技術論文 한국정밀공학회지 제14권 제1호 (1997년 1월)
Journal of the Korean Society of Precision Engineering Vol. 14. No.1. JANUARY. 1997.

금형 연마작업을 위한 로봇 오프라인 프로그래밍 시스템의 개발 및 적용

강 성 철*, 김 문 상*, 이 교 일**

Development and Verification of a Robot Off-line Programming System for Die Polishing Process

Sungchul Kang*, Munsang Kim*, Kyo Il Lee**

ABSTRACT

본 논문은 금형의 마무리 연마 작업을 로봇을 이용하여 자동화하기 위한 오프라인 프로그래밍 시스템 개발을 그 내용으로 하고있다. 3차원 자유곡면 형상을 갖는 금형을 연마하기 위한 로봇 작업 경로를 효율적으로 생성하기 위해서는 기존의 교시 방법이 아닌 CAD시스템과 연계된 시뮬레이션 방식의 자동 경로 생성 방법이 요구된다. 본 연구에서 개발된 금형 연마 작업을 위한 오프라인 프로그래밍 시스템은 연마 작업 시뮬레이션을 위한 기하학적 모델링 기능, 로봇의 작업 공간을 고려한 작업장 배치 기능, 연마 로봇의 효율적인 기구학 해, 3 차원 그래픽 시뮬레이션, 3차원 물체간의 충돌 검사 기능 및 유기적인 관계형 데이타 베이스 기능 등으로 구성된다. 본 시스템의 시뮬레이션 결과를 로봇의 위치 보정 과정을 거쳐 로봇 작업 프로그램으로 변환함으로써 최종적으로 실제 연마 작업이 가능한, 정확하고 안전한 로봇 프로그램을 생성하였다.

Key Words : off-line programming(오프라인 프로그래밍), robot polishing(로봇 연마), robot inverse kinematics(로봇 역기구학), simulation(시뮬레이션), collision check(충돌 검사), robot calibration(로봇위치보정), relational data base(관계형 데이타베이스)

1. Introduction

During the processes of die manufacturing, polishing is done to remove tool marks and improve the smoothness and flatness of die surfaces. The polishing process affects the parts surfaces in two ways : fine-scale abrasive removal and softening / smearing of surface layers by frictional heating during polishing[1]. When the polishing operations are done by skilled workers, manual polishing takes a long time to meet the required accuracy,it has been a major bottleneck consuming 20 - 30 % of the total die manufacturing time. To improve the productivity, it is required to develop an

* 한국과학기술연구원(KIST) 기전연구부
** 서울대학교 공과대학 기계설계학과

automated polishing system using manipulators[2][3][4]. Since robot motion programmed for a particular die is used only once and cannot be used for any other die, labor-intensive teach-and-playback approach is not cost-effective. Therefore, in order to automate this work, it is efficient to adopt the off-line programming approach, which can overcome the problems mentioned above[5][6][7]. This off-line programming system for robot polishing is a task-oriented programming system which generates optimal collision-free polishing tool paths through the use of graphical simulation. The architecture of the polishing-robot off-line programming system (PROPS) is shown in Fig.1.

This paper is organized as follows. Chapter 2 describes the graphical simulation environment, and some important techniques like an efficient generalized inverse kinematics algorithm, a collision checking procedure and a cell calibration technique are discussed in Chapter 3. Finally, conclusions and future work are discussed in chapter 4.

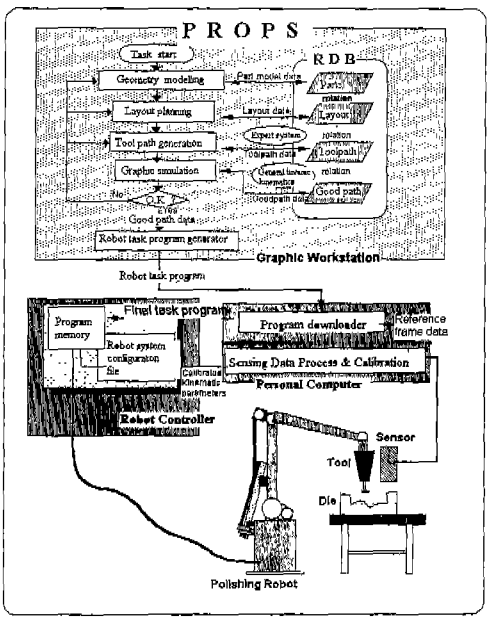


Fig. 1 Architecture of the automated polishing robot system

2. Polishing Robot Simulator

2.1 Part modeling and layout planning

To model the parts and devices in the polishing work cell such as robots, dies, tools, tool-tips, working tables, etc., two kinds of polyhedra, namely, boxes and pyramids are used as basic geometric primitives[8]. These simple shapes are suitable for reducing computation time for shading, animation and collision checks. The relatively complicated geometries such as robot manipulators can be built by proper combination of these boxes and pyramids[9]. Mathematically, both boxes and pyramids are described as an intersection of planar half spaces defined by the plane equations of their surfaces.

In this research, three-dimensional objects are assumed to be polyhedra or unions of polyhedra. Therefore, all three-dimensional objects can be defined with coefficients $\{a^i, b^i, c^i, d^i\}$ of the plane equation $a^i x + b^i y + c^i z = d^i$ corresponding to object surfaces. This mathematical definition of the objects is used to draw, shade and check collisions between parts in the workcell. The die geometries are imported from an external modeler through CATIA or IGES format, and is represented with Bi-cubic spline surfaces. When checking the collision between dies and other parts, die models are meshed into a set of triangular patches. A die data imported from CATIA is shown in Fig.2.

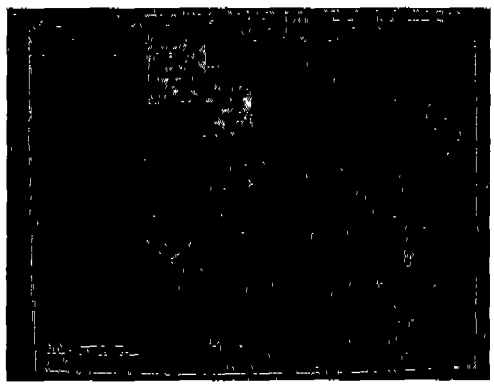


Fig.2 Imported die data (phone)

After modeling the parts — dies, robots, tools, tool-tips, working tables and tool stands — completely, these should be placed to a suitable location within the workcell. It is important to determine the position and orientation of a part considering the robot's reachable workspace and tool approaching directions. This layout planning can be done interactively using the graphical user interface that shows multiple views of the workcell (Fig.3).

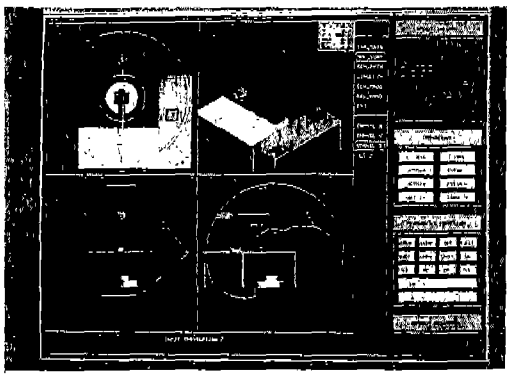


Fig.3 Layout planning for robot polishing workcell

2.2 Polishing tool path generation

The first step of generating a polishing tool path is to select proper tools and tips. Then the tooling parameters such as the feedrate of the tool, the approaching height, the path pattern can be determined. Once they are determined, we can generate a polishing tool path along the given surface automatically. Basically, the approach direction of the tool (or tool-tip) is normal to the tangential plane of a tooling point on the die surface. A polishng tool usually has a compliance mechanism that is compliant along the approach direction of the tool. The existing robot kinematic errors can be compensated in this direction. Note that our goal is not to enhance the cutting accuracy of milling process but to reduce the roughness and waviness of die surfaces.

Since a die is composed of several parametric surface patches, the tooling sequence is determined by ordering the tool paths for each surface patch, after all the tool paths for each patch have been generated. After the tool paths and their sequence are generated completely, they are saved in a data base. In the simulation of robot polishing, these tool paths and the sequence are used to generate collision-free tool paths called "good-paths." A sample tool path generated on die surfaces is shown in Fig.4.

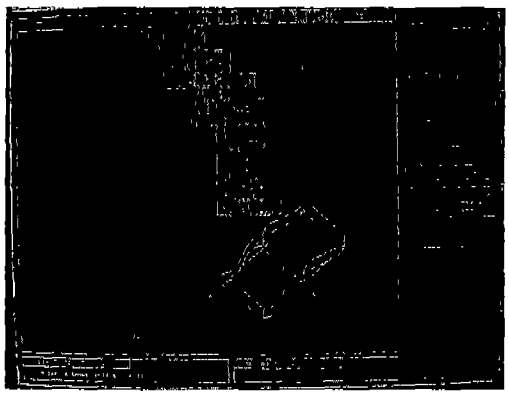


Fig.4 Tool path on die surfaces

2.3 Relational database

A Relational data base (RDB) is indispensable to store geometric data of dies and various information of robots, tools, and other peripheral devices in a polishing workcell. In addition, the point data of tool paths and final good-paths are also stored in RDB for retrieval and update. Therefore, RDB makes it possible to program the robot on the task-level. Consequently, RDB provides an environment to build an expert system for polishing operations which automatically determines the best polishing conditions without depending on human intelligence.

A RDB for polishing work has been developed in this research. The RDB is mainly composed of

three groups of data tables. The first table is concerned with dies, and the second group contains the data describing devices and parts in polishing workcell. Finally, the third group has the data produced in polishing simulation, such as layout, tool path points, tooling sequences and good-path points. The RDB is structured as shown in Fig.5.

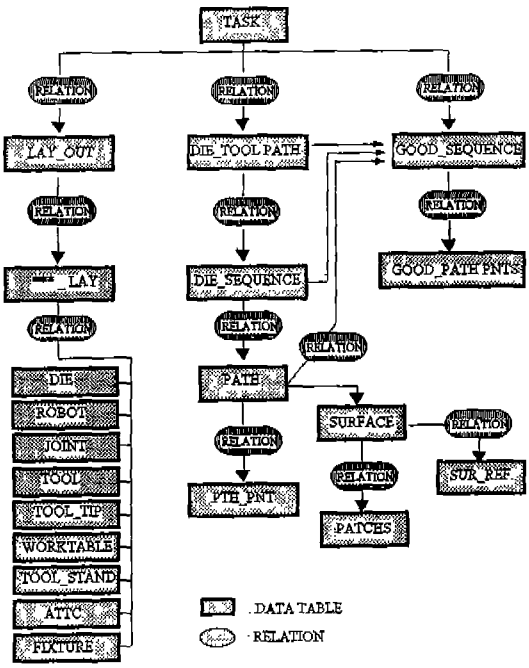


Fig.5 Structure of relational database for robot polishing

3. Algorithms for Motion Planning

3.1 Inverse kinematic analysis of the robots

To simulate the polishing process using various types of industrial robots, a generalized inverse kinematic algorithm has been proposed[13].

While analytical methods are robot specific[10][11], numerical methods can be used regardless of kinematic structure and degrees of freedom of the robot. However, these numerical methods suffer from inconsistency in convergence depending on kinematic configurations and initial conditions. To minimize these defects in numerical methods, the deviation-function minimizing method[12] was adopted in this research. This method is stable in comparison with Newton-Raphson-like method that has the possibility of resulting in the singularity problem due to a singular Jacobian matrix. Selected initial values for iteration have an effect on the number of iterations. A new algorithm called "near position method" has been proposed to determine proper initial guesses to reduce the iteration time to converge[13].

The near position method finds the position and orientation analytically which are as close to target position and orientation of TCP (Tool Center Point) as possible. Then the iteration procedure using the deviation-function minimizing method begins to find the solution closest to the initial guess. The robot considered has a shift hand, it can be reconstructed to have a kinematics which has no link shift at robot hand by setting the link length simply to zero. After a near position of virtually simple robot which is close to the target position is determined, joint variables $q_i^*, i = 1,.,6$ corresponding to near position and orientation can be calculated from position deviation vector $\Delta\{p_H\}$. These joint values $P\{q^*\}$ are not final solution {q} but only initial guesses to be used in the iterative inverse kinematic algorithm. The procedure is shown in Fig.6. This technique requires a more complicated computer program but guarantees convergence stability and reduced calculation time.

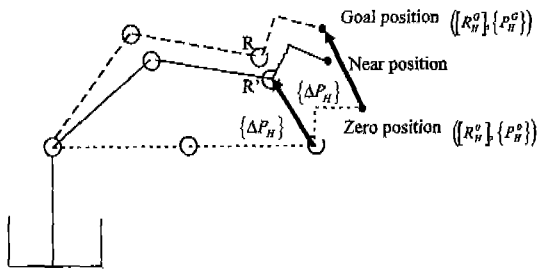
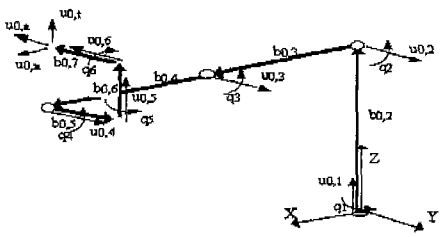


Fig. 6 Estimation of a near initial position

The deviation-function minimizing method proposed by Kazerounian[4] was used as an iterative inverse kinematic procedure to calculate the joint values $\{q\}$ of the robot. The inverse kinematic algorithm combined with the near initial position method and the deviation-function minimizing method was varified to Nachi 7601 6-DOF robot manipulator whose inverse kinematics could not be solved analytically due to the link shifts at wrist. The kinematic parameters of Nachi 7601 robot described in zero reference position method[14] is shown in Fig.7 along with the kinematic structure.



Joint	Type	$u_{0,k}$	$b_{0,k}$
1	R	(0, 0, 1)	(0, 0, 1)
2	R	(0, 0, 1)	(0, 0, 1)
3	R	(0, 0, 1)	(0, 0, 1)
4	R	(0, 0, 1)	(0, 0, 1)
5	R	(0, 0, 1)	(0, 0, 1)
6	R	(0, 0, 1)	(0, 0, 1)
$u_{0,s}$ = (0, 0, 1)		$u_{0,t}$ = (0, 0, 1)	

Fig. 7 Kinematic parameters of Nachi 7601 robot manipulator

To check the convergence, the proposed inverse kinematic algorithm has been compared with the deviation-function minimizing method which has no rule for determining initial guesses. Convergence criteria defined in Kazerounian's work were used[4]. The comparison of convergence between two inverse kinematic algorithms is shown in Fig.8. As shown in Fig.8, the proposed algorithm shows a good performance in case of small convergence criteria.

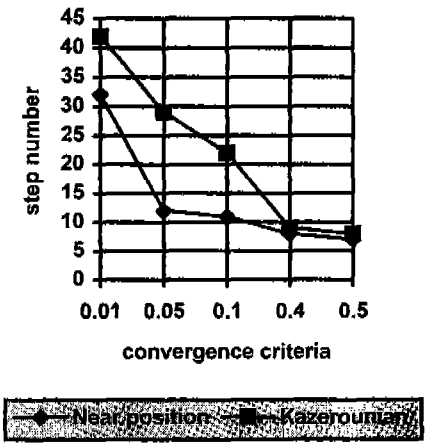


Fig. 8 The comparison between near position method and Kazerounian's method

3.2 collision check

To generate an accurate and safe robot tool path that can be used with a real robot, it is necessary to check collisions between the robot and the objects in the workcell. When a collision occurs, the path has to be modified to avoid the collision. A simple and efficient collision detection algorithm has been developed for polyhedral objects.

The part geometry is described as a polyhedron or an union of polyhedra. A union is used for concave parts. Although a distance computation routine can be used for collision checking, it is more expensive than a collision checking and thus not used in this work.

A simple collision checking algorithm is now described. Let a convex polyhedron be defined as

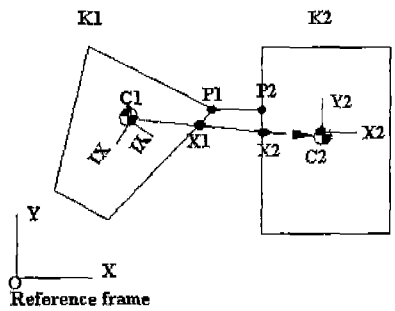
$$O = \bigcap_{j=1}^{np} \{x : \langle x, n^j \rangle \le d^j\} \quad (1)$$

where, $\langle .,. \rangle$ denotes inner product, $x = (x,y,z)$ is a vector in space, $n^j = (n_x^j, n_y^j, n_z^j)$ is a unit surface normal to the plane j pointing outward from the object O, $n_x^j x + n_y^j y + n_z^j z = d^j$ is j-th plane equation, and np is the number of planes defining a polyhedron.

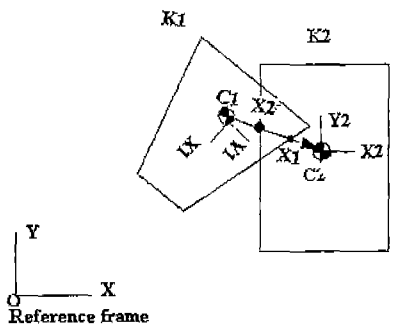
A part K composed of several convex polyhedral objects O_i's is defined as

$$K = \bigcup_{i}^{no} O_i \tag{2}$$

where no is the number of objects defining the part K. The condition for collision between two parts is shown in Fig.9.



(a) Non-collision if < X1, ni2> > di2 , i = 1,2, ..,k.



(b) Collision if < X1, ni2> <= di2 , i = 1,2, ..,k

Fig.9 The condition for collision between two objects

Let $K1$ and $K2$ be the two parts, which for $i=1,2$, Ci denotes the centroid of Ki, Xi is the pierce point which is the intersection of the line connecting $C1$ and $C2$ and the boundary of Ki, and Pi is the witness point of Ki resulting in the minimum distance between $K1$ and $K2$.

By observing that the pierce point $X1$ on object $O1$ lies within or on the object $O2$, whether the collison between $O1$ and $O2$ occurs or not can be detected. The pierce points $X1$, $X2$ are not exact minimum-distance points but approximately close to that points. Therefore, when checking collision using $X1$,$X2$, the error called near-miss would take place. The error ε_d can be defined as follows.

$$\varepsilon_d = |X2 - X1| - |P2 - P1| \tag{3}$$

Here, the positive value ε_d varies depending on the centroid position and orientation of each object. Thus, we should ε_d set as a proper constant value considering near-miss accuracy. Using ε_d, one of two objects can be to expanded volumetrically. For example, the following equation and inequality represent that the volume of $O2$ is expanded.

$$d_2' = d_2 + \varepsilon_d \tag{4}$$

$$\langle X1, n_2 \rangle \le d_2' \tag{5}$$

This algorithm minimizes the distance function

$$f(P1,P2) = |P1 - P2|^2 / 2 \tag{6}$$

To minimize the distance function, there have been mainly two researches, interior penalty function method[15] and direct minimizing method using gradient projection[9]. These approaches

require complicated iterative numerical computation proportional to the number of planes constructing the polyhedra[9]. Since our collision detection algorithm only aims at detecting collision, the distance between two objects has not been considered. The collision detection algorithm was implemented to the animated simulation as shown in Fig.10. It is noted here that our algorithm works well for convex objects whose aspect ratio is close to 1, i.e., close to sphere shape. A more complete collision-detection algorithm such as that in Bobrow[9] is being considered for future use.

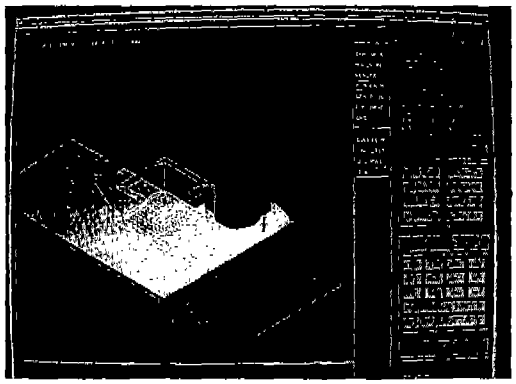


Fig.10 Collision detection during robot polishing simulation

The polishing path is generated as follows. The robot tool tip is placed at each grid point on the surface to be polished, and the points involving collision are marked with "1." The polishing path is generated in two stages. First, a path visits all the collision-free points is connected using Cho's algorithm[16]. Then the path that polishes the collision points are interactively planned using the simulation module.

Off-line simulation generates collision-free toolpath points and robot motion parameters are stored in RDB. This data is independent of the robot type, and is translated to robot's native controller language before execution.

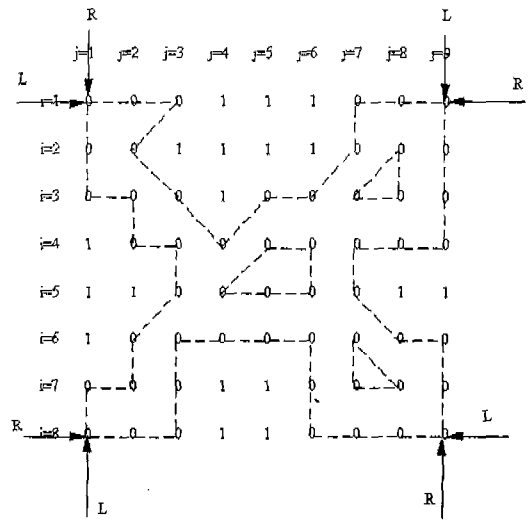


Fig11. Generation of a continuous collision-free path

3.3 Robot and workcell calibration

A good-path data generated through simulation are to be converted to various robot programs and downloaded to the real polishing robot. However, the resulting robot motion cannot track the path accurately in real world. The main reason for this inaccuracy comes from calibration errors in the robot and its workcell. There are geometrical errors, such as variations in link lengths, joint zero position error and a mismatched base location of the robot, which are dominant factors affecting robot's absolute positioning accuracy. There exist also non-geometrical errors, such as gear eccentricity, backlash, and joint compliance due to an applied payload at hand. For polishing operation, it is adequate to reduce the geometrical errors only. This is done by measuring the absolute position of TCP, and adjusting the robot geometrical parameters by using the offset between measured positions and nominal positions[17]. Many researchers have proposed various methods to calibrate robot kinematic parameters. In KIST, a calibration system that is coupled with the robot hand mechanically has been developed

and used to increase absolute positioning accuracy of robot[18] (Fig.12).

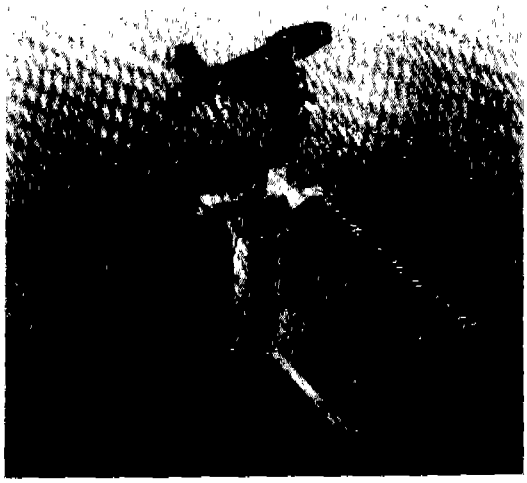


Fig. 12 Calibration mechanism attached to Nachi7601 robot

Furthermore, the positional inaccuracy also comes from errors in placing a die in the polishing workcell. This mismatch in transformation between the simulation model and the real model often takes place. This inaccuracy can be calibrated by designating robot base coordinates to reference coordinates. For this purpose, robot equipped with distance sensors attached to end-effector is used as a sensor.

4. Conclusions and Future Work

A polishing tool / robot off-line programming system has been developed to automate the polishing work for the free formed surfaces of dies. The polishing tool[19] is a mechanism that generates motion on a two-dimensional manifold and has a compliance in the direction perpendicular to the manifold shown in Fig.13.

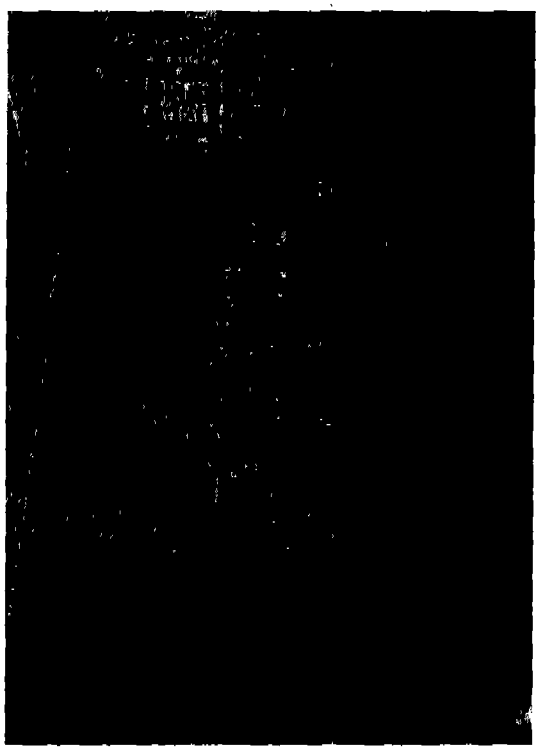


Fig.13 The appearance of a polishing tool

A robot off-line programming system makes it possible for users to build a polishing workcell, and simulate a robot polishing operation accurately on a graphic terminal. An iterative robot inverse kinematic algorithm in this research has eliminated the need of computation of inverse Jacobian, and proposed near initial guesses to increase the computational speed and reliability. By a simplified collision check algorithm, collisions among the parts modeled as unions of convex polyhedra are checked quickly. The proposed approach to path generation on polishing surfaces produces safe paths, where the tool does not collide with the side-walls of die. The RDB manages all of the polishing data on the task level. A task organizes all of the interconnected polishing data with multiple relations.

For future work, more studies on the abrasives for better smoothness and flatness and the design

of a tool mechanism suitable for robot polishing are required. More work should be done to improve the off-line simulation, and develop an expert system to determine the optimal tooling conditions from die materials, heat treatment and desired roughness and waviness.

REFERENCES

1. Kalpakjian, S., "Manufacturing Engineering and Technology", Addison-Wesley Publishing Company, U.S.A., pp. 777-779 , 1989.
2. Kunieda, M. and Nakagawa, T. and Higuchi, T., "Development of Polishing Robot for Free Form Surface", Proceedings of the 5th International Conference on Production Engineering, pp. 265-270, 1984.
3. Saito, K., "Finishing and Polishing of Free-Form Surface", Bulletin of Japan Society of Precision Engineering, Vol. 18, No. 2, pp. 104-109, 1984.
4. Saito, K. and Miyoshi, T., "Automation of polishing process for a cavity surface on dies and molds by using an expert system", Annuls of CIRP, Vol. 42, No. 1, pp. 553-556, 1993.
5. Bennaton, J. A., "Integrated Systems for Off-line Programming", The Industrial Robots, Vol. 15, No. 4, pp. 216-218, 1988.
6. Carter , S., "Off-line Programming : the state-of-the-art", The Industrial Robots, Vol. 14, No. 4, pp. 213-215, 1987.
7. Kim, M. S., "Off-line Programming System for Polishing Sculptured surface", ICDM'90, pp. 200-207, 1990.
8. Buchal, R. O., Cherchas, D. B., Sassani, F. and Duncan, J. P., "Simulated Off-Line Programming of Welding Robots", The International Journal of Robotics Research, Vol. 8, No. 3, pp. 31-43, 1989.
9. Bobrow, J. E., " A Direct Minimization Approach for Obtaining the Distance between Convex Polyhedra", The International Journal of Robotics Research, Vol. 8, No. 3, pp. 65-76, 1989.
10. Pieper, D., "The Kinematics of Manipulators Under Computer Control", Ph.D. Thesis Stanford University, U.S.A., 1968.
11. Craig, J. J., "Introduction to Robotics Mechanics & control", Addison-Wesley Publishing Company, U.S.A., pp. 129-131, 1986.
12. Kazerounian, K., "On the Numerical Inverse Kinematics of Robotic Manipulators" ASME Journal of Mechanisms, Transmissions and Automation in Design, Vol. 3, pp. 190-196, 1987.
13. Kang, S. C. and Kim, M. S., "An efficient iterative inverse kinematic analysis for general robot manipulator using near position", Journal of Korean Society of Mechanical Engineers, pp. 1640-1648, 1991.
14. Gupta, K. C., "Kinematic Analysis of Manipulator Using the Zero Reference Position Description", International Journal of Robotic Research, Vol. 5, No. 2, pp. 5-13, 1986.
15. Fox, R. L., "Optimization Methods for Engineering Design", Addison-Wesley, U.S.A., 1971.
16. Cho, S. H., Park, K., Kang, S. C. and Kim, M. S., "The Development of Robot Off-line Programming System for Die Polishing", Journal of Korean Society of Mechanical Engineers, Vol. 15, No. 4, pp. 1387 -1397, 1991.
17. Kim, M. S., "A New Calibration System", CIRP-Conference, Berlin, 1990.
18. Kim, M. S., You, H. S. and Jang, H. S., "Measuring robot performance and calibration system", Korea Automation and Control Conference, pp. 596-601, 1990.
19. Kim, M. S., Park, J. O., Kang, S. C., Uhm, D. G., Annual report on "Development of Automatic Die Polishing System", pp. 101-170, 1992.