TM-1032

An Application of CAL to Robotics

by
S. Sato & A. Aiba

February, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

An Application of CAL to Robotics

Shinichi Sato and Akira Aiba

ICOT Research Center
4-28 Mita 1-chome, Minato-ku, Tokyo 108, JAPAN

January 7, 1991

1 Constraint Logic Programming Experimental System CAL

We previously developed a constraint logic programming experimental system CAL[4] which stands for "Contrante Avec Logique" as one of the attempts of CLP. This is a configration of the system. Although CAL has a similar framework which is the combination of the logic programming and the constraint programming as other CLP languages, it has the feature of being able to handle non-linear polynomial equations on complex numbers by employing the Buchberger Algorithm[1] as one of the constraint solvers. The Buchberger algorithm calculates a canonical form of system of equations called Gröbner Bases. This algorithm has been widely used in the field of computer algebla in these years.

2 Application of thc Greobner bases(in RISC)

At Research Institute for Symbolic Computation (RISC) of Johanes-Kepler university, there has been many researches on its application to various fields as these. Among these researches, one of the most practical problems is an application to handling robot kinematics. For a given handling robot, it deals with the problem of determining the extension of each arm and the rotation angle of each joint that will result in a target position and an orientation of the end effector. It is a typical problem of handling non-linear equations. In their research, the main work has been a mathematical evaluation of computing Gröbner bases. We also took the similar robotics problem, but tried to write and solve the problem on oyr system CAL. By applying CAL to the problem, having the merits of both of logic programming and Gröbner bases, we intend to write more feasible and useful program easily as well as to evaluate the computing efficiency.

3 Programs for Handling Robots on CAL

In this presentation, we introduce two programs for the control of handling robots as an example of engineering application of CAL, one is a program for kinematics and the other is for static

dynamics. First, we will formalize the kinematics and static dynamics of handling robots. Handling robot is a kind of industrial robot like this which picks up an object with its end-effector and changes its position and orientation by rotating each joint and changing the length of each arm. To move an object to the desired position and the orientation, we need to control the rotation angle of each joint and the length of each arm of the robot. Kinematics represents the relation among the target position and orientation of the end-effector, and the extension of each arm and the rotation angle of each joint. Note that the position and the orientation of the end-effector are same as these of an object. Static dynamics represents the relation between the torque working on each joint and the force working on the end-effector.

4 Definition of the Position and the Orientation of an object

Then, before formalizing kinematics of handling robots, we will define parameters of the position and orientation which represent a state of an object held by the end-effector. The position of the object is expressed by the three dimensional vector P (px, py, pz), which indicates the center of the object. The orientation is expressed by two unit vectors a (ax, ay, az) and b (bx, by, bz), which are perpendicular to each other. These vectors are fixed on the object and their directions will be changed by its rotation.

5 Definition of the position and Orientation Matrix

Then to represents the position and orientation together, we define the position-orientation matrix $[P^{**}]$ like this. We abbreviate it as *p-o matrix* here after.

$$[P^{**}] = \begin{pmatrix} x & ax & bx \\ y & ay & by \\ z & az & bz \\ 1 & 0 & 0 \end{pmatrix}.$$

Remark that any state of an object that is, position and orientation, can be expressed by this matrix. Although the matrix contains 9 variables, there are 3 redundancies among the components of the orientation vectors because they are 2 unit vectors perpendicular to each other.

$$\begin{aligned} ax^2 + ay^2 + az^2 &= 1 \\ bx^2 + by^2 + bz^2 &= 1 \\ ax * bx + ay * by + az * bz &= 0 \end{aligned}$$

Therefore, actual degree of freedom of the *p-o matrix* is 6. In other words, any state of an

object has 6 degree of freedom. Therefore, by using a robots having 6 degree of freedom, we can transform the state of an object to any position and any orientation. In this matrix the 1 and 0's on the fourth row are components added for convenience of calculation managing rotation and straight movement of an object together and therefore, not related to degree of freedom.

6 The Transformation of the Position and Orientation

By the way, any transformation of an object in three dimensional space can be expressed by a combination of the straight movement and the rotation. Any state of an object can be expressed by the $P-O$ *matrix* defined before. If you rotate an object having a state of $[P^{**}]$ by an angle θ around an axis $W(=(Wx, Wy, Wz))$ crossing its center, then move it by vector $dP(=(dx, dy, dz))$, the new *p-o matrix* $[P'^{**}]$ can be expressed as the multiplication of $[P^{**}]$ by matrix $[M]$ like this.

$$
\begin{aligned}
[P'^{**}] &= [M][P^{**}] \\
[M] &= \begin{pmatrix} E & dP \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} e_{11} & e_{12} & e_{13} & dx \\ e_{21} & e_{22} & e_{23} & dy \\ e_{31} & e_{32} & e_{33} & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
$$

In these expression, $[M]$ is a matrix called *transformation matrix* which contains both of rotation and straight movement transformation . Each element of the rotation matrix E is determined by the rotation axis W and the rotation angle θ.

7 Kinematics of Handling Robots

As we have shown, the movement of the end-effector is performed by the rotation of each joint and the extension of each arm. As they respectively correspond to the rotation and the straight movement, the transformation of the state of the end-effector can be expressed by the sequential multiplication of transformation matrix corresponding to each arm and joint. In general, the relation among the *p-o matrix* P^{**} of the end-effector and each transformation matrix M_i of a handling robot having m joints is described by this expression.

$$
\begin{aligned}
[P^{**}] &= \begin{pmatrix} E_1 & R_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} E_2 & R_2 \\ 0 & 1 \end{pmatrix} \cdots \begin{pmatrix} E_i & R_i \\ 0 & 1 \end{pmatrix} \cdots \begin{pmatrix} E_m & R_m \\ 0 & 1 \end{pmatrix} \begin{pmatrix} G_0 & a_0 & b_0 \\ 1 & 0 & 0 \end{pmatrix} \\
&= [M_1][M_2] \ldots [M_i] \ldots [M_m][G^{**}], \qquad (1)
\end{aligned}
$$

This expression represents the Kinematics of handling robots. In this expression, $[P^{**}]$ is the p-o matrix which represents the target position and the target orientation of the end-effector. $[G^{**}]$ is the p-o matrix which represents initial grip vector and orientation. E_i is the rotation matrix of the i-th joint, R_i is the vector of the i-th arm before rotation,

and m is the number of joint. Each element of the transformation matrix $[M_i]$ is determened by the rotation angle θ_i, the rotation axis W_i of the i-th joint and the i-th arm vector R_i before rotation. Therefore , we call θ_i, (Wx_i, Wy_i, Wz_i), and (Rx_i, Ry_i, Rz_i), *transformation matrix* parameters because they determines the content of each *transformation matrix* $[M_i]$ in this expression.

8 Statics Dynamics of Handling Robots

Next, we formalize static dynamics of handling robots. When a force works on the end-effector, a torque works on each joint. To rotate a joint, the robot has to rotate the mortor shaft of the joint against the torque. If the torque is stronger than the torque permission of the mortor, the robot cannot rotate the joint. Thus, we need to calculate the torque working on each joint. The torque of the i-th joint is calculated as functions of the position of the joint, and the position of the end-effector, and the force working on the end-effector like this expression. Note that the only force we consider here is the weight of the object held by the end-effector.

$$T_i = Z_i \cdot (P_m - P_i) \times F_m \tag{2}$$

In this expression,

Z_i is the direction vector of the rotation axis of the i-th joint. P_m is the position vector of the end-effector. P_i is the position vector of the i-th joint. And F_m is the force vector working on the end-effector.

Among these parameters, as Z_i, P_i, and P_n changes according to the movement of the robot, we need to calculate Z_i, P_i, P_n, and T_i every time the robot finishes moving one of the arms or rotating one of the joints. By calculating the position of each joint, we can recognize the whole orbit of robot hand on that time. Therefore, we can check whether certain obstacle will be in the way of the orbit, or not.

9 The Program for Kinematics

We developed two CAL programs. One is for kinematics, and the other is for static dynamics. Kinematics program calculates the value of each *transformation matrix* parameter from the target position and the target orientation parameters by applying Kinematics expression.

In kinematics program, the head of the predicate on the top level is as follows.

$robot(Mlist, Gx, Gy, Gz, Ax_0, Ay_0, Az_0, Bx_0, By_0, Bz_0, Px, Py, Pz, Ax, Ay, Az, Bx, By, Bz)$

In this predicate, (Gx, Gy, Gz) is the intial grip vector on the end-effector, $(ax_0, ay_0, az_0), (bx_0, by_0, bz_0)$ are the intial orientation of the end-effector. (Px, Py, Pz) is the target position of the object and $(ax, ay, az), (bx, by, bz)$ are the target orientation of the object. $Mlist$ is the list of

transformation matrix parameters like this.

$$[[cosm, sinm, Rx_m, Ry_m, Rz_m, Wx_m, Wy_m, Wz_m],$$
$$\ldots$$
$$\ldots$$
$$[cos2, sin2, Rx_2, Ry_2, Rz_2, Wx_2, Wy_2, Wz_2],$$
$$[cos1, sin1, Rx_1, Ry_1, Rz_1, Wx_1, Wy_1, Wy_3]]$$

In Mlist, you can find *sin*'s and *cos*'s, in place of the rotation angle θ. Since our system cannot handle tigonometrical function directly, we are applying $sin\theta$ and $cos\theta$ in place of θ as an variable respectively. Of course, the sum of the square of each of them is forced to be 1. As you can see, since *transformation matrix* parameters can be expressed by lists like them, the program can handle any structure of handling robot by manipulating the contents of them.

10 Vector Sketch of a robot

Let us consider, for example, a robot having 3 arms and 3 joints expressed by this vector sketch. The robot can rotate each joint and change the length of each arm. Therefore, the robot has 6 degree of freedom. If you want to know the general relation among the target position and the target orientation, and the rotation angle of each joint and the length of each arm, you can obtain the result by this query.

11 example1

$robot([[cos3, sin3, 0, 0, z_3, 0, 0, 1],$
$[cos2, sin2, x_2, 0, 0, 1, 0, 0],$
$[cos1, sin1, 0, 0, z_1, 0, 0, 1]],$
$5, 0, 0, 1, 0, 0, 0, 1, 0,$
$px, py, pz, ax, ay, az, cx, cy, cz).$

In this query, px,py, and pz represent the target position, and ax, ay, az, cx, cy, and cz are components of two unit vectors which represent the orientation of the end-effector. *Sin*'s and *cos*'s represent the rotation angle of each joint, and z_3, x_2, z_1 are the final length of each arm. This is the answer to the query.

$$\begin{aligned} cos2^2 &= 1 - sin2^2 \\ cos1^2 &= 1 - sin1^2 \\ cos3^2 &= 1 - sin3^2 \\ px &= -5 * cos2 * sin3 * sin1 + z_3 * sin2 * sin1 + 5 * cos3 * cos1 + x_2 * cos1 \end{aligned}$$

$$
\begin{aligned}
py &= 5 * cos3 * sin1 + x_2 * sin1 + 5 * cos1 * cos2 * sin3 - z_3 * cos1 * sin2 \\
pz &= 5 * sin3 * sin2 + z_1 + z_3 * cos2 \\
ax &= -1 * cos2 * sin3 * sin1 + cos3 * cos1 \\
ay &= cos3 * sin1 + cos1 * cos2 * sin3 \\
az &= sin3 * sin2 \\
cx &= -1 * cos1 * sin3 - cos3 * cos2 * sin1 \\
cy &= -1 * sin3 * sin1 + cos3 * cos1 * cos2 \\
cz &= cos3 * sin2
\end{aligned}
$$

In this way, the parameters of the target position and the orientation are expressed as functions of the length of each arm and the rotation angle of each joint.

Next, we will calculate the values of the length of each arm and the rotation angle of each joint when concrete values are given to all of the target position and orientation parameters. This is the query and the answer in that case.

12 example2

$robot([[cos3, sin3, 0, 0, z_3, 0, 0, 1],$
$[cos2, sin2, x_2, 0, 0, 1, 0, 0],$
$[cos1, sin1, 0, 0, z_1, 0, 0, 1]],$
$5, 0, 0, 1, 0, 0, 0, 1, 0,$
$40, -30, 20, -1/3, 2/3, -2/3, 2/3, 2/3, 1/3).$

$$
\begin{aligned}
sin1^2 &= 4/5 \\
sin2 &= 5/6 * sin1 \\
sin3 &= -1 * sin1 \\
cos2 &= 2/3 \\
cos1 &= -1/2 * sin1 \\
cos3 &= 1/2 * sin1 \\
z_3 &= 26 \\
x_2 &= -105/2 * sin1 \\
z_1 &= 6
\end{aligned}
$$

Therefore, you can get the concrete values of all arm lengthes and joint rotation angles by determing the value of $sin1$ from the answer.

13 The Program for Static Dynamics

The static dynamics programs caluculates the position of each joint and torque working on it from the values of *transformation matrix* parameters by applying static dynamics expression. The head of the predicate on the top level of the program is as follows.

$robot2(Qlist, Mlist, Tlist, Gx, Gy, Gz, Px_m, Py_m, Pz_m, Fx_m, Fy_m, Fz_m)$,

Qlist is the list of position parameters of each joint like this.
$[[qx_1, qy_1, qz_1], [qx_2, qy_2, qz_2], \ldots\ldots\ldots [qx_m, qy_m, qz_m]]$

Mlist is the list of *transformation matrix* parameters same as that of kinematics program.
Tlist is the list of torque paramter working on each joint like this. $[t_1, t_2, \ldots t_m]$
(Gx, Gy, Gz) is same as that of kinematics program
(Px_m, Py_m, Pz_m) is the position of the end-effector
(Fx_m, Fy_m, Fz_m) is the force vector working on the end-effector

The program calculates the values of the parameters in *Qlist* and *Tlist* from the values of the *transformation matrix* parameters in *Mlist*. Of cource, the static dynamics program is also structure-free as kinematics program, and able to describe any type of handling robot by changing the contents of the query to it . We will consider the same robot as this figure and show the example of using both of kinamatics and static dynamics program succesively. This is the query and answer in the same case as shown before.

14 example3

$robot([[cos3, sin3, 0, 0, z_3, 0, 0, 1],$
$[cos2, sin2, x_2, 0, 0. 1, 0, 0],$
$[cos1, sin1, 0, 0, z_1, 0, 0. 1]],$
$5, 0, 0, 1, 0, 0. 0, 1, 0,$
$40, -30. 20, -1/3, 2/3, -2/3, 2/3, 2/3, 1/3).$

$robot2([[qx_1, qy_1, qz_1], [qx_2, qy_2, qz_2], [qx_3, qy_3, qz_3]],$
$[[cos1, sin1, 0, 0, z1, 0, 0, 1],$
$[cos2, sin2, x_2, 0, 0, 1, 0, 0],$
$[cos3, sin3, 0, 0, z_3, 0, 0, 1]],$
$[t_1, t_2, t_3], 5, 0, 0, px_m, py_m, pz_m, 0, 0, fz_m).$

$$
\begin{array}{ccc}
sin1^2 = 4/5 & sin2 = 5/6 * sin1 & sin3 = -1 * sin1 \\
cos2 = 2/3 & cos1 = -1/2 * sin1 & cos3 = 1/2 * sin1 \\
z_3 = 26 & x_2 = -105/2 * sin1 & z_1 = 6 \\
t_1 = 0 & t_2 = -25 * fz_m * sin1 & t_3 = 5/3 * fz_m \\
qx_1 = 0 & qy_1 = 0 & qz_1 = 6 \\
qx_2 = -105/2 * sin1 & qy_2 = 0 & qz_2 = 6 \\
qx_3 = -105/2 * sin1 & qy_3 = -65/3 * sin1 & qz_3 = 70/3 \\
px_m = 40 & py_m = -30 & pz_m = 20
\end{array}
$$

In this way, both of the position parameters of each joint and torque parameters are calculated as functions of $sin1$ and fzm, which is the weight of the object held by the end-effector. By finding the concrete values of those parameters, we can check of the validity of the solutions to the condition of static dynamics.

15 table1 Total Computation Time

We applied our two programs to several robots on the market. The number of joints and freedoms of each robot, and the total time consumed for computing each transformation matrix parameter and torques working on each joint by those programs are on table-1. The total time has been mainly consumed by the kinematics program. It is because kinamatics program includes problem of solving non-linear equations which is problem of computing the Gröbner Bases. On the contrary, the static dynamics program only computes the position of each joint and torque working on it using the solutions computed by the kinematics program. Therefore, the total efficiency mostly depends on that of the kinematics program. As shown in the table, even in the case of the most complicated robot which has 5 joints and 6 degree of freedom, we can obtain a solution in tens of seconds. But actually, the time for computing the Gröbner Bases strongly depends on the arrangement of constraints and the ordering among variables on constraints. If we want to obtain the solutions in time as short as we can, we need to consider these factors. The computation times on table-1 are those of the best cases in our experiments. If we ask for further improvement of computing efficiency, one of the effective methods is to parallelize the Buchberger algorithm and imprement it on parallel inference machine. We are researching about parallelization of CAL and trying to implement such parallel constraint logic programming languages on Multi-PSI machine.

16 discussion

Our approach has the following advanteges compared to the conventional ones. If we apply an approach simply using the Gröbner Bases, at first, we have to deduce complicated equations of an handling robot like expression (1) to handle this kind of problem. Thus, if we get another robot with different structure, then we have to deduce another equations. If we apply the

conventional programming languages, we have to find out a process to solve the system of equations and describe it as a program .

On the contrary, on our approach, the program we have to write on CAL is quite simple because it simply defines matrix multiplications. If we only input a query corresponding to the structure of the robot to the program, then the system will deduce and solve the complicated equations expressing the structure of the robot automatically. Therefore, we can manege any type of handling robot without changing the program, and of course, finding out a process of system of equations. Moreover, once we obtain the general symbolic solutions, then we can reuse it many times as a kind of equational program to obtain the concrete solutions. We are now realizing such a function in the latest version of CAL. Therefore, by applying CAL to this kind of problem, we can expect great improvement of programmer productivity compared to other conventional approaches .

17 Concluding Remark

We have developped two application programs for the control of handling robot on our constraint logic programming system CAL. One is for kinematics and the other is for static dynamics. Our approach which connect the Gröbner Bases with the logic programming has the following advantages. First, by employing the Buchberger algorithm we can deal with non-linear equations deduced by kinematics of handling robots. It is actually the first attempt in CLP languages. Second, by utilyzing Gröbner bases in the framework of CLP, we can handle any type of handling robots by the same program, and manipulating a query to it. Moreover, since we can reuse the symbolic solutions many times as an equational program, we can state that the system will execute a kind of program generation. Such abilities have not been realized in the conventional approach using Gröbner Bases, and of course by robotics programs written in conventional programming languages. Therefore, we can expect great improvements in terms of programmer productivity. As for the efficiency, it much depends on the arrangement of constraints and the ordering of variables on constraints. Thus, we need to consider these factors and manipulate them if we want to obtain a solution as fast as possible. By optimizing it, we can obtain a solution that will result in the concrete target position and orientation of the end-effector in tens of seconds even in case of a complicated robot which has 5 joints and 6 degree of freedom on PSI machine. For more improvements of efficiency, we are now researching about parallelization of CAL, and implementing such parallel constraint logic programming languages on the top of Multi-PSI machine.

References

[1] B. Buchberger. : Applications of Gröbner Bases in Non-linear Computational Geometry, Lecture Notes in Computer Science 296, Trends in Computer Algebra, 1987

[2] A. Colmerauer : A new generation of Prolog Promoses some powerful capabilities, BYTE, pp177-182, 1987

[3] J. Jaffar and J-L. Lassez : Constraint Logic Programming, 4th IEEE Symposium on Logic Programming, 1987

[4] K. Sakai and A. Aiba : CAL: A Thoretical Background of Constraint Logic Programming and its Applications, J.Symbolic Computation 8, pp589-603, 1989

[5] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoum, T. Graf, and F. Berthier : The Constraint Logic Programming Language CHIP, Proceedings of the International Conference on Fifth Generation Computer Systems 1988, 1988

[6] S. Tooyama : Robotics for Machine Engineer, Sougou Densi Shuppansha, 1989

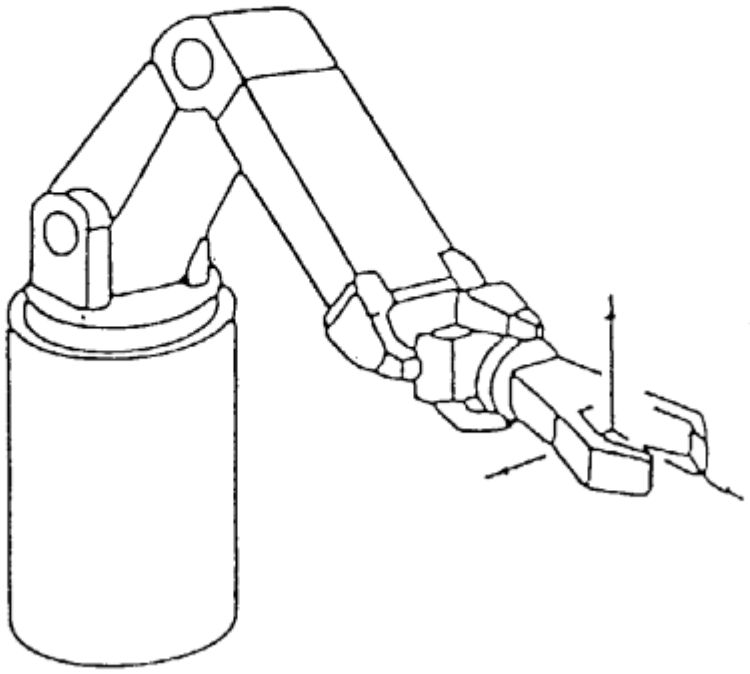

fig1. Typical View of an handling robot

table1. Total Computation Time

robot name	number of joints	degree of freedom	computation time
SR-5	1	3	less than 2 sec
AH−40	2	4	less than 2 sec
AMF Berthatran	3	6	3～6 sec
IRA-50	4	6	4～11 sec
Unimate 2600	5	6	15～60 sec

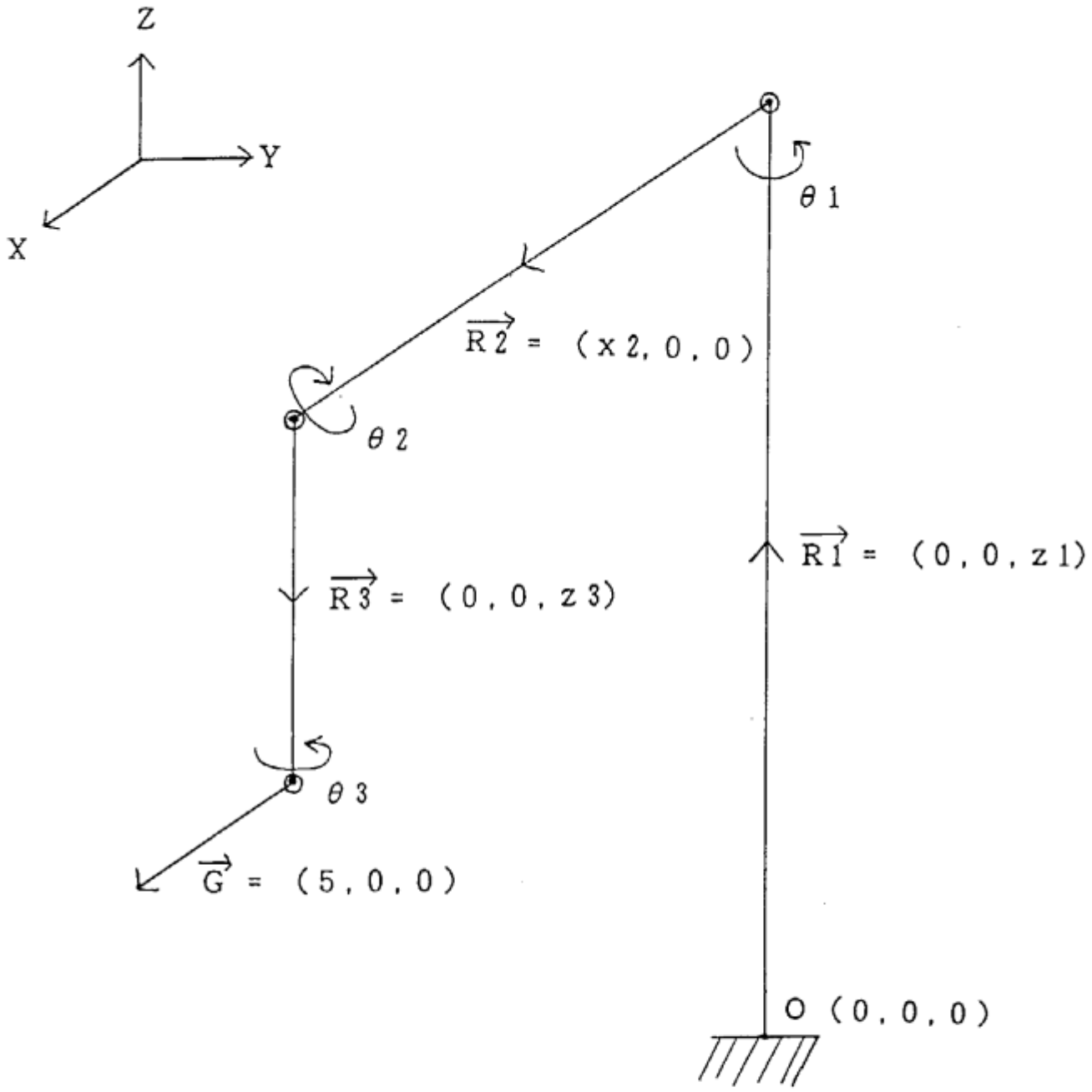

fig2. Vector Sketch of a Robot
(3 joints and 6 degree of freedom)