Hardware Realization of Inverse Kinematics for Robot Manipulators

Seo-Wook Park and Jun-Ho Oh

Abstract—**For real-time processing of kinematic information required for intelligent robotic applications, a hardware realization of an inverse kinematics algorithm is a challenging task. This paper adopts an incremental unit computation method to accomplish the inverse kinematics of a three-axis articulated robot. This method starts from defining incremental units in joint and Cartesian spaces, which represent the position resolutions in each space. With this approach, calculation of the inverse Jacobian matrix can be realized through a simple combinational logic gate circuit. Furthermore, the incremental direct kinematics can be solved by using a digital differential analyzer (DDA) integrator. The hardware architecture to implement the algorithm is also described. Applying the hardware implemented by an erasable programmable logic device (EPLD) to the straight-line trajectory of an experimental robot, we have obtained the end-effector's maximum speed of 12.6 m/s.**

I. Introduction

In order to make robot manipulators adaptable to an unknown working environment, it is necessary to obtain sensory information from the latter. However, the computational requirement for real-time processing of the information places a heavy burden on computational resources. Therefore, for future intelligent robot manipulators, it is required to develop a special-purpose processor that implements various robotic functions such as sensor signal processing, trajectory planning, man–machine interface, coordinate transformation, and dynamic control. In recent years, a remarkable progress in VLSI and parallel processing technologies offers an opportunity to meet the requirement [1].

When a general-purpose processor such as a microcomputer is adopted, an algorithm has been focused on the software viewpoint. On the other hand, in the case of a special-purpose processor, its architecture strongly depends on the adopted algorithm. That is, the use of a floating-point arithmetic, the width of datapaths, and the type of operators have a significant influence on the complexity of the architecture.

From the hardware implementation viewpoint, the conventional numerical approaches for the inverse kinematics algorithm have several difficult points. Namely, complicated multiplication and trigonometric function calls are needed to perform a repetitive calculation of the inverse Jacobian matrix. To meet a desired accuracy and speed, a floating-point arithmetic processor and a sufficiently wide datapath are usually required [2]. The requirement leads to a complex hardware design, an increased system size, and a high cost.

In this paper, we adopt an inverse kinematics algorithm based on an incremental unit computation method, which has several attractive features for the hardware implementation. In the following section, the algorithm will be briefly described. The procedure of mapping the algorithm into a hardware implementation will also be presented.

Manuscript received February 26, 1993; revised September 19, 1993.

S.-W. Park was with the Department of Precision Engineering and Mechatronics, Korea Advanced Institute of Science and Technology. He is now with the New Automotive Electronics Development Team, Hyundai Electronics Industries Company, Ltd., San 136-1, Ami-ri, Bubal-eub, Ichon-kun, Kyoungki-do, 467-860, South Korea.

J.-H. Oh is with the Department of Precision Engineering and Mechatronics, Korea Advanced Institute of Science and Technology, 373-1 Kusong-dong, Yusong-gu, Taejon, 305-701, South Korea.

IEEE Log Number 9214347.

II. Incremental Unit Computation Method

A. Overview

The target robot under consideration, shown in Fig. 1, is frequently adopted as a major linkage mechanism for a standard 6 DOF industrial robot. The three-axis articulated robot can be represented or decomposed into two kinds of two-axis planar robot. Namely, if the robot is projected onto the XY plane, it can be considered as a polar robot whose arm length is given by

$$r = l_1 \cos\theta_2 + l_2 \cos\theta_3. \tag{1}$$

Furthermore, the target robot can be thought of as a two-axis planar articulated robot in the RZ plane. Since the inverse kinematics for a two-axis robot is a straightforward problem, the development of the algorithm becomes much easier by using the above-mentioned approach. In the following sections, the incremental unit computation method is described first for the polar robot, then for the two-axis planar robot. Then, inverse kinematics for the overall three-axis robot is constructed from the two simpler solutions.

B. Polar Robot

1) Incremental Unit: When a rotary actuator has a reduction gear and an incremental encoder as a feedback device, the resolution of the joint motion can be expressed by

$$\Delta\theta_1 = \frac{2\pi}{nN} \text{ (rad)} \tag{2}$$

where n denotes the resolution of the encoder (pulse per revolution, ppr) and $N(> 1)$ gear ratio. The resolution $\Delta\theta_1$ is the smallest allowable position increment, and is termed the *joint incremental unit* (JIU).

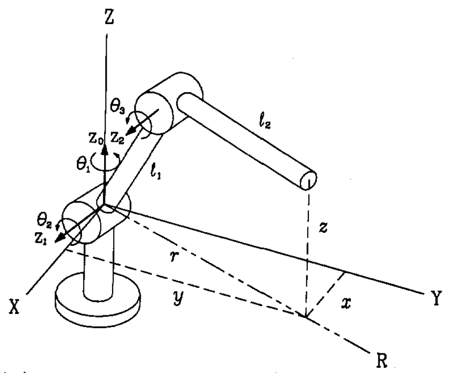


Fig. 1. Schematic of the three-axis articulated robot.

TABLE I
SIGN TABLE OF THE INVERSE JACOBIAN MATRIX FOR THE POLAR ROBOT

CIU	dx		dy	
JIU	$d\theta_1$	dr	$d\theta_1$	dr
Quadrant				
1	−	+	+	+
2	−	−	−	+
3	+	−	−	−
4	+	+	+	−

The resolution at the end tip corresponding to the JIU is given by

$$\Delta r = r\,\Delta\theta_1. \tag{3}$$

Note that the resolution is not constant and depends on the arm length, so addressable points of the polar robot have a nonuniform distribution in Cartesian space. Let us define the worst resolution as the *Cartesian incremental unit* (CIU) ΔR:

$$\Delta R \equiv R\,\Delta\theta_1 \tag{4}$$

where R is the arm length when the robot is fully stretched out. We choose the CIU as the quantum to represent the position of the end tip.

2) Inverse Jacobian Matrix: The inverse Jacobian matrix relates the Cartesian increment to joint one, and depends on the robot configuration. For the polar robot, the inverse Jacobian matrix is

$$J^{-1}(\vec{\theta}) = \begin{bmatrix} -\frac{1}{r}\sin\theta_1 & \frac{1}{r}\cos\theta_1 \\ \cos\theta_1 & \sin\theta_1 \end{bmatrix} \tag{5}$$

where $\vec{\theta} = [\theta_1 \quad r]^T$.

Our strategy for solving the inverse Jacobian matrix is to restrict the joint increments to the JIU for each iteration. Subsequently, the only necessary information for a joint command is a sign of the increment, and this can be determined by taking the sign of elements of the inverse Jacobian matrix. The sign of the joint increment corresponding to the quadrant of the robot and the end tip command is summarized in Table I. This table can easily be realized through a combinational logic gate circuit.

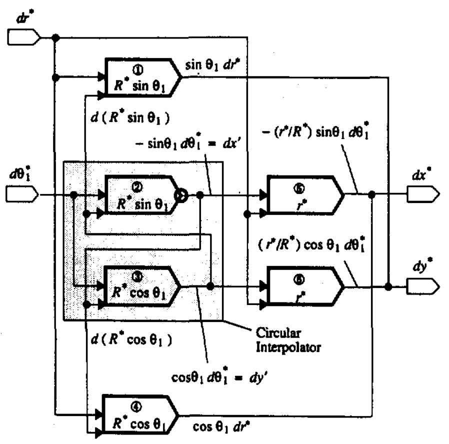


Fig. 2. Schematic of DDA block for solving the incremental direct kinematics of the polar robot.

3) Incremental Direct Kinematics: To check the error between the desired and estimated position of the end tip, the computation of direct kinematics must be performed for each iteration. If we digitize the end tip coordinate by the CIU, the incremental direct kinematic equations are derived:

$$\begin{aligned} dx^* &= int[\cos\theta_1\,dr^* - (r^*/R^*)\sin\theta_1\,d\theta_1^*] \\ dy^* &= int[\sin\theta_1\,dr^* + (r^*/R^*)\cos\theta_1\,d\theta_1^*] \end{aligned} \tag{6}$$

where

$$dr^* = d(r/\Delta R),\ d\theta_1^* = d(\theta_1/\Delta\theta_1),\quad r^* = int[r/\Delta R],\ \text{and}\ R^* = int[R/\Delta R].$$

Equation (6) can be solved by a digital differential analyzer (DDA) integrator, which gives a digital solution of a differential equation at high speed [3]. A schematic of the DDA block for solving the increment direct kinematics is shown in Fig. 2: the integration constants of each DDA must be adjusted to $C = 1/R^*$. The DDA integrator can be implemented in a hardware circuit consisting of an adder, up/down counter, and register.

4) Algorithm: With the previously described method, the inverse kinematics algorithm is summarized as follows.

Step 1: The end tip command is given by a sensory feedback or performing interpolation for a desired trajectory. Note that the command takes the form of a sequence of the CIU's.

Step 2: From Table I, determine the joint increments corresponding to the end-tip command and the current quadrant of the robot.

Step 3: By using the DDA block shown in Fig. 2, obtain Cartesian increments in response to the joint increments given in Step 2.

Step 4: Compare the obtained Cartesian increments with the end-tip command, and repeat the procedure from Steps 2–4 until an error criterion is satisfied.

In this way, we can obtain a sequence of the joint increments, which is fed as a reference signal to a joint servo system.

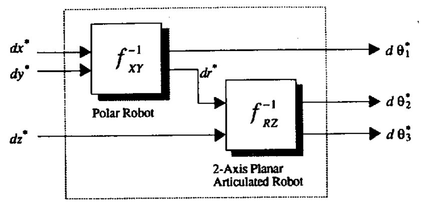


Fig. 3. Proposed coordinate transformer.

C. Two-Axis Planar Articulated Robot

An inverse kinematics algorithm of this type of robot is very similar to that of the polar robot. In this case, the CIU is defined as follows:

$$\Delta L \equiv \max[\Delta l_1, \Delta l_2] \tag{7}$$

where

$$\Delta l_1 = l_1 \Delta\theta_2, \ \Delta l_2 = l_2 \Delta\theta_3.$$

We can solve the inverse Jacobian matrix by adopting the same strategy used in the previous section. The incremental direct kinematics solution can also be obtained through the circular interpolator, which consists of two cross-coupled DDA integrators (see Fig. 2). In general, an interpolator is the major part of the CNC machine tools, and its function is to generate coordinated movement of the separately driven axes of motion in order to achieve a desired path of the tool relative to the workpiece [3]. Typical interpolators are capable of generating linear, circular, and parabolic paths. A more detailed description of the algorithm appears in [4].

D. Coordinate Transformer

When the inverse kinematics of the polar robot is denoted by f_{xy}^{-1}, and that of the two-axis planar articulated robot by f_{rz}^{-1}, the inverse kinematics for the target robot can be accomplished by adopting the coordinate transformer whose structure is shown in Fig. 3. Note that one of the solutions obtained from the polar robot is transferred into the two-axis planar articulated robot in the form of a serial pulse train. Each input pulse of the coordinate transformer is equivalent to one CIU, and each output pulse generates a joint motion corresponding to one JIU.

III. Hardware Realization

In this section, the procedure of mapping the inverse kinematics algorithm proposed in the previous section into a hardware realization will be described.

Generally speaking, an algorithm is a decomposition of a computation into subcomputations with an associated precedence relation [5]. A system that executes the algorithm has a structure consisting of two subsystems: the data subsystem and the control subsystem. The data and control subsystems communicate by means of control signals and conditions. The conditions, generated by the data subsystem, are used by the control subsystem in data-dependent signals.

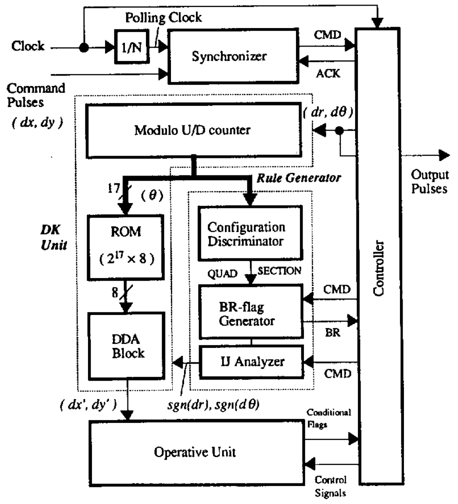


Fig. 4. Hardware architecture for implementing the inverse kinematics algorithm of the polar robot.

Based on the above-mentioned basic structure, we now discuss a hardware architecture of the proposed coordinate transformer.

A. Hardware Architecture for Polar Robot

The overall block diagram of the hardware architecture for implementing inverse kinematics algorithm of the polar robot is shown in Fig. 4. The hardware architecture consists of five major units: synchronizer, rule generator, direct kinematic (DK) unit, operative unit, and controller. In this section, we discuss the function of these units.

1) Synchronizer: The function of the synchronizer is to synchronize the command pulses with the system clock. The command pulses (dx, dy) may overlap, but the outputs of the synchronizer are separated by clocking them on opposite phases of a polling clock. The polling clock is obtained by dividing the system clock by N. The clock divisor N has to be carefully determined by considering the required maximum number of system clock pulses in response to single command pulse.

2) Rule Generator: It is composed of a configuration discriminator, inverse Jacobian (IJ) analyzer, and branching flag generator.

- *The configuration discriminator* determines the quadrant and section that the polar robot lies in, which are later used as inputs of the remaining units. The section is used to determine the joint increment which makes a dominant contribution to the given end tip command.
- *The IJ analyzer* is a combinational logic circuit that realizes the sign table of the inverse Jacobian matrix (i.e., Table I). It generates the direction bits of joint increments according to the end-tip command and the quadrant of the robot.
- *The branching flag generator* produces flags used to branch a control flow of the controller. The flags indicate

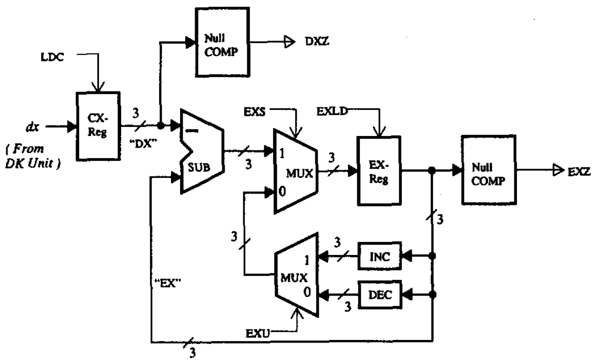


Fig. 5. Block diagram of the operative unit for the polar robot.

TABLE II
REQUIRED OPERATIONS FOR OPERATIVE UNIT OF POLAR ROBOT

EXS	EXU	EXLD	Functions
0	0	↑	EX← EX−1
0	1	↑	EX← EX+1
1	×	↑	EX← EX−DX

where × represents "don't care."

which joint makes a dominant contribution to the given end-tip command.

3) DK Unit: This unit solves the incremental direct kinematics. In other words, this unit yields Cartesian increments that correspond to joint increments given by the controller and the IJ analyzer. The DK unit is directly realized from Fig. 2. The circular interpolator, however, adopts the direct search method to enhance the accuracy of the solution [7]. It is implemented through employing a ROM look-up technique. A modulo-m up/down counter is used to address the ROM that contains precalculated Cartesian increments.

4) Operative Unit: The function of the operative unit is to compare the end-tip command with Cartesian increments generated by the DK unit. To obtain this function, simple operations as shown in Table II are required. Fig. 5 shows the block diagram of the operative unit to implement the function. In this figure, DX denotes the Cartesian increment obtained from the DK unit, and EX is the position error of the end tip. This unit has four control points—LDC (LoaD Cartesian increment), EXS (Select EX), EXU (Update EX), and EXLD (LoaD EX)—and two conditional flags—DXZ (DX Zero) and EXZ (EX Zero). The input of this unit is the control points and the output is the conditional flags, which are fed to the controller. Through a simulation, the width of datapaths for "EX" and "EY" has been designed to be a 3-bit, which is sufficient to guarantee that an overflow never occurs. Thereby, the operative unit requires several simple operators of just a 3-bit size.

5) Controller: The controller generates a sequence of control signals that is distributed to the previously described units. This unit plays a central role among the units of the architecture, and it is closely related to the efficiency of the algorithm implementation. In this paper, the Moore approach is used to implement the controller. Thus, the control flow of

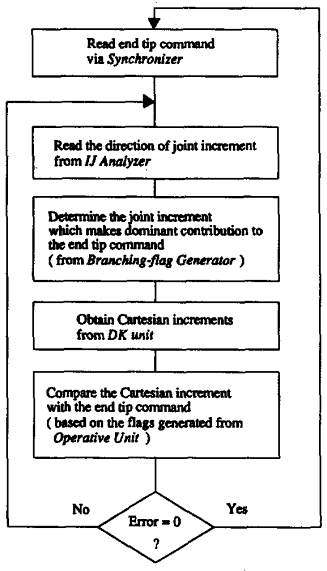

Read end tip command via *Synchronizer*
Read the direction of joint increment from *IJ Analyzer*
Determine the joint increment which makes dominant contribution to the end tip command (from *Branching-flag Generator*)
Obtain Cartesian increments from *DK unit*
Compare the Cartesian increment with the end tip command (based on the flags generated from *Operative Unit*)
No
Error = 0 ?
Yes

Fig. 6. Block diagram of the controller for the polar robot.

the algorithm is performed according to conditional flags, and synchronous control signals with the system clock are directly generated from each state. Fig. 6 shows the block diagram of the controller.

B. Hardware Architecture for Two-Axis Planar Articulated Robot

The overall hardware architecture for the two-axis planar articulated robot is very similar to that of the polar robot, but each unit has a slightly different structure. The DK unit consists of just circular interpolators, which are implemented by the ROM look-up technique described in the previous section. The operative unit has more or less complex structure in comparison with that of the polar robot. Its datapath width has been designed to be a 4-bit. The structure of the controller is also similar to that of the polar robot.

C. Implementation

The hardware architecture described in the previous section can readily be implemented in a gate level. We have used an erasable programmable logic device (EPLD) to realize the hardware architecture. The EPLD's used are EPM5064, EPM5128, and EPM5192 made by the Altera Corporation. The EPLD provides many advantages such as smaller board size, higher reliability, increased flexibility, and so on. The system clock has been chosen to be 8 MHz. The clock divisor N has been designed to be 32; thus, the polling clock becomes 250 kHz. The polling clock determines the maximum allowable frequency of the command pulse. For the straight-line trajectory of the experimental robot whose specification is given in Table III, the end-effector's maximum speed of 12.6 m/s has been achieved. This speed is fast enough to realize real-time processing of kinematic information required for intelligent robotic applications in an unknown environment.

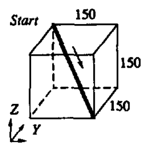


Fig. 7. Test trajectory.

TABLE III
SPECIFICATION OF THE EXPERIMENTAL ROBOT

Joint	Power (W)	Gear Ratio	Encoder Resolution
1	165	1:79	1000 ppr
2	67	1:50	1000 ppr
3	67	1:50	1000 ppr

Link length: $l_1 = l_2 = 400$ mm.

The test trajectory, described as shown in Fig. 7, requires motion of three degrees of freedom.

IV. CONCLUSIONS

The hardware architecture to implement the inverse kinematics algorithm of the three-axis articulated robot has been described. The adopted algorithm has many attractive features from the hardware implementation viewpoint. That is, the algorithm entirely removes the necessity of a floating-point arithmetic and trigonometric function. Instead, the algorithm can be accomplished by using several preliminary operators such as adder, comparator, and so on, and calls for just narrow width of datapath: 3 or 4 bit. The evaluation of the inverse Jacobian matrix has easily been realized though a combinational logic gate. Moreover, the incremental direct kinematics has been accomplished by introducing a DDA integrator. The designed hardware architecture has been implemented by using an EPLD. For the straight-line trajectory of the experimental robot, we have obtained the end-effector's maximum speed of 12.6 m/s.

APPENDIX

The *DDA integrator* is an incremental device which is suitable for digital solutions of differential equations in real time. The basic operation of the integrator is performed by successive additions using a rectangular approximation method. Its operation is described briefly in this Appendix.

If we perform a digital integration for a variable, $p = f(t)$:

$$z(t) = \int_0^t p\,dt \cong \sum_{i=1}^{k} p_i\,dt \tag{A.1}$$

then the value of $z(t)$ at $t = k\,dt$ is denoted by z_k, which may be written as follows:

$$z_k = \sum_{i=1}^{k-1} p_i\,dt + p_k\,dt \tag{A.2}$$

or

$$z_k = z_{k-1} + dz_k \tag{A.3}$$

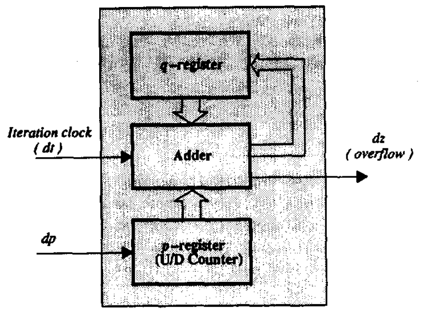
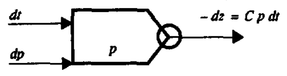


Fig. 8. Schematic diagram of the DDA integrator.



• A circle denotes that the sign of the output is changed from +dz to −dz.

Fig. 9. Symbolic representation of the DDA integrator.

where

$$dz_k = p_k\,dt. \tag{A.4}$$

The value of p_k is computed by adding the increment dp_k to, or subtracting it from, the previous value:

$$p_k = p_{k-1} \pm dp_k. \tag{A.5}$$

From (A.3), (A.4), and (A.5), the digital solution z_k is obtained in an iterative manner. Actually, the DDA integrator consists of two n-bit registers (one is denoted the p register, which is an up/down counter, and the other the q register, which is a temporary memory), and one binary adder. A schematic diagram of the integrator is shown in Fig. 8.

In general, the output increment dz of the DDA integrator is given by

$$dz = Cp\,dt \tag{A.6}$$

where p denotes the current value of the p register and C is the DDA integration constant, which is proportional to the iteration frequency $f = 1/dt$ and inversely proportional to the capacity of the integrator S, i.e.,

$$C = \frac{f}{S}. \tag{A.7}$$

Fig. 9 shows a symbolic representation of the integrator.

REFERENCES

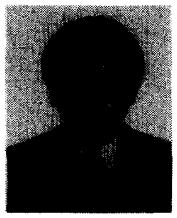
[1] M. Kameyama and T. Higuchi, "VLSI computer for robotics," *J. Robotics Soc. Japan*, vol. 6, pp. 64–70, Aug. 1988.

[2] Y. T. Tsai and D. E. Orin, "A strictly convergent real-time solution for inverse kinematics of robot manipulators," *J. Robotic Syst.*, vol. 4, no. 4, pp. 477–501, 1987.

[3] Y. Koren, *Computer Control of Manufacturing Systems.* New York: McGraw-Hill, 1983.

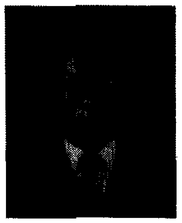
[4] J.-H. Oh and S.-W. Park, "A new approach for solving the inverse kinematics in real-time applications," *SME Trans. Robotics Res.*, vol. 1, pp. 3/17–3/28, 1990.

[5] M. D. Ercegovac and T. Lang, *Digital Systems and Hardware/Firmware Algorithms.* New York: Wiley, 1985.
[6] Y. Koren and O. Masory, "Reference-pulse circular interpolators for CNC system," *Trans. ASME, J. Eng. Ind.*, vol. 103, pp. 131–136, Feb. 1981.



Seo-Wook Park was born in Chinju, South Korea, in 1963. He received the B.S. degree in mechanical engineering from the Hanyang University, the M.S. degree in production engineering from the Korea Advanced Institute of Science and Technology (KAIST), and the Ph.D. degree in precision engineering and mechatronics from the KAIST, in 1986, 1988, and 1993, respectively.

Since then, he has been employed by the Hyundai Electronics Industries Corporation. He is now an Assistant Manager in the New Automotive Electronics Development Team. His research interests are in the areas of computer-controlled robotics, compliance control, numerical control systems, and microprocessor application systems. His current interests include the design of an electronic control unit for airbag systems, a car crash detection algorithm, and the development of automotive electronics systems such as an anti-lock bracking system, navigation system, and engine management system.



Jun-Ho Oh was born in Seoul, South Korea, in 1954. He received the B.S. and M.S. degrees in mechanical engineering from Yonsei University in 1977 and 1979, respectively, and the Ph.D. degree in mechanical engineering from the University of California, Berkeley, in 1985.

Since then, he has been with the Korea Advanced Institute of Science and Technology (KAIST) as a faculty member. He is now an Associate Professor of Precision Engineering and Mechatronics at KAIST. His interests are in the areas of adaptive control, artificial intelligence, vision systems, and robotics. His research is focused on the realization of complicated control theories in actual applications. His current research themes include multirate generalized predictive control of a MIMO system, design of a self-organizing fuzzy controller, a control for a flexible arm, gait control of a walking robot, design of a servo-type precision accelerometer, and moving target tracking by visual information.