26

Robot Programming in Machining Operations

Bjørn Solvang[1], Gabor Sziebig[1] and Peter Korondi[2]
[1]Narvik University College, [2]Budapest University of Technology and Economics
[1]Norway, [2]Hungary

1. Abstract

Driven by the need for higher -flexibility and -speed during initial programming and path adjustments, new robot programming methodologies quickly arises. The traditional "Teach" and "Offline" programming methodologies have distinct weaknesses in machining applications. "Teach" programming is time consuming when used on freeform surfaces or in areas with limited visibility /accessibility. Also, "Teach" programming only relates to the real work-piece and there is no connection to the ideal CAD model of the work–piece. Vice versa during offline programming there is no knowledge about the real work-piece, only the ideal CAD model is used. To be able to relate to both the real- and ideal- model of the work-piece is especially important in machining operations where the difference between the models often represents the necessary material removal (Solvang et al., 2008 a).
In this chapter an introduction to a programming methodology especially targeted for machining applications is given. This methodology use a single camera combined with image processing algorithms like edge- and colour- detection, combines information of the real and ideal work-piece and represents a human friendly and effective approach for robot programming in machining operations.

2. Introduction

Industrial robots are used as transporting devices (material handling of work pieces between machines) or in some kind of additive- (e.g. assembly, welding, gluing, painting etc.) or subtractive- manufacturing process (e.g. milling, cutting, grinding, de-burring, polishing etc.). Also the industrial robot controller has good capability of I/O communication and often acts as cell controller in a typical set-up of a flexible manufacturing cell or system (Solvang et al., 2008 b)
Until now, driven by the need from the car manufactures, material handling and welding has been the most focused area of industrial robot development. Zhang et al. (2005) reports that material handling and the additive processes of welding counted for 80% of the industrial robot based applications in 2003, but foresights an extension into subtractive manufacturing applications. Brodgård (2007) also predict that a lightweight type robot, capable of both subtractive and additive manufacturing, will have an impact on the car industry and the small and medium sized enterprises (SMEs).

Large European research programs, like the framework programme 6 (FP 6) has put a major focus on the integration of the robot into the SME sector (SMErobot, 2005) and also in the newer research program FP 7: Cognitive Systems, Interaction, Robotics (CORDIS Information and Communication Technologies, 2008) there is focus onto the integration of the intelligent robot system with the human being.

In general there exist several strong incentives to open new areas for the usage of the industrial robot. However, many challenges in this development have been identified such as: lack of stiffness in the robot arm (Zhang et al., 2005) and difficulties in integration of various sensors due to vendor specific and their "normally closed" robot controllers.

In the case of subtractive manufacturing, contact forces must be taken into account. Forces must be measured and provided as feedback to the robot control system. Such force measurement usually involves sensors attached to the robot end-effector. At least older robot controllers do not support such possibility of sensor fusion within the control architecture. Several academic researches have been focusing in this field where some choose to entirely replace the original controller like Garcia et al. (2004) while others again include their own subsystem in-between the control loop, like Blomdell et al. (2005) and Thomessen & Lien (2000).

To address problems related to sensor integration in robotic applications several general architectures, named middleware's, are being developed for easy system integration. In Table 1. some developers of robot middlewares are identified. There are several differences between these units, among others; how many robots are supported; plug and play discovery; etc. Each of theses middleware solutions are composed from modularized components and have a hierarchical setup (Solvang et al., 2008 b). Brodgård (2007) states that one prerequisite for successful introduction of the industrial robot to the SMEs, and their rapidly changing demands, are modularity in the assembly of the robot arm (task dependant assembly of the mechanical arm) but also modularity of the software elements in the controller. As of today the robot manufactures seem to show more interest in sensor integration, perhaps driven by the competition from the middleware community as well as the possible new market opportunities represented by the huge SME sector.

Another challenge related to the introduction of the industrial robot into lighter subtractive machining operations is an effective human-robot communication methodology. Traditionally, man-machine interaction was based on online programming techniques with the classical teach pendant. Later we have seen a development into virtual reality based offline programming methodologies. Actually today, a wide range of offline software tools is used to imitate, simulate and control real manufacturing systems. However, also these new methodologies lack capability when it comes to effective human-machine communication. Thomessen et al. (2004) reports that the programming time of grinding robot is 400 times the program execution time. Kalpakjian & Scmid (2006) states that a manual de-burring operation can add up to 10% on the manufacturing cost. In general manual grinding, de-burring and polishing are heavy and in many cases unhealthy work-tasks where the workers need to wear protective equipment such as goggles, gloves and earmuffs (Thomessen et al., 1999).

Name	*Middleware technology*	*Relevant contributors*
ASEBA (Magnenat et al., 2007)	CAN	EPFL
CLARAty (Nayar & Nesnas, 2007)	Multi-level mobility abstraction	NASA
Microsoft RS (Jackson, 2007)	Web-Services	Microsoft
Miro (Weitzenfeld et al., 2003)	CORBA	University of California
Orca (Ozaki & Hashimoto, 2004)	ICE	KTH Stockholm
OrIN (Mizukawa et al., 2002)	DCON, SOAP, XML	JARA
Open-R (Lopes & Lima, 2008)	Aperios OS	Sony
Orocos (Bruyninkx et al., 2003)	RealTime Toolkit	Katholieke Universiteit Leuven
Player (Kranz et al., 2006)	Client / Server architecture	Multiple
RT-Middleware (Ando et al., 2005)	CORBA	AIST
YARP (Metta et al., 2006)	Client / Server architecture	MIT
UPnP (Veiga et al., 2007)	HTTP, SOAP, XML	University of Coimbra
Urbi (Baillie, 2005)	Client / Server architecture	Gostai

Table 1. Middleware architectures, based on Solvang et al. (2008 b)

Thus, there is a need to develop new methodologies for programming of industrial robots, especially associated to lighter machining operations like grinding/de-burring and polishing

In this chapter, focus will be kept on the human friendly and effective communication between the human operator and the robot system.

The organization of this chapter is as follows: Section 3 gives an overview of the presented programming methodology while section 4 presents details of some system components. Section 5 concludes and gives recommendations for further work.

3. Programming of the industrial robot in lighter machining operations: A conceptual methodology

Based on the challenges introduced above, the authors have developed a conceptual programming methodology, especially targeted for lighter machining operations.

The concept of the methodology is captured in Fig.1. The key-issue of the proposed methodology is to capture the knowledge of a skilled operator and make a semi- automatic knowledge transfer to a robot system. The expertise of the worker is still needed and

appreciated. The man- machine interaction is carried out in a human friendly way with a minimum time spent on robot programming.

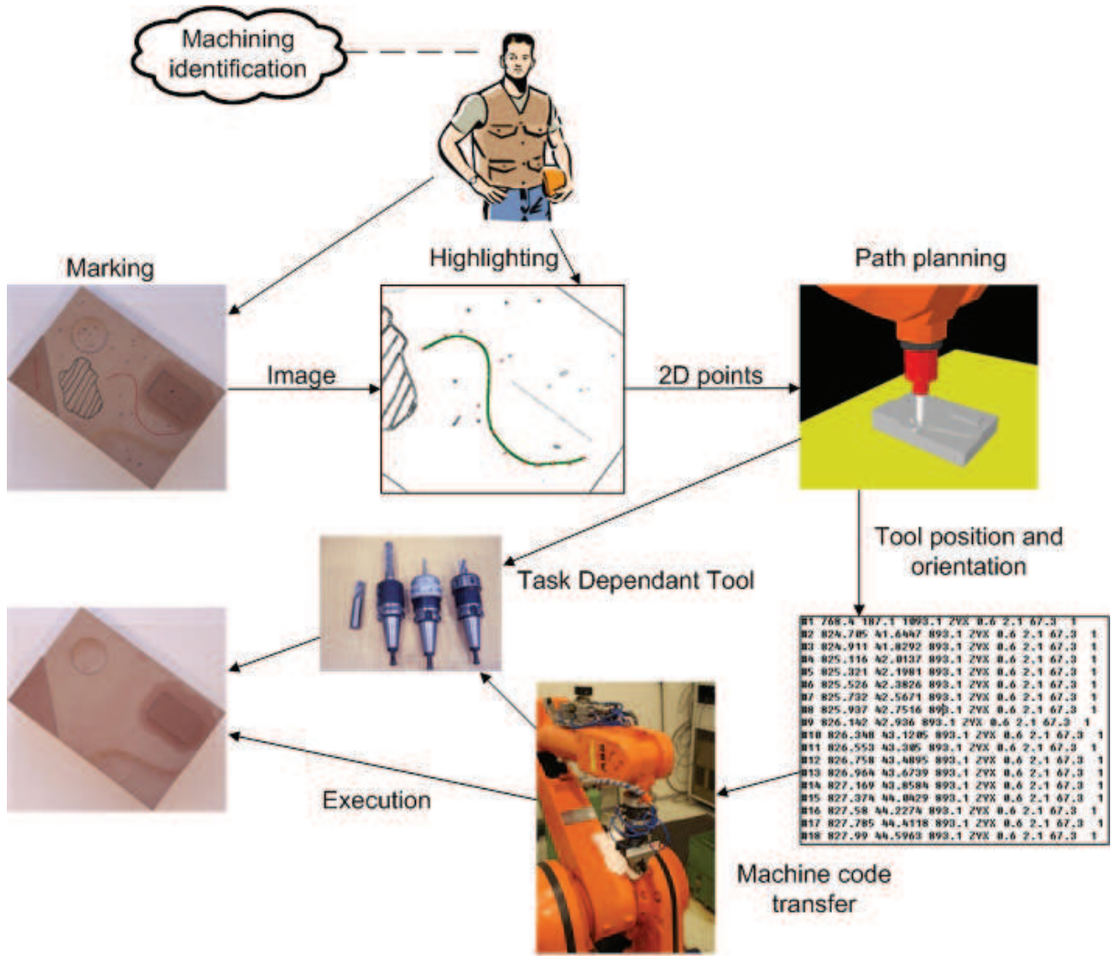


Figure 1. Conceptual overview

The concept is the following: a manufactured work-piece is inspected by an operator, who decides if there is any more machining necessary. The operator indicates the machining tasks by drawing (Marking) different shapes on the surface of the work-piece. Different colour mean different machining operation (e.g. green = polishing). The size of the machining is depending on the marking technique (e.g. curve = one path (if tool size equals curve's width)). After the marking a photo is taken. The marked surface is converted first to 2D points (with the help of an operator and image processing techniques) and second to robot position and orientation data. The operator is mainly involved in the error highlighting process. Finally the "error-free", cleaned work-piece is manufactured by the robot.

The methodology steps are formalized in the following sequence:

1. Work-piece inspection
2. Machining selection
3. Image processing (see Section 4.1)
4. Image to cutting points conversion (see Section 4.2)

5. Tool orientation establishment (see Section 4.3)
6. Locating the work-piece (see Section 4.4)
7. Simulation
8. Execution of path

The steps in details are the following:

First of all the work-piece error identification should be carried out. This is done by a human operator, who can spot the error very easy and can identify that there are irregularities at a certain area, but cannot at the same time state the magnitude and exact location of the error.

Next the operator should try to determine how such an error will impact on the functionality or the aesthetics of the work-piece and state when ether this is a critical error in such aspects. In fact, these operator messages can be written directly onto the work-piece by using standard marker pens. At a later stage, in the image processing module, colour differentiation is used to pick out messages from the operator.

Further the operator should determine if the error is of type point, line or region. Also in this case different colour pens can be used to draw lines, to mark regions, to scatter point clouds directly onto the surface.

After the error identification and classification the operator should select an appropriate machine tool for the material removal. This of course requires an experienced operator which is trained to evaluate error sources, its significance and the available machine tools to correct the error. Cutting depths are, at this stage, unknown to the operator and represents a challenge. Increasing depth means higher cutting forces and is a challenge for the less stiff industrial robot, compared with the conventional NC-machine. Zhang et al. (2005) states that the NC machines typically are 50 times stiffer than the industrial serial robots. Unfortunately, what is gained in flexibility and large working area is paid off by a decreased stiffness of the robot. However, in case of doubt, high accuracy verification of cutting depths can be undertaken by measuring the work-piece in a coordinate measuring machine (CMM). Also, at a later stage in the programming methodology there is a possibility to check for cutting depths by setting a few teach points on top of the machining allowance along the machining path. Anyway in lighter subtractive machine processes forces are small, and could as well be monitored by a force sensor attached to the robot end-effector.

Assuming selection of a robot as machine tool, the next step in the procedure is image retrieval and processing. A single camera system is used to capture the area of error and the possible written messages from the operator. A key issue is to capture some known geometrical object on the picture which can be used for sizing the picture and establishes a positional relationship between the error and the known geometry. The final goal of the image processing is to give positional coordinates of the error source related and stored in a work-piece reference frame. Details of the image processing module will be presented further in section 4.1

The image processing module present the machining path as 2D coordinates with reference to a work-piece coordinate system. In the next module this path must transferred into 3D coordinates by determination of the depth coordinate. In an offline programming environment the 2D path is transferred to the surface of a CAD model of the work-piece by an automatic “hit and withdrawal” procedure. After recovering the depth coordinate the 3D cutting points (CPs) are known. This procedure is further described in section 4.2

The best cutting conditions, in a given cutting point, is met when a selected tool is aligned with the surface at certain angles. To enable such tool alignment, information of the surface

inclination must be determined. Such procedure starts with finding the surface coordinate system in each cutting point by seeking the surface inclination in two perpendicular directions. Tool orientation is further described in section 4.3.
Out in the real manufacturing area, the relationship between the work-piece coordinate system and the industrial robot must be established. Here some already existing (vendor specific) methodology can be utilised. However, in section 4.4 a general approach can be found.
Before execution, path simulations could be undertaken in an offline programming environment in order to seek for singularities, out of reach problems or collisions.
The generated path previously stored in the work-piece coordinate system can be transferred to robot base coordinates and executed. As mentioned above, cutting depth calculations may be undertaken in order to verify that the operation is within certain limits of the machinery.

4. Some system components

In this paragraph several of the important modules of the programming methodology is presented in an overview manner, seeking to give valuable ideas and references for the reader's more than detailed technical information of each of the building blocks.

4.1 Vision and image processing module

In order to help the operator during path planning, different type of image processing techniques are used. After a short introduction of these techniques, the module structure will be shown.
Usually an image is represented as a matrix of pixels in the computer's memory.

$$I = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdot & \cdot & \cdot & \cdot & C_{1,n} \\ C_{2,1} & \cdot & & & & & \cdot \\ \cdot & & \cdot & & & & \cdot \\ \cdot & & & & & & \\ & & & & & & \\ \cdot & & & & & \cdot & \cdot \\ \cdot & & & & & \cdot & \cdot \\ C_{m,1} & \cdot & \cdot & & \cdot & \cdot & C_{m,n} \end{bmatrix} \tag{1}$$

where $C_{m,n} \in ((0\ldots255),(0\ldots255),(0\ldots255))$ represents the pixel's colour. The three values give the decomposition of the colour in the three primary colours: red, green and blue. Almost every colour that is visible to human can be represented like this. For example white is coded as (255, 255, 255) and black as (0, 0, 0). This representation allows an image to contain 16.8 million of colours. With this decomposition a pixel can be stored in three bytes. This is also known as RGB encoding, which is common in image processing.
As the mathematical representation of an image is a matrix, matrix operations and functions are defined on an image.
Image processing can be viewed as a function, where the input image is I_1, and I_2 is the resulting image after processing.

$$I_2 = F(I_1) \tag{2}$$

F is a filter or transformation. Applying a filter changes the information content of the image.

Many types of filters exist. Some of them are linear and others are non-linear. Range is from basic filters (Colour extraction, Greyscale converter, Resize, Brightness, Rotation, Blending functions) to Matrix convolution filters (Edge detectors, Sharpen filters).

The basis of the convolution filters comes from signal processing (Smith, 2002). When you have a filter you can compute its response ($y(t)$) to an entering signal ($x(t)$), by convolving $x(t)$ and the response of the filter to a delta impulse ($h(t)$).

Continuous time (Smith, 2002):

$$y(t) = h(t) \times x(t) = \int_{-\infty}^{\infty} h(\alpha) \cdot x(t-\alpha) d\alpha = \int_{-\infty}^{\infty} h(t-\alpha) \cdot x(\alpha) d\alpha \tag{3}$$

Discrete time (Smith, 2002):

$$y[k] = h[k] \times x[k] = \sum_{i=-\infty}^{\infty} h[i] \cdot x[k-i] = \sum_{i=-\infty}^{\infty} h[k-i] \cdot x[i] \tag{4}$$

where $\times$ sign is the convolution integral. The same way that we can do for one-dimensional convolution, it can be easily adapted to image convolutions. To get the result image the original image has to be convolved with the image representing the impulse response of the image filter.

Two-dimensional formula (Smith, 2002):

$$y[r,c] = \frac{1}{\sum_{i,j} h[i,j]} \cdot \sum_{j=0}^{M-1} \sum_{i=0}^{M-1} h[j,i] \cdot x[r-j, c-i] \tag{5}$$

where *y* is the output image, *x* is the input image, *h* is the filter and width and height is *M*. For demonstration of the computation of (5) see Fig. 2., which shows an example of computing the colour value (0..255) of the output image's one pixel ($y[i, j]$).

So in general, the convolution filtering loops through all the pixel values of the input image and computes the new pixel value (output image) based on the matrix filter and the neighbouring pixels.

Let observe a sample work-piece in Fig. 3. The image processing will be executed on this picture, which shows a work-piece and on the surface of it, some operator instructions. The different colours and shapes (point, line, curve, and region) describe the machining type (e.g. green = polishing) and the shape defines the machining path.

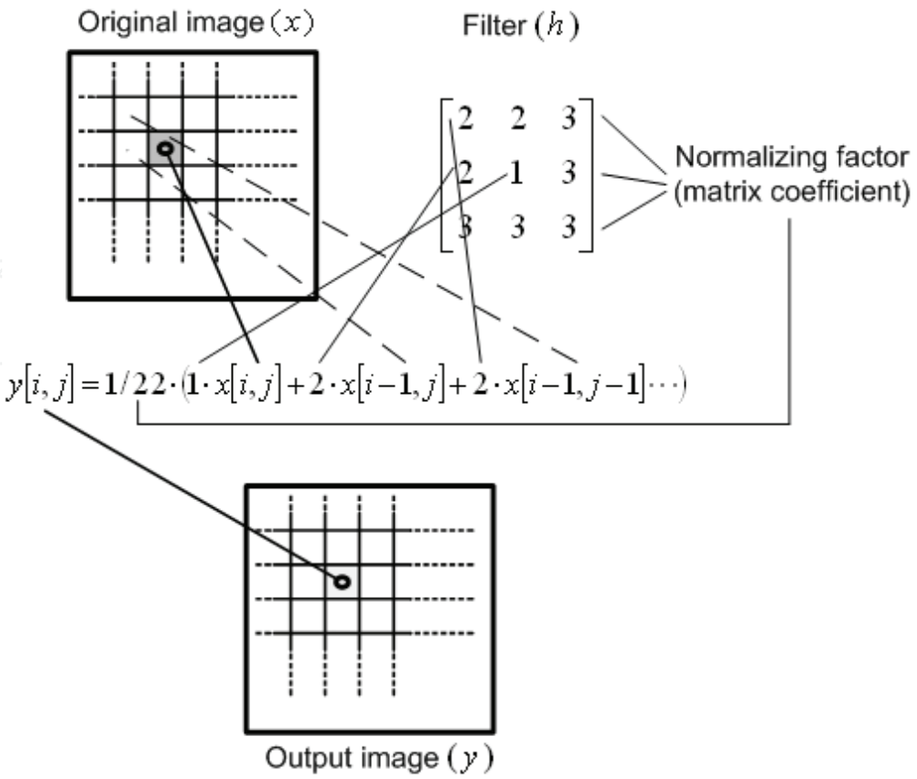


Figure 2. Demonstration of matrix convolution

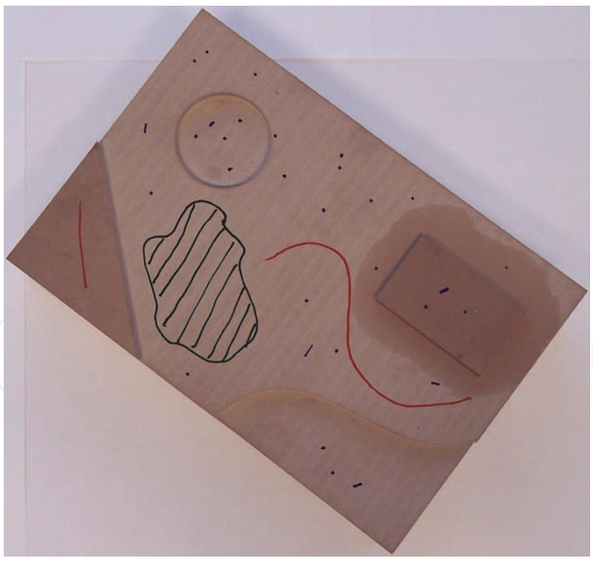
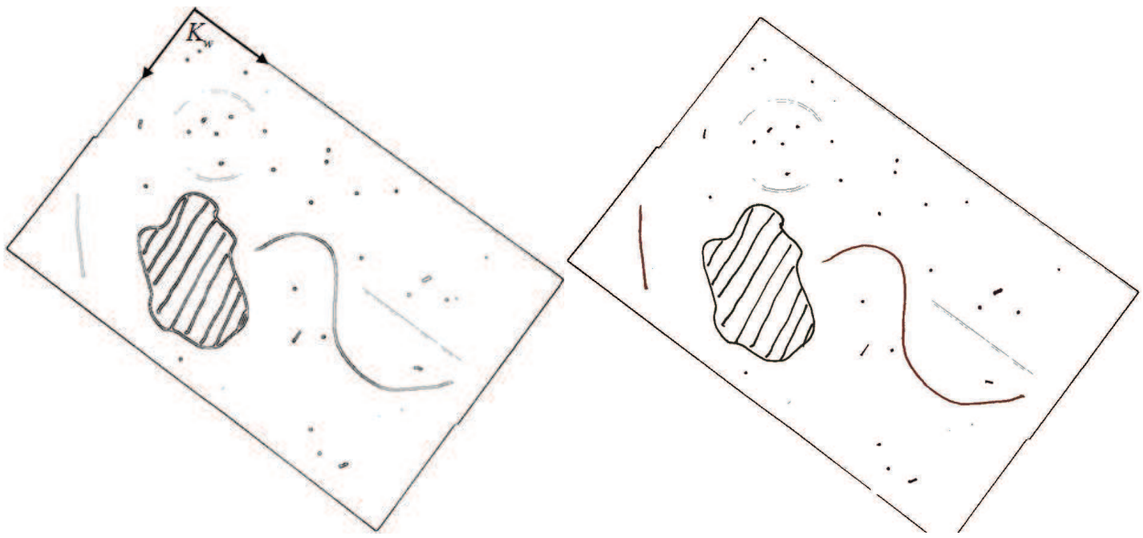


Figure 3. Work-piece example

As mentioned before the best result filters are the matrix convolution filters. Edge detection is used to make transitions more visible, which results high accuracy in work-piece coordinate system establishment (as seen in Fig. 4. (a)) and colour filtering is used to highlight the regions of the error (Fig. 4. (b)).



(a) (b)

Figure 4. (a) Work-piece coordinate system establishment. (b) Colour filtering

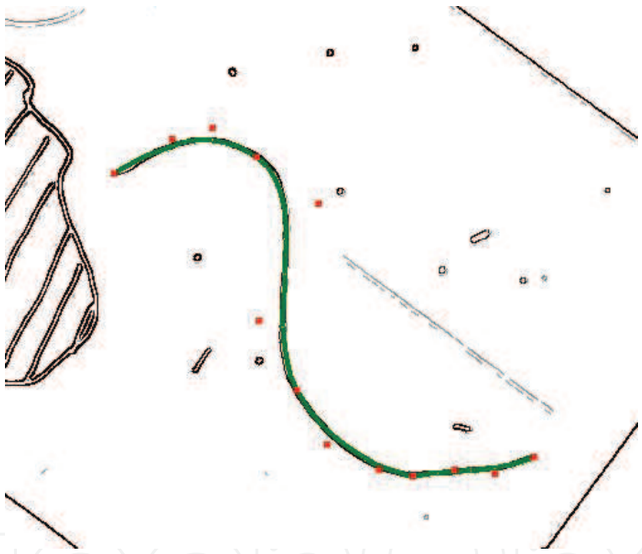


Figure 5. Curve highlighting example

The used Edge detector is the Canny edge detector. The canny edge detection is known as the optimal edge detector (Canny, 1986). With low error rate and low multiple responses (An edge is detected only once). Canny edge detection is built up from several steps for the best results. The steps contain smoothing filter, searching for edge strength (gradient of image), finding edge direction and eliminating streaks. The detailed step descriptions can be found in (Canny, 1986). Here only the filter matrix and gradient formula is presented, as proposed in (Canny, 1986):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad |G| = |G_x| + |G_y| \tag{6}$$

This performs a 2D spatial gradient measurement on the image. Two filter matrices are used for the function, one estimating the gradient in x direction and the other estimating it in the y direction (6).
The operator executes the following tasks to select machining path:

- Scale picture according to a known geometric object
- Establish a work-piece coordinate system K_w
- Select machining type, based on the colour
- Highlight the 2D path for machining
- Save the 2D path for further processing with reference to K_w

Path highlighting consists of very basic steps. The operator selects what type of shape she/he wants to create and points on the picture to the desired place. The shape will be visible just right after the creation. The points and curves can be modified by "dragging" the points (marked as red squares) on the picture. A case of curve can be observed in Fig 5.
From the highlighting the machining path is generated autonomously. In case of a line or a curve the path is constructed from points, which meets the pre-defined accuracy. The region is split up into lines in the same way as the computer aided manufacturing software's (CAM) do.
The result of the module is a text file with the 2D coordinates of the machining and the machining types. This file is processed further in the next sections.

4.2 From image to 3D cutting points

As the result of the previous section is only 2D (x and y) coordinate, the depth coordinate (z) must also be defined. This is achieved by a standard commercial available simulation program, where the industrial robot maps the surface of the work-piece. The mapping process (as indicated in Fig. 6.) is a "hit and withdrawal" procedure: the robot moves along the existing 2D path and in every point of the path the robotic tool tries to collide with the work-piece surface. If there is a collision the z coordinate is stored and a cutting position p_w^n (index w= work -piece reference coordinate system and index n= cutting point number) is established (Sziebig, 2007).

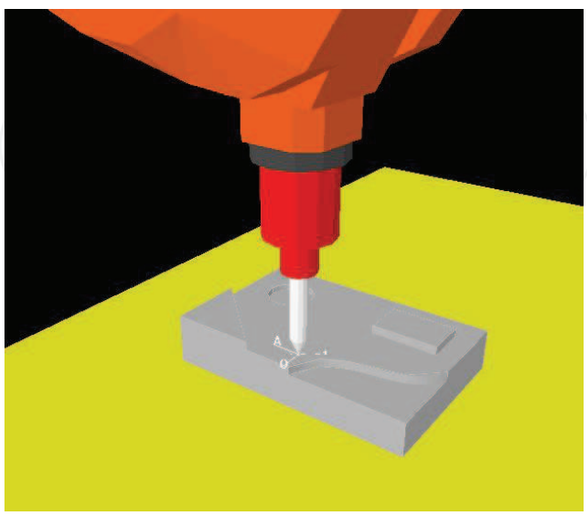


Figure 6. Demonstration of the "hit and withdrawal" procedure

4.3 Surface inclination and alignment of the cutting tool (3D to 6D)

From the previous paragraph, a set of cutting positions p_w^n were identified. However, a 3D positional description of the path is not enough.

To achieve the best and most effective cutting conditions, with a certain shaped and sized cutting tool, the tool must be aligned with the surface of the work piece at certain angles.

To enable such a tool alignment a surface coordinate system must be determined in each cutting point. The authors have previously developed an automatic procedure for finding the surface coordinate system (Solvang et al., 2008 a). According to this procedure the feed direction is determined as the normalized line X_n between two consecutive cutting points p_w^n and p_w^{n+1}.

$$X_n = \frac{p_w^{n+1} - p_w^n}{\left| p_w^{n+1} - p_w^n \right|} \tag{7}$$

Surface inclination Y_n in a direction perpendicular to the feed direction is also determined by an automatic collision detection procedure using a robot simulation tool to determine two points p_w^{n1} and p_w^{n2} perpendicular to the X_n line. This procedure consists of 6 steps, as shown in Fig.7.:

(0. Robot tool-axis Z_v ($|Z_v| = 1$) already aligned with current work- piece reference axis, Z_w ($|Z_w| = 1$)

1. Rotate robot tool axis Z_v around $Z_v \times X_n$ so that $Z_v \perp X_n$
2. Step along the robot tool- axis Z_v, away from the cutting point (p_w^n)
3. Step aside in a direction $Z_v \times X_n$
4. Move to collide with the surface and store position as p_w^{n1};

Relocate above the cutting point (p_w^n); according to step 2.

5. Step aside in a direction $-(Z_v \times X_n)$
6. Move to collide with the surface and store position as p_w^{n2}.

$$Y_n = \frac{p_w^{n2} - p_w^{n1}}{\left| p_w^{n2} - p_w^{n1} \right|} \tag{8}$$

The surface normal Z_n in the cutting point is found as

$$Z_n = X_n \times Y_n \tag{9}$$

In each cutting point p_w^n the directional cosines X_n, Y_n, Z_n forms a surface coordinate system K_n. To collect all parameters a (4×4) transformation matrix is created. The matrix (10) represents the transformation between the cutting point coordinate system K_n and the work piece current reference coordinate system K_w.

$$T_w^n = \begin{bmatrix} X_n & 0 \\ Y_n & 0 \\ Z_n & 0 \\ p_w^n & 1 \end{bmatrix} \tag{10}$$

Tool alignment angles are often referred to as the "lead" and "tilt" angles. The lead angle (β) is the angle between the surface normal Z_n and the tool axis Z_v in the feeding direction while the tilt angle (α) is the angle between the surface normal Z_n and the tool axis Z_v in a direction perpendicular to the feeding direction (Köwerich, 2002). Also a third tool orientation angle (γ), around the tool axis itself, enables usage of a certain side of the cutting tool, or to collect and direct cutting sparks in a given direction.
In case of the existence of a lead, tilt or a tool orientation angle the transformation matrix in (10) is modified according to:

$$T_w^n = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot T_w^n \tag{11}$$

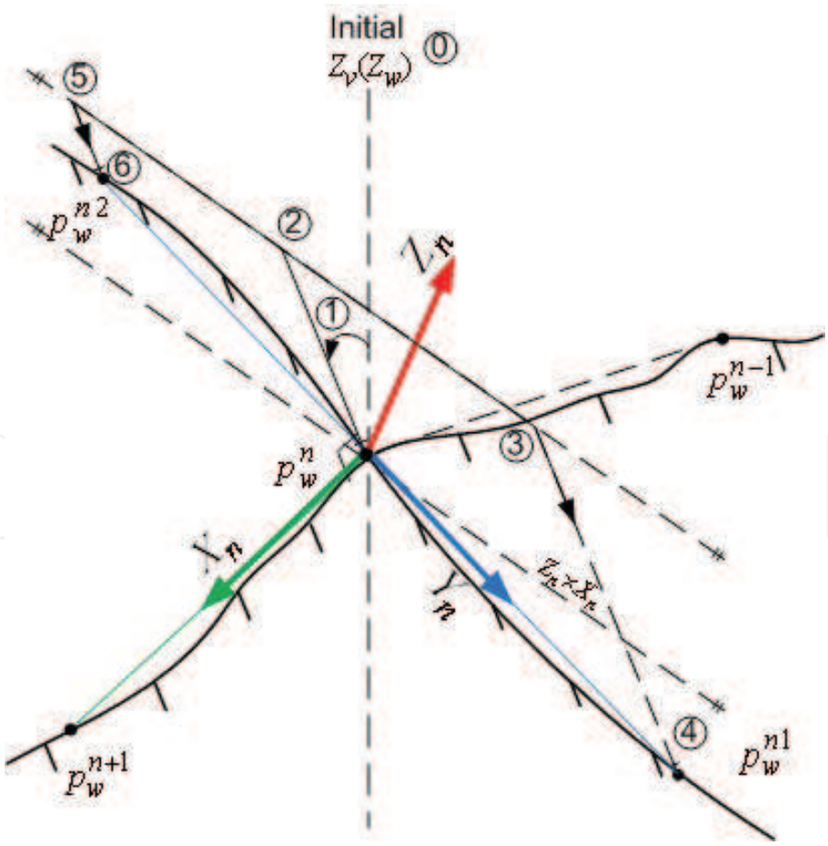


Figure 7. Steps in procedure to determine surface inclination

Fig. 8. shows the angular relationships described above.

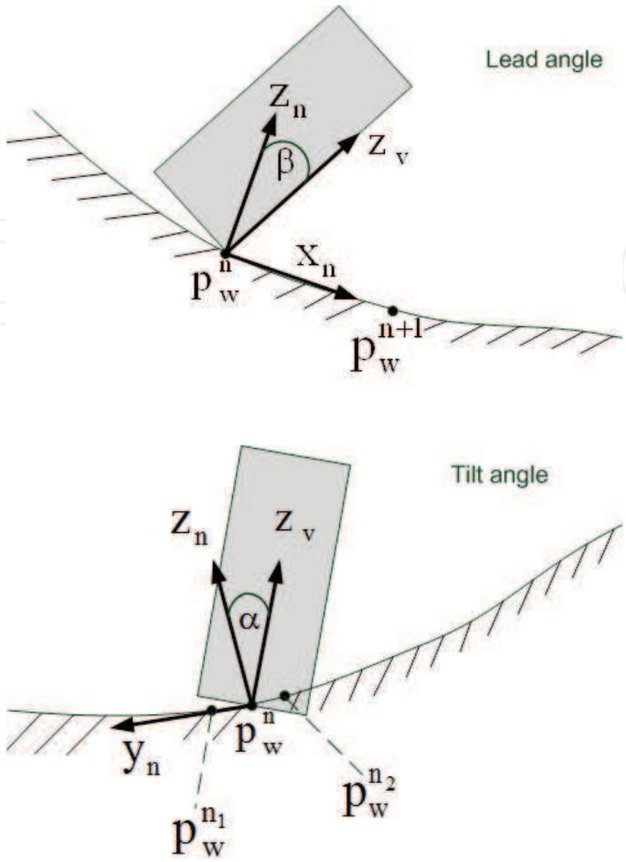


Figure 8. Angular relationships (based on Solvang et al., 2008 a)

4.4 Locating the work-piece

The robot path generated in the previous section is stored with reference to the work-piece coordinate system K_w. By such an arrangement the generated path is portable together with the work-piece. Next, before machining can start the path must be re-stored with reference to the robot base coordinate system K_0.

Typically, industrial robot systems have their own set-up procedure to determine such coordinate system relations but in many cases these are based on a minimum number of measurements and focus on a rapid simplicity more than accuracy. However, in robot machining operations the accuracy in reproducing the path heavily depends on the set-up procedure of these coordinate relationships. By adapting the typical coordinate set-up procedures found in coordinate measurement machines (CMMs) accuracy issues are well undertaken.

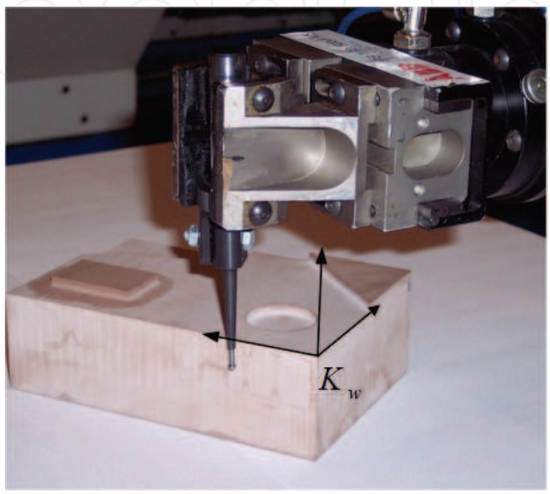
For the most significant (largest geometries) substitute (ideal) geometries are created based on robot point measurements.

The creation of substitute geometries is done by letting an operator identify what kind of ideal geometry should be created (plane, cylinder, sphere, cone etc.) Then, based on a minimum number of measurement points, such geometry is created so that the squared sum

of the distance l_i to each measurement point is minimised according to the Gaussian Least Square Methodology (Martinsen,1992).

$$\sum_{i=1}^{m} l_i^{\,2} \rightarrow \min \tag{12}$$

These substitute geometries are given a vector description with a position and a directional (if applicable) vector.



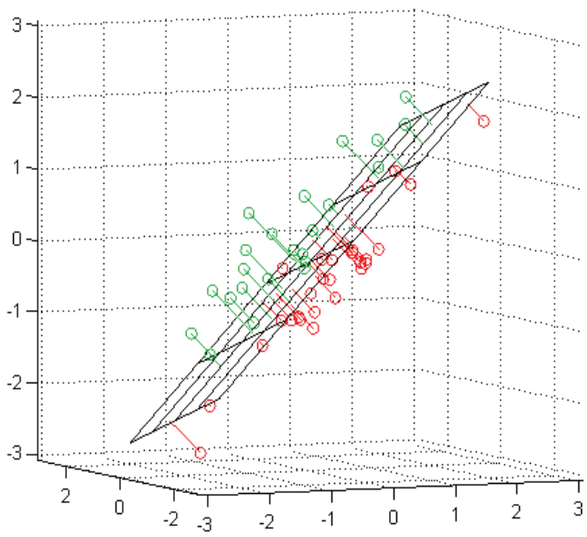


Figure 9. Probing and defining a plane

When establishing K_w, typically a directional vector from the most widespread geometry is selected as one of the directional cosines X_w, Y_w, or Z_w. The next axis could be defined by the intersection line from the crossing of two substitute geometries while the final third is found with the cross product of the two first directional cosines. The origin of K_w is

typically located in the intersection of three substitute geometries. Fig. 9. shows this methodology for a given work-piece.

The directional cosines X_w, Y_w, or Z_w and the origin p_0^w is collected in the transformation matrix T_0^w, as introduced in the previous paragraph. Finally, the T_0^w transformation matrix is multiplied with the T_w^n transformation to create the total transformation matrix T_0^n:

$$T_0^n = T_w^n \cdot T_0^w \tag{13}$$

(13) hold information on the position and orientation of the machining path in each cutting point, related to robot base coordinate system K_0. According to (10) position data is extracted from the matrix as the last row in the matrix and orientation is given by the by the directional cosines. The nine parameter directional cosines can be reduced to a minimum of three angular parameters dependant on the choice of the orientation convention. In Craig, (2005) details of most used orientation conventions and transformation are found.

Finally, the robot position and orientation in each cutting point are saved as a robot coordinate file (vendor dependant).

Before execution of the machining process, simulations can be undertaken to search for singularities, out of reach or collisions.

5. Conclusion and future work

In this chapter a conceptual methodology for robot programming in lighter subtractive machining operations have been presented, seeking to give the reader an overview of the challenges and possible solutions for effective communication between the human and the robot system. Some key-modules have been elaborated, in order to let the readers in on more technical details necessary when developing their own systems.

Some important features of the proposed methodology are:

- User communication is human friendly and rapid
- Information of the real work-piece is combined with the ideal CAD-model
- Lines, regions ,dots and messages can be drawn directly onto the work-piece
- Uses only one camera
- Semi –automatic robot path generation

The authors have carried out some initial laboratory tests with the various modules around the proposed methodology (Solvang et al., 2008 a), (Solvang et al., 2007) and (Sziebig, 2007). These results are very promising, first of all operator communication is improved and the operator is presented only those tasks he can master intuitively. When it comes to accuracy measurements results shows that the selection of camera is important.

A high resolution camera (2856*2142) produced twice as good results as a low resolution web camera (640*480). To measure the complete error chain, a set of experiments was carried out with an ABB irb 2000 robot. For these tests the low resolution web camera were used to capture a hand drawn line on a sculptured surface After the image processing the robot was instructed to follow the generated path.. The deviation between the hand drawn and the robot drawn answer was approximate 1 mm, the same accuracy as for the web camera.

The initial test shows promising results for the methodology. Tests were carried as path generations only, without any machining action undertaken. The next step would be to look at accuracy tests under stress from the machining process. For these test we need to integrate our equipment with a robot system, preferably equipped with a force sensor at the end-effector.

8. References

Ando, N.; Suehiro, T.; Kitagaki, K. & Kotoku, T. (2005). RT-middleware: distributed component middleware for RT (robot technology), *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pp. 3933- 3938, ISBN 0-7803-8912-3, Aug. 2005.

Baillie, J.C. (2005). URBI: towards a universal robotic low-level programming language, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pp. 820-825, ISBN 0-7803-8912-3, Aug. 2005.

Blomdell, A.; Bolmsjo, G.; Brogårdh, T.; Cederberg, P.; Isaksson, M.; Johansson, R.; Haage, M.; Nilsson, K.; Olsson, M.; Olsson, T.; Robertsson, A. & Wang, J. (2005). Extending an industrial robot controller: implementation and applications of a fast open sensor interface. *IEEE Robotics & Automation Magazine*, Vol. 12, No. 3, Sep. 2005, pp. 85-94, ISSN 1070-9932

Brogårdh T. (2007). Present and future robot control development—An industrial perspective. *Annual Reviews in Control*, Vol. 31, No. 1, 2007, pp. 69-79, ISSN 1367-5788

Bruyninckx, H.; Soetens, P. & Koninckx B. (2003). The real-time motion control core of the Orocos project, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA '03)*, pp. 2766-2771, ISBN: 0-7803-7736-2, Sep. 2003.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, Nov. 1986, pp. 679–698, ISSN 0162-8828

CORDIS (2008). *European Union Seventh Framework Program (FP7) to improve the competitiveness of European industry,* http://cordis.europa.eu/fp7/ict/, 2008-2012.

Craig, J.J. (2005). *Introduction to robotics, Mechanics and control*, Pearson Education, ISBN 0-13-123629-6, USA

Garcia, J.G.; Robertsson, A.; Ortega, J.G. & Johansson, R. (2004) Sensor fusion of force and acceleration for robot force control, *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, pp. 3009-3014, ISBN 0-7803-8463-6, Sep. 2004.

Jackson, J. (2007). Microsoft robotics studio: A technical introduction. *IEEE Robotics & Automation Magazine*, Vol. 14, No. 4, Dec. 2007, pp. 82-87 ISSN: 1070-9932

Köwerich, S. (2002). *5 akse maskinering: En state-of-the-art studie*, SINTEF report STF 38 A97247, ISBN 82-14-00695-3, Norway

Kranz, M.; Rusu, R.B.; Maldonado, A.; Beetz, M. & Schmidt, A. (2006). A Player/Stage System for Context-Aware Intelligent Environments, *Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys 2006) at 8th International Conference of Ubiquitous Computing (UbiComp 2006),* pp. 17-21, ISBN 3-5403-9634-9, Sep. 2006.

Lopes, N. & Lima P. (2008). OpenSDK - An Open-source Implementation of OPEN-R, *Proceedings of 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, ISBN 978-0-9817-3813-0, May. 2008.

Magnenat, S.; Longchamp, V. & Mondada, F. (2007). ASEBA, an event-based middleware for distributed robot control, *Proceedings of Workshops DVD of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, ISBN 978-1-4244-0912-9, Oct. 2007.

Martinsen, K. (1992). *Kompendium i måleteknikk og vektorielle toleranser*, Norges Tekniske Høgskole, Norway

Metta, G.; Fitzpatrick, P. & Natale, L. (2006). YARP: Yet Another Robot Platform. *International Journal on Advanced Robotics Systems*, Vol. 3, No. 1, Mar. 2006, pp. 43-48, ISSN 1729-8806

Mizukawa, M.; Matsuka, H.; Koyama, T.; Inukai, T.; Noda, A.; Tezuka, H.; Noguchi, Y. & Otera, N. (2002). ORiN: open robot interface for the network - the standard and unified network interface for industrial robot applications, *Proceedings of the 41st SICE Annual Conference (SICE 2002)*, pp. 925- 928, ISBN 0-7803-7631-55-7, Aug. 2002.

Nayar, H. D. & Nesnas, I. A. (2007). Measures and Procedures: Lessons Learned from the CLARAty Development at NASA/JPL, *Proceedings of Workshops DVD of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, ISBN 978-1-4244-0912-9, Oct. 2007.

Ozaki, F. & Hashimoto, H. (2004). Open Robot Controller Architecture (ORCA), the Robot Platform. *Toshiba Review*, Vol. 59, No. 9, 2004, pp. 20-24, ISSN:0372-0462

SMErobot (2005). *European Union Sixth Framework Program (FP6) proposal number 011838*, http://www.smerobot.org/, 2005-2009.

Smith, S. (2002). *Digital Signal Processing: A Practical Guide for Engineers and Scientists*, Elsevier, ISBN 0-7506-7444-X, USA

Solvang, B.; Korondi, P.; Sziebig, G. & Ando, N. (2007). SAPIR: Supervised and Adaptive Programming of Industrial Robots, *Proceedings of 11th International Conference on Intelligent Engineering Systems (INES 2007)*, pp. 281-286, ISBN 1-4244-1147-5, Jun. 2007.

Solvang, B.; Sziebig, G. & Korondi, P. (2008 a). Vision Based Robot Programming, *Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC 2008)*, pp. 949-954, ISBN 978-1-4244-1685-1, Apr. 2008.

Solvang, B.; Sziebig, G. & Korondi, P. (2008 b). Multilevel Control of Flexible Manufacturing Systems, *Proceedings of IEEE Conference on Human System Interactions* (HSI'08), pp. 785–790, ISBN 1-4244-1543-8, May. 2008.

Sziebig, G. (2007). Interactive vision-based robot path planning, *Master of Science thesis*, Budapest University of Technology and Economics, 82 pp., May 2007.

Thomessen, T.; Lien, T. K. & Solvang, B. (1999). Robot control system for heavy grinding applications, *Proceedings of 30th International Symposium on Robotics*, pp. 33–38, Oct. 1999.

Thomessen, T. & Lien T. K. (2000). Robot control system for safe and rapid programming of grinding applications. *Industrial Robot: An International Journal*, Vol. 27, No. 6, 2002, pp. 437-444, ISSN 0143-991X

Thomessen, T.; Sannæs, P. K. & Lien, T. K. (2004). Intuitive Robot Programming, *Proceedings of 35th International Symposium on Robotics*, ISBN 0-7695-0751-6, Mar. 2004.

Veiga, G.; Pires J.N. & Nilsson, K. (2007). On the use of service oriented software platforms for industrial robotic cells, *Proceedings of Intelligent Manufacturing Systems (IMS2007)*, May. 2007.

Weitzenfeld, A.; Gutierrez-Nolasco, S. & Venkatasubramanian, N. (2003). MIRO: An Embedded Distributed Architecture for Biologically inspired Mobile Robots, *Proceedings of 11th IEEE International Conference on Advanced Robotics (ICAR'03)*, ISBN 972-96889-9-0, Jun. 2003.

Zhang, H.; Wang, J.; Zhang, G.; Gan, Z.; Pan, Z.; Cui, H. & Zhu, Z. (2005). Machining with flexible manipulator: toward improving robotic machining performance, *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1127-1132, ISBN 0-7803-9047-4, July 2005.

Robot Manipulators

Edited by Marco Ceccarelli

ISBN 978-953-7619-06-0

Hard cover, 546 pages

Publisher InTech

Published online 01, September, 2008

Published in print edition September, 2008

In this book we have grouped contributions in 28 chapters from several authors all around the world on the several aspects and challenges of research and applications of robots with the aim to show the recent advances and problems that still need to be considered for future improvements of robot success in worldwide frames. Each chapter addresses a specific area of modeling, design, and application of robots but with an eye to give an integrated view of what make a robot a unique modern system for many different uses and future potential applications. Main attention has been focused on design issues as thought challenging for improving capabilities and further possibilities of robots for new and old applications, as seen from today technologies and research programs. Thus, great attention has been addressed to control aspects that are strongly evolving also as function of the improvements in robot modeling, sensors, servo-power systems, and informatics. But even other aspects are considered as of fundamental challenge both in design and use of robots with improved performance and capabilities, like for example kinematic design, dynamics, vision integration.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen