor system. In the future, the system will be moved to a faster microprocessor system with the Motorola 68040 processor. This will probably reduce the data processing time with a factor between 5 and 10.

5. Conclusion

A system for automatic programming of grinding and deburring robots has been presented. The main modules are a module for automatic tracking of the workpiece contour and automatic generation of the robot program. The system may be implemented on most robot controllers provided that they have an open interface for the connection of sensors. A sufficiently high sampling frequency will also be required (larger than 100 Hz).

During scanning, automatic contour tracking was used. The system was capable of following contours with varying curvature even at speeds approaching 100 mm/s. Even acute angles were traced without difficulty. But the accuracy of the measurements was reduced by high speeds because vibrations occurred due to large dynamic forces. Speeds up to 30 mm/s gave satisfactory accuracy on the measurements.

The noise reduction was done using force threshold filtering and a Gaussian window filter. This removed efficiently the noise from the measurements.

The amount of data was reduced by the deviation height method. This gave a good approximation of the tracked contour presented by a minimum number of points. A 94% reduction in the amount of data was obtained in the experiments (deviation height 0·5 mm).

The programming of grinding and deburring robots is greatly simplified through these measures. Letting the system determine the optimal grinding speed is also a pronounced advantage. Compared wo the jog-and-teach approach, the experimental results show a reduction in programming time more than 80%.

Using the automatic programming approach, the robot is easily reconfigured for working on different products. This makes it profitable to apply grinding robots in companies with small batch sizes.

Acknowledgment

This project is financed by The Swedish Foundries' Association, The Nordic Industrial Fund, MultiCraft a.s and Sperre Foundry A/S.

References

Abele, E., Boley, D. and Sturz, W. (1985). Interactive programming of industrial robots, Proceedings of the 14th International Industrial Robot Symposium.

De Scutter, J. (1986). Compliant Robot Motion, PhD. Thesis, Katholieke Univesiteit Leuven.

Godtlibsen, F. and Spjøtvoll, E. (1990). A Gaussian window filter, Division of Mathematical Sciences, The Norwegian Institute of Technology.

Hirzinger, G. (1982). Robot Teaching via Force-Torque Sensors, 6th European Meeting on Cybernetics and Systems Research, EMCSR 82, Vienna, 1982.

Schmid, D. (1990). Sensor simulate tools, *The Industrial Robot* June 1990, 97–99.