AN ALGORITHM FOR MOVING A COMPUTER-CONTROLLED MANIPULATOR WHILE AVOIDING OBSTACLES

Eugene Grechanovsky and I.Sh. Pinsker

ABSTRACT

In programming a computer to control a manipulator, the problem of avoiding obstacles present in the environment must be faced. This paper presents a simple world model containing the manipulator as well as obstacles, and describes an algorithm for guiding the manipulator to a specified destination point while avoiding collisions with obstacles. The algorithm combines a number of heuristics with a method for stepwise optimization based on a linear programming technique. A brief description of the computer program AVOID and a sample of computer simulated tasks solved by it are presented.

I INTRODUCTION

One of the main problems in robotics is the problem of devising efficient computer methods to control a manipulator. A typical approach to this problem involves the use of information obtained from visual as well as other sensory feedback for specifying an internal world model in computer memory. Having this model and an appropriate algorithm, the computer would hopefully be able to direct the manipulator to perform tasks requiring some intelligence and dexterity.

II WORLD MODEL

In the situation shown in Figure 1, the "world" consists of a part of the Euclidean plane containing the planar manipulator and the obstacles 0 and 0 . The manipulator is assumed to be a group of four links 1 , 1_2, 1 , 1_4, connected by the revolute joints B , B , B , B_4. The first link l_1 is connected to ground at the origin by the first joint B1 while the endpoint B5 of the last link 1, contains the jaws J, or some other type of terminal device of negligible size and orientation. Thus, to describe a configuration, say, cnf , of the manipulator it will suffice to specify the vector of relative angles $\varphi=(\varphi^1,\varphi^2,\varphi^3,\varphi^4)$. The position of each joint B. ($i=2,\ldots,5$) in the plane is given by the formula

$$B_i = B_i(\varphi^1, \ldots, \varphi^{i-1}) \qquad (1)$$

where the vector functions Bj. are combinations of trigonometric functions. The set of all possible values $\varphi=(\varphi^1,\varphi^2,\varphi^3,\varphi^4)$ forms a 4-dimensional state space Φ. The class of obstacles under consideration is restricted to convex polygons. The ith obstacle has vertices Ai, *1*, Ai,2.., and edges ai,1 a. „,... The position and shape of each obstacle is'assumed to be completely specified.

III BASIC TASK

Assuming the path G (with the initial point G1 and final point G) for the jaws is specified in advance, the task consists in moving the manipulator from the initial configuration to a final one while avoiding obstacles. This problem could be stated as follows: Given the initial configuration of the manipulator and the path G for the jaws, compute a continuous sequence of configurations such that (i) The first configuration in the sequence is the initial configuration; (ii) The jaws in each configuration in the sequence lie on the curve C and in the final configuration coincide with the final point G ; (iii) No configuration in the sequence intersects with any of the obstacles. It is presumed that the jaws in the initial configuration are located at the point G , and that neither the initial configuration nor trie path G intersects with the obstacles.

Our aim is to devise a computationally efficient algorithm for planning manipulator motions, i.e. capable of using the model described above for obtaining a curve in the Φ-space such that the corresponding sequence of configurations should satisfy conditions (i)-(iii) listed above. The major difficulty arises from the fact that the manipulator moves in the plane containing obstacles whereas the state space Φ, in which the control variables φ move, is of an entirely different nature.

IV PRELIMINARIES

First of all, the continuous decision curve in-space and the corresponding continuous configuration sequence are to be replaced with a finite, possibly small, number of points. This is to be done with care, however, since the existence of a finite sequence of permissible configurations (i.e. configurations free of intersections with obstacles) does not guarantee the existence of a continuous permissible sequence. To overcome this difficulty, the obstacles should be replaced with 6-covers, that is, the polygons which contain the obstacles as subsets and have at least €-wide free margins. This procedure has an additional advantage of enabling the original obstacles to have any shape. For simplicity, the symbols cnf and φ will be used for the ith configuration in a discrete sequence and the corresponding vector of angles respectively, while cnf and $\tilde{\varphi}$ will denote the next,

(i+1)th, configuration and its angle vector. For computing a continuous sequence of configurations permissible with respect to obstacles, it will suffice to compute a finite configuration sequence in which all the configurations are permissible with respect to ϵ-covers and each one is close enough to the next, i.e.

$$|\tilde{\varphi}^k - \varphi^k| \leq \delta^k \qquad (k=1,\ldots,4) \qquad (2)$$

The choice of δ^k depends on the value of ϵ. Now the manipulator motion reduces to a number of steps each consisting in computing an increment $d\varphi^k$ for the equation

$$\tilde{\varphi}^k = \varphi^k + d\varphi^k \qquad (k=1,\ldots,4) \qquad (3)$$

These increments enter into the linear version (1)

$$dB_i = \sum_{k=1}^{i-1} (\partial B_i / \partial \varphi^k) d\varphi^k \qquad (i=2,\ldots,5)$$

To simplify matters further, the path G for the jaws is assumed either to be a broken line with verticies $G_1, G_2, \ldots, G_N$ and segments $g_1, g_2, \ldots, g_N$ or to have been approximated with such a line. The jaws are permitted to move nearby this line in such a way that the distance between the jaws and the line should never exceed a tolerance constant ϵ_c. Thus, the basic task has been reduced to securing the motion of the jaws along each of the N corridors CR_i in succession.

V OVERVIEW OF THE METHOD

Now we outline the major ideas and techniques of the AVOID system, leaving details for Sections 6 and 7.

Supposing all δ^k have been computed, the current configuration enf is permissible, and the jaws J are inside the corridor CR_i, the problem reduces to computing increments $d\varphi^k$, k=1,...,4, which on substituting in (3) imply that

1) inequalities (2) hold;

2) the jaws in configuration $\tilde{cnf}$ are inside CR_i;

3) the resulting configuration $\tilde{cnf}$ is permissible;

4) the distance covered by the jaws is maximized.

Here the design constraints are temporarily omitted. Dropping condition 3 for a moment, the problem nay be viewed as a linear programming problem (LPP). Indeed, condition 4 can be transformed into the form

$$\min_{d\varphi} F(d\varphi)$$

where F is a scalar linear function (Sect.7), whilst condition 1 combined with 2 is just a linear constraint set (LCS). Supposing such an LPP could be easily solved (see Sect. 6), we get $d\varphi$ satisfying conditions 1,2,4. Computing first $\tilde{\varphi}$ from (3) and then $\tilde{R}_i$ from (1), we obtain cnf and can check it for permissibility. If $\tilde{cnf}$ is permissible, we replace enf with $\tilde{cnf}$ and proceed as before; otherwise, the LPP must be modified to render $\tilde{cnf}$ permissible. By definition $\tilde{cnf}$ is not permissible if it collides with at least one obstacle. Such a collision occurs, when either 1) joint B_i puts into edge a_j of obstacle 0; or 2) link T_i cuts off vertex A_j. Collision of type 1 takes place when joint B_i in its motion tries to run through edge a_j. To prevent this, we can forbid the joint B_i to puncture the edge a_j at the moment of collision, thereby forcing it to ride the edge a_j. Let the edge a_j have equation

$$a_j(r) = a_j^1 r^1 + a_j^2 r^2 + a_j^3 = 0$$

where r^1 and r^2 are Euclidean coordinates of a point r. Coefficients a_j^k are chosen in such a way as to insure positive values for $a_j(r)$ at exterior points of the polygon, $a_j(B_i) > 0$. Then to express the forbidding restriction, we require that the point $\tilde{B}_i = B_i + dB_i$ lie outside a_j:

$$a_j(B_i + dB) > 0 \qquad (5)$$

Using (4), equation (5) can be transformed into the form

$$a_{\varphi,j}(d\varphi) > 0 \qquad (6)$$

and appended to the LCS. Clearly, a solution to the thus modified LPP is permissible if the detected collision of B. with a. is the only one at the moment. Subsequent collisions of both types, if any, could be similarly dealt with one at a time.

The design constraints (which can be readily incorporated into the LCS) and the constraints resulting from conditions 1-2 above (put into the LCS earlier) are called permanent constraints, whilst the constraints (6) are temporal ones. In Sect. 6 it will be shown that as a result of the proper choice of a solving method, only minor modifications of a solution $d\varphi$ or the LPP are required to account for the additional constraints (6).

As long as the joint B. keeps riding the edge a., the temporal constraint (6) remains in the LCS. Sooner or later, however, either B. will reach an end of the edge a. and slide off, or it will take off from the edge. Either of these cases is accounted for by dropping the corresponding constraint, i.e. by deleting the inequality (6) from the LCS. Treatment of the take-off case requires no special care since it involves an immediate dropping of the constraint in question from the basis (see Sections 6 and 7). To account for sliding-off, a special checkup procedure has been set up in the AVOID system for signalling to the control structure the fact of B. being off edge a_j. In either case the constraint in question is eliminated, thereby cutting down on the number of constraints in the LCS. Type 2 constraints, if any, are treated similarly.

Summing up, it may be said that a major function of AVOID consists in generating and solving a scries of LPPs, each with a slightly different constraint set. Some of the constraints, the temporal ones, are inserted or deleted as a need arise, reflecting interaction of the manipulator with obstacles. Thus, the difficulty mentioned in Section 3 is overcome stepwise by converting the restrictions imposed by obstacles in the vicinity of the current point into state space restrictions where they can be easily dealt with. AVOID's ability to insert and delete these additional constraints at each step renders the system sufficiertly flexible.

VI THE DUAL METHOD IN LINEAR PROGRAMMING

Now we will present briefly a particular implementation format for the dual method in linear programming as being most suitable for the LPP under consideration. There are two major features to the LPP to be solved. The first is the existence of a subproblem of the given LPP, solution to to which is readily available. Subproblem here means another LPP with the same objective function and fewer constraints in the LCS. The second feature is the probable need for incorporating additional (temporal) constraints into the LCS after the solution has been obtained, which entails modifying the solution to account for the newcomers. Both features are easily accomodated in the dual method format described below.

Consider the following LPP: Find $x=(x1,\ldots,xn)$ which minimizes the linear objective function

$$\min_{x} F(x) \qquad (7)$$

and satisfies the constraints

$$x^k \geq 0 \qquad (k=1,\ldots,n) \qquad (8)$$

$$f_1(\underline{x}) \geq 0,\ldots,f_n(\underline{x}) \geq 0 \qquad (9)$$

$$f_{n+1}(\underline{x}) \geq 0,\ldots,f_s(\underline{x}) \geq 0, \qquad (10)$$

where all f. are linear functions. (7)-(9) constitute the 1subproblem. We proceed in two stages. First, the subproblem (7)-(9) is solved, i.e. a point x1 is found which minimizes (7) subject to the constraints (8)-(9). This is a fairly simple task since our case (see Section 7): (i) inequalities (8) hold; (ii) inequalities (9) which combine the design constraints and inequalities (2) have the form

$$y^k = c^k - x^k \geq 0 \qquad (k=1,\ldots,4) \qquad (11)$$

Here x^k are given by the formulas

$$x^k = d\varphi^k + h^k, \qquad (12)$$

for which constants h^k are easily computed. Thus, the n-dimensional subproblem factorizes into n one-dimensional ones (n=4). The solution x1 for this subproblem is a vertex of the n-dimensional parallelepiped determined by inequalities (8), (11). The constraints generating the solution x1. are called basic. The second stage, involving the entire problem (7)—(10), consists in checking one by one whether the solution x1 found thus far satisfies each of the remaining constraints (10). If it does, the process terminates, and x1 is the solution for the entire LPP; otherwise, the first violated constraint enters the basic constraint set at the expense of some other constraint which drops from the basis. The solution point shifts from x1 to some x2 with $F(x1) < F(x2)$. The process terminates when the entire list of constraints is run through without modifying the basis. It is seen now that a newly emerged constraint causes at most a slight modification in solution, which allows us to save on the bulk of initial computations.

VII STRUCTURE OF THE PLANNING ALGORITHM

In this Section the planning algorithm of the AVOID system is presented on a step-by-step basis. A simplified flow-chart for computing the motion of the jaws along a corridor CR1 from its back side to the front side F is shown in Fig.2. More details are given in [3],

PRINCIPAL MATRIX

		1	2	3	4	5
		1	x^1	x^2	x^3	x^4
1	$F(J+dJ)$	F^0	F^1	F^2	F^3	F^4
2	y^1	c^1	-1	0	0	0
3	y^2	c^2	0	-1	0	0
4	y^3	c^3	0	0	-1	0
5	y^4	c^4	0	0	0	-1
6	y^5	$p^0+\delta_c$	p^1	p^2	p^3	p^4
7	y^6	$-p^0+\delta_c$	$-p^1$	$-p^2$	$-p^3$	$-p^4$
8	y^7	u^0	u^1	0	0	0
9	y^8	v^0	v^1	v^2	v^3	0

Box 1. Here the Principal Matrix PM for the LPP is computed for the current configuration cnf. PM has 5 columns and (7+t) rows, where t is the number of temporal constraints in the LCS at the moment. Column 1 consists of constant terms of the corresponding inequalities; columns from 2 to 5 contain their coefficients in x . Row 1 contains coefficients F in x of the objective function F(J+dJ), where F(.)=0 is the equation of the front side of the corridor.. and nonnegative variables x are obtained from $d\varphi^k$ by linear substitutions (12). Rows from 2 to 7 contain permanent constraints, rows below 7 contain temporal constraints if any. Coefficients in rows 2 to 5 are those from inequalities (11), whilst coefficients in rows 6-7 are determined by the fact of the jaws J moving inside the corridor, i.e., by the formula

$$g(J+dJ) \leq \delta_c ,$$

or

$$p^0 + \delta_c + \sum_{k=1}^{k=4} p^k \, d\varphi^k \geq 0$$

$$-p^0 + \delta_c - \sum_{k=1}^{k=4} p^k \, d\varphi^k \geq 0$$

Rows 8 and 9 feature coefficients for exemplary constraints for collisions of type I (for joint $B_?$ involved in the collision) and of type 2 (for link l_g) respectively. The derivation of formulas for v^k and v is straightforward.

Box 2. The initial basis is chosen by means of the Jordan Elimination Procedure applied to the elements (2,2), (3,3), (4,4), (5,5) of PM in turn. The number of eliminations required for solving this subproblem (cf. Section 6) is 0 to 4.

Box 3. Now a solution for the entire LPP is obtained. The resulting values of x are to be used in formulas (12) for computing $d\varphi$. The corresponding value of the objective function F occupies the space (1,1) in PM.

Box 4. The new configuration $\tilde{cnf}$, i.e. angles $\tilde{\varphi}^k$ and positions of joints $\tilde{B}_i$ are computed by formulas (3) and (1).

Boxes 5 and 6. Having obtained $\tilde{cnf}$, it is checked for collisions with the obstacles. If no collision is detected, control is transferred to Box 7; otherwise, a proper temporal constraint is appended to the LCS (Box 6), and the process of solving the new LPP resumes (Box 3; see also Section 6).

Box 7. After colliding with an obstacle, the joint keeps riding it, and the corresponding temporal constraint remains in the LCS for a number of steps. But after a while the joint either slides off the edge or takes off. To account for either event without detecting relevant intersection afresh at each step, the inheritable basis technique is introduced. If a constraint enters the basis, it is likely to remain there for the time being. Therefore, after computing $\tilde{cnf}$, all the necessary information for the collision (i.e., type of the collision, number i for B_i, etc.) is retained for future use. Entering this most-likely to-be-encountered constraint into the LCS at the next step prior to the main body of calculations (Box 1) saves the system a lot of trouble. Sooner or later, however, any temporal constraint will dr p from the basis and therewith from the LCS.

Box 8. This stage of computation involves outputting configuration $\tilde{cnf}$ to storage as the latest one in the sequence and installing it as the current configuration cnf.

Boxes 9 and 10. Clear without comments.

VIII SIMULATION RESULTS

To illustrate the performance of the computer-implemented version of the AVOID system, some of the experiments run with it are displayed below. The chosen manipulator model has four links, each of length 1. Bold lines are used for obstacles and configurations while ϵ-covers (ϵ=0.1) and corridors for the jaws are shown with thin ones.

TASK 1 (Fig.3). The coordinates of the three vertices of the only obstacle available are A -(0.120, -0.120), A -(1.620, -1.620), A =(].620, -0.120) whilst those of the ϵ-cover are (-0.121, -0.020), (1.720, -1.861), (1.720,-0.020). The broken line for the jaws[1] motion consists of a single segment with initial point G = (2.975,-1 .000) and final point G2=(2.975, -1.810). It requires 7 steps of the system to solve the problem. The initial (no.l) and the final (No. 8) configurations are shown in Fig. 3. In the process of planning the second step the joint B2 strikes on the edge 3 of the cover, which leads to the insertion of a temporal constraint of type 1. During the planning of the fourth step, the link 1 2 cuts off the. vertex 3 while the joint B2 takes off the edge 3, which results in dropping the first temporal constraint. The second temporal constraint, remains in the basis until the end. The distance covered by the jaws for the sequence of 7 steps is 0.763, the mean step size being 0.109.

TASK 2 (Fig.4). The task site here features two obstacles 01 and 02 and the broken line, of 4 segments for the jaws. This rather difficult task takes 36 steps to accomplish. After a number of interactions of the links with both obstacles, the jaws happily reached their destination point G .

TASK 3 (Fig.5) has been solved in 38 steps.

IX CONCLUSIONS

Though the problem of planning manipulator motion in between obstacles has been taken up in a number of papers (see References) no comparative study of different methods for solving this problem has been attempted. Various facets of our approach may be found in [l] - [4] . Advantages of the AVOID system are. as follows.

1. The method is very simple. All the constraints imposed on the manipulator, including those implied by obstacles, are imbedded into a set of linear inequalities and treated uniformly by the dual method, neatly fitting the problem format. The amount of computation involved is fairly moderate.

2. Having encountered an obstacle in its way, the system does not attempt any precarious movements in space trying to get clear of it. Instead, the trouble-maker is incorporated into the system for the time being, i.e. as long as it really matters. The moment the obstacle in question does not hinder the manipulator any more, it is dismissed for good. This device provides considerable flexibility for the system.

3. The concept of inheritable basis employed throughout the planning process helps further to reduce the amount of computation.

The described method may be extended to 3-dimensional tasks, and to curves and obstacles more sophisticated than those used in the actual work done. It could be integrated into a completely autonomous robot system as an intermediate level of planning manipulator motions. Also, it could be used in a man-machine system for computing a path between points in space specified by a human operator.

As currently implemented, the AVOID system has no facilities for computing velocities and accelerations. The task of modifying the system to enable it to compute dynamic characteristics will need further study.

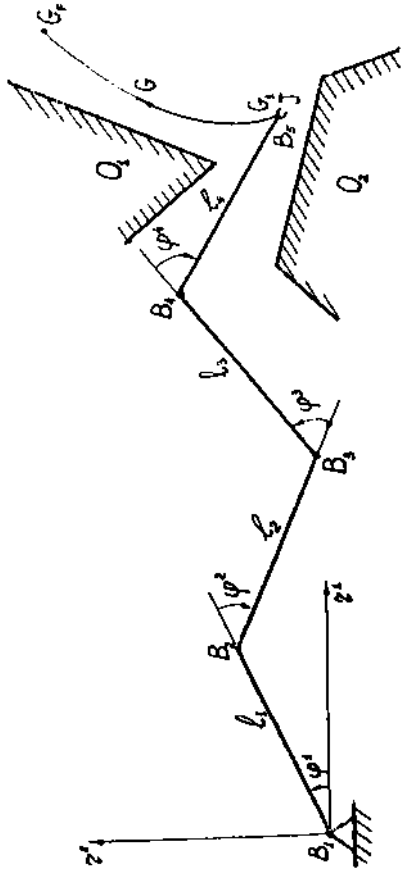

Fig. 1

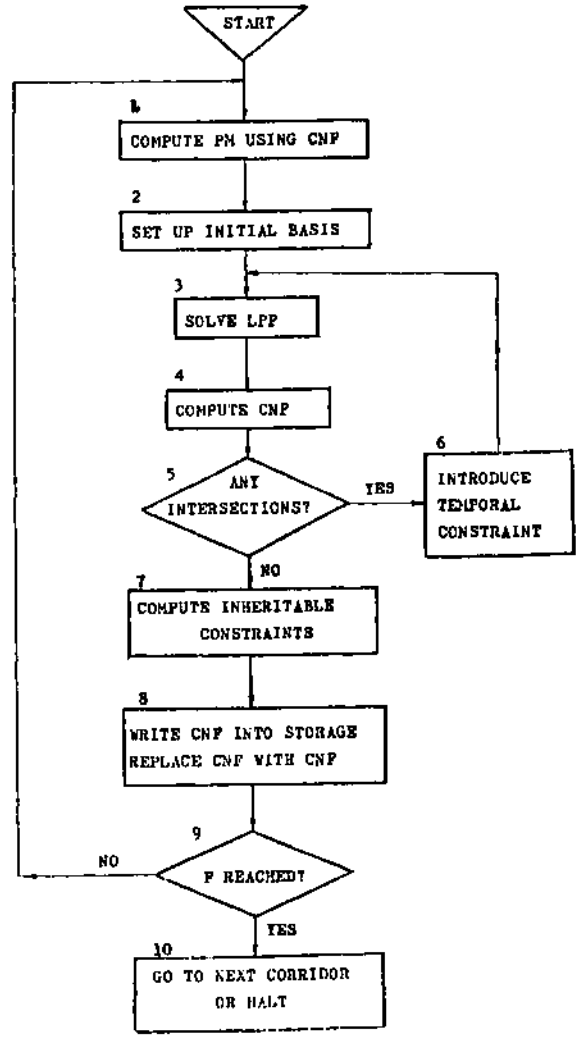

Fig.2 Flow-chart for the algorithm

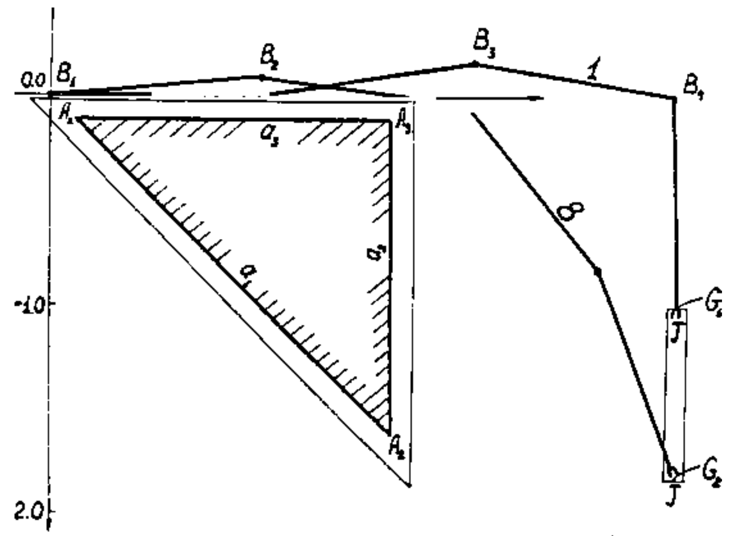
Fig. 3

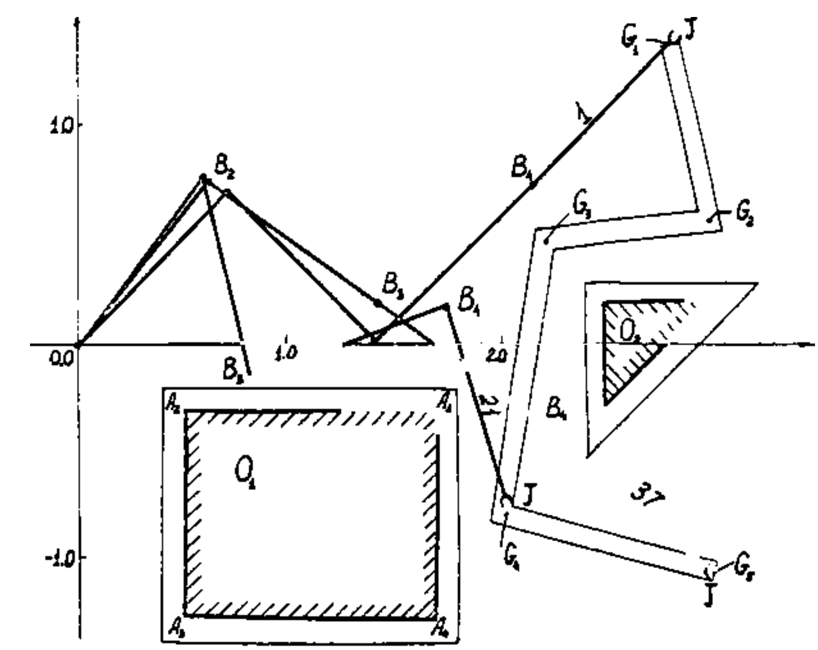
Fig. 4

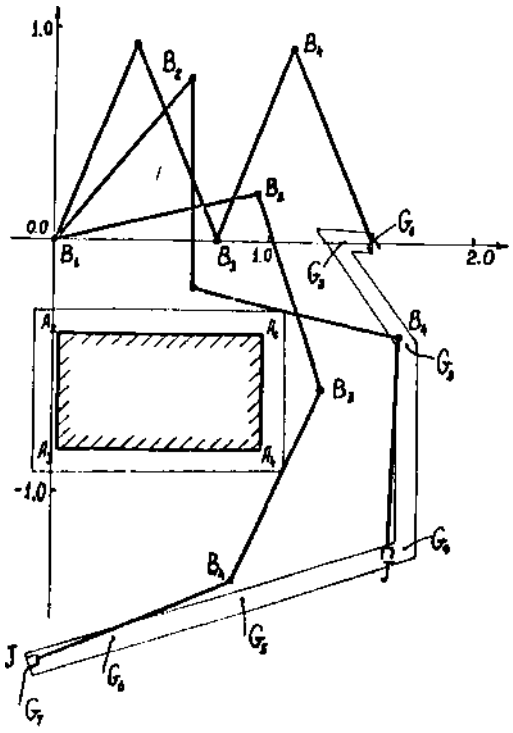

Fig. 5

REFERENCES

1 Grechanovsky, E. "Manipulation Systems and Motion Planning" (in Russian). Mashinovedenie, no. 4, pp.5-11, 1975.

2 Grechanovsky, E. "On Motion Planning in Manipulation Systems" (in Russian). In: Modelirovanie Zadach Mashinovedeniya na EVM. "Nauka" Publ., Moscow, pp.50-58, 1976.

3 Grechanovsky, E., Pinsker, I.Sh. "A Method for Planning Manipulator Motion in the Presence of Obstacles" (in Russian). In: Pinsker, I.Sh. (Ed.), Modeli. Algoritmy. Prinyatie Resheniy. "Nauka" Publ., Moscow, pp.100-142, 1979.

4 Grechanovsky, E., Pinsker, I.Sh. "Controlling Manipulator Amidst the Obstacles" (in Russian). Mechanika Mashin, no.55,pp.67-72, 1979.

5 Kobrinski, A.A., Kobrinski, A.E. "Constructing Motions for a Manipulation System Amidst the Obstacles" (in Russian). Doklady Akademii Nauk, vol.224, no.6, pp.1279-1282, 1975.

6 Kobrinski, A.A., Kobrinski, A.E. "Constructing Optimal Motions for Manipulation Systems" (in Russian). Mashinovedenie, no.I, pp.12-18 1976.

7 Loeff, L.A,, Soni, A.H. "An Algorithm for Computer Guidance of a Manipulator in Between Obstacles" Transactions of the ASME, Journal of Engineering for Industry, Ser.B, vol.97, no.3, pp.836-842, 1975.

8 Malyshev, V.A. "Representing External Environment, Planning, and Constructing Programmed Manipulator Motions" (in Russian). Izvestiya Akademii Nauk, Tekhnicheskaya Kibernetika, No.3, pp.53-55, 1981

9 Malyshev, V.A., Timofeev, A.V. "An Algorithm for Constructing Programmed Manipulator Movements Allowing for Design Constraints and Obstacles (in Russian). Izvestiya Akademii Nauk, Technicheskaya Kibernetika, no.6, pp. 64-72, 1978.

10 Pavlov, V.A., Timofeev, A.V. "Constructing and Stablizing Programmed Motions for a Movable Robot-Manipulator" (in Russian). Izvestiya Akademii Nauk, Tekhnicheskaya Kibernetika, no.6, pp.91-101, 1976).

11 Piper, D.L. "The Kinematics of Manipulators Under Computer Control. Stanford Artificial Intelligence Project, Memo AI-72, Stanford University, Stanford, 1968.