A VERSATILE COMPUTER-CONTROLLED ASSEMBLY SYSTEM

A.P. Ambler
H.G, Barrow
CM. Brown
R.H. Burstall
R.J. Popplestone

Department of Machine Intelligence
University of Edinburgh

Abstract

A versatile assembly system, using TV cameras and oomputer-controlled arm and moving table, is described. It makes almple assemblies such aa a peg and rings and a toy car. It separates parts from a heap, recognising them with an overhead camera, then assembles them by feel. It can be instructed to perform a new task with different parte by spending an hour showing it the parts and a day or two programming the assembly manipulations. A hierarchical description of parts, views, outlines etc. is used to construct models, and a structure matching algorithm is used in recognition.

1, Introduction

A computer-controlled versatile assembly system has been programmed during the past 12 months using, the Edinburgh hand-eye hardware (Barrow aid Crawford) The equipment (Fig. 1) consists of a moveable table, a mechanical hand with sensors and rotating palms, and two TV cameras, all conneoted via an 8X Honeywell 316 to a 128K time-shared ICL 4130 running POP-2 programs. Several other programs are running on thia equipment, including a program for recognising irregular objects and one which packs arbitrarily shaped objects into a box (MIchie et al). The program described here is our most ambitious effort. It is capable of assembling a variety of structures, and much of our effort has been spent in enabling the machine to acquire descriptions of the parts for itself using an overhead TV camera.

Related work has been carried out at Hitachi (EJiri et al), at MIT (Winston[45]) and at Stanford University (Feldman). The Hitachi program could build a variety of simple structures of blocks from line drawings of the structure, the HIT programs can learn concepts about structures and copy an arbitrary structure of simple blocks given spare parts, and a recent Stanford program can assemble a simple automobile water pump using preprogrammed hand manipulations.

2. The TAsk

A number of parta are placed by the operator in a heap on the table (Fig. 2, peg and rings). The machine's task is to separate the parts and recognise them, then to assemble them into some predetermined configuration (Fig. 3). Figs. 4 and 5 show another example, a toy oar. We are currently thinking in terms of up to a dozen parts with outlines described by up to twenty or so straight or ourved segments from any one view, possibly with some holes of similar complexity.

In order to explore the capabilities of a computer-controlled system aa opposed to a conventional electromechanical device we seek a versatile assembly system. The demand for versatility is also calculated to raise interesting aspects from an Artificial Intelligence point of view.

Our goal has been to develop a system which enables one to:-

(i) think up a new aaaembly involving a kit of parta which have not been used before,

(ii) spend a day *or so* familiarising the machine with the appearance and manipulation of the parts and instructing it how to assemble them into the required structure,

(iii) leave the machine unattended, busily making the structures ad nauseam, provided that a fresh heap of parte is dumped on the table from time to time.

We have achieved this goal for simple structures. We can show the machine half a *dozen* new parts and instruct it how to lay them out ready for assembly in about two hours; interactively programming the assembly operations themselves takes four hours or so. The machine can make the structures like the peg and rings, the toy car or a toy ship unassisted,but slowly, taking an hour or two to find and assemble the parts. It completes the assembly correctly about four times out of five. It should be possible to ease the programming of the assembly operations somewhat, but removing the need to program movements in terms of numerical co-ordinates would need a new approach. The system is about 50K of POP-2 code.

3. Hbw The Syatem Performs the Task

The system transforms a heap of parts into *a* completed assembly, typically going through the following step:-

Layout: Identify parts visually and put them in standard positions.

1. Locate all parts or heaps of parts on the table using the side camera (wide angle).

2. Inspect each part or heap with the overhead camera. If it is recognisable as an isolated part pick it up and lay It out in a standard position and orientation.

3. If there are no parts or heaps left, or if all the required parts have been found go to step 5.

4. Using the overhead oamera look for a protrusion in the smallest heap, grasp the protruding part or parts and separate them from the heap and go to step 1. If it has no protrusions try to break up the heap by picking it up as a whole and turning it over so that it falls apart. If this does not work, try pushing it with the hand at various heights. If stil no success, try another heap. Go to step 1.

5. If some parts are missing, complain. If there are more parte than are needed, clear them away.

Assembly:

Pick up each part from its standard position and insert it into the assembly, by feel. A workbench with a simple vice and various working surfaces is used.

The program is written as two quite distinct subprograms, layout and assembly. At present they do not communicate; they both know the standard

positions and orientations of the parts. The layout subprogram uses descriptions of the parts, acquired during the instruction phase, to recognise them, pick them up, turn them over if necessary and put them into standard positions. The assembly subprogram works blind, using no internal descriptions of the parts. It is written interactively during the instruction phase.

The system carries cut the steps listed above at 'execution time'. It is able to do so because the following steps have been carried out previously at 'instruction time'.

A. Instructing the layout program

For each part in each stable state (e.g. right way up, or on its side).

1. The operator places the part on the table under the vertical camera, the machine takes a picture and creates from it an internal deacription of that view of the part. This is repeated several times, and the machine adjusts its descriptions each time, taking note of the variations caused by reorientation of the part and imperfect picture information.

2, The programmer types in on-line commands to move the hand, to pick up the object, turn it over, put it down and pick it up again if necessary, then to put it down in the standard position and orientation, (if the assembly has several identical parts separate commands are given for putting each one down in its own place.) The programmer intersperses these commands with instructions to remember the current state, e.g. when the hand has closed over the part. The system takes a note of these specified states and at execute time constructs a sequence of actions to put the part into its standard position and orientation,

B. ....Writing the assembly program

For each part

1. The programmer puts the part on the table in its standard position and orientation, and he interactively devises and edits some POP-2 program to make the machine pick up the part and fit it into the assembly. His program uses basic move and grasp operations, and two high level manipulation operations provided for constrained moves and hole fitting.

4. The Layout Subprogram; Descriptions

We must first explain the kind of internal descriptions of parts used in the layout program. We can then show how it creates these descriptions at instruction time and how it uses them to recognise partB at execution time. For clarity we have simplified a few programming details, glossing over some unnecessary or uninteresting distinctions.

The program works in terms of a hierarchy of concepts called entities, each represented by program data structures, having other entities as its components. The entities used are summarised in Table 1 and Table 2 gives a brief description of each.

TAble

Hierarchical structure of entitles

An entity is either a table top, or an object-set, or an object, or a part, or a hand, or a workbench, or a heap, or a stable state, or a view, or a region, or a hole-set, or a hole, or an outline, or a segment.
A table top has an object-set.
An object-set has objects.
An object is either a part, or a hand, or a workbench, or a heap.
A part has stable states.
A stable state has a view.
A view has a region.
A region has an outline and a hole-set.
A hole-set has holes.
A hole has an outline.
An outline has segments.

Table 2

The entities used

The Table top is the whole collection of things on the table.
An Object is any physical thing on the table which can be seen or touched; it is initially distinguished from its surroundings by clear space on the table.
A Part is one of the separate pieces needed for the assembly e.g. one of the wheels of the car.
A Stable-state is one of the states in which a part can rest on the table, irrespective of orientation or position e.g. on its side, upside down. There should be only a small number of distinguishable such states.
A View is an analysed TV picture.
A Region is a connected light area in a picture, possibly with darker holes (we use light objects on a dark background).
A Hole is a dark area inside a region.
An Outline is the outer boundary of a region or hole.
A Segment is a segment of a circle (up to 360) with specific length and curvature (zero curvature means a straight segment). The irregular boundary of a region or hole is analysed into a small number of segments by curve fitting (Fig. 6).

Bach entity either has an n-tuple of components, or it has a aet of components. The size of the n-tuple is fixed as in the above 'syntax', for example a region has a pair of components; but the size of the set is not fixed until instruct time, for example the system discovers that the hole-set of a car body side view has two holes.

An entity may possess properties and some relations (at present only binary ones) may subsist between its components. The properties and relations have names and may be truth-valued or take values in some other domain such as numbers.

We make an important distinction between two kinds of entities: model and individual. Bach has entities of its own kind as components. An individual entity is an internal description generated by a particular exposure to a physical objeot using information from TV camera and hand aensors. Thus when the operator puts a car body on the table, the machine takes a picture, turns it over and takes another picture, one individual entity of type 'part' is generated and two individual entitiea of type 'view' are generated, together with individual outlines, holes, segments etc. A model entity, on the other hand, is a summary or composite of a number of such experiences. The end result of the instruction phase is a collection of model entities incorporating the system's knowledge about the parts, their views, outlines etc.; the individual parts, view and out-

lines which gave rise to these will have been discarded.

The important operation in recognition is the creation, from visual and tactile sense data, of an individual entity which matches a model entity and whose components match the components of the model entity. The individual entity contains a pointer to its model and one to its sense data, thus binding them together; it also contains certain specific information, e.g. position and orientation, not appropriate to model entities.

There are a few exceptions to the above. The system does not create models for its hand or the workbench at instruction time; these are given beforehand. There is no model for a heap since an individual heap is generated by elimination, on failure to recognise a part, hand or workbench.

The system has a data structure for each model entity and individual entity; these are POP-2 records linked by pointers to their components into tree structures. There is also a data structure for each entity class, for example the class 'view' and the class 'region'. Each model or individual entity belongs to some class and certain data pertains to the class as a whole - for example a list of the properties which an *entity of that class* enjoys and functions for computing their values.

To build entities the machine needs raw material which we shall call 'sense data1, information from the TV camera or possibly the hand sensors which has not yet been recognised as referring to any known entity. The recognition process, which we describe below, takes a model entity and some sense data and tries to create an individual entity which corresponds to the model.

Table 3 summarises the four notions of entity class, model entity, individual entity and sense data showing what information is associated with each. The use of this information will be clearer when we discuss the recognition process.

The matching process which recognises parts

To understand the recognition process let us consider what happens when the system has taken a TV picture and tries to interpret it as a side view *of* a car body. This may be during the instructionphass when it has been told that it is looking at the side view of a car body, or during the execution phase after it had found an upright car body and turned it over. Or again it might be dealing with an unknown object, and •sideview of car body' might be just one possible interpretation among several which it was trying.

The TV picture, a 2-dimensional array of brightness levels, constitutes a sense datum, d in D, A matching function is now applied to this sense datum and the model of the side view of the car body. This function produces a set of individual side views of car bodies, an empty set if there is no way of interpreting the picture as such a view, otherwise one element for each possible interpretation. Thus

matchi Sense data *x* Models -> Set *of* individuals

The matching function works its way recursively down the hierarchy from top to bottom, comparing gross properties on the way down and failing if they are too discrepant. Thus it might fail because the area of the region it is looking at in the picture is too small for a car body, without bothering to analyse the outline of the region. As it goes down it refines the sense data, using a thresholding region finder routine to set region level sense data from the view level brightness array, finding holes with the same routine to get hole data and fitting

Table 3

Notion and symbol	Associated descriptive information
Entity classes, C	Property names P, Relation names R, Property finding functions $f_p: D \to V$ for $p \in P$ (V is the set of values for properties and relations), Relation finding functions $f_r: D^c \to V$ for $r \in R$, Classes of components, Component finding functions in D->D, matching function in $D \times M \to 2^1$.
Model entities, M	Class, values of properties and relations, components.
Individual entities, I	Class, values of properties and relations, components, parameters, pointers to model and to sense data.
Sense data, D	At view level: 2-D brightness array from TV picture. At region level: region perimeter as list of vector increments, 2-D Boolean array showing whether point is inside and brightness array. At hole level: as for region level referring to dark holes At segment level: segment length, curvature and position.

curves to the perimeters to get segment data. (We have called the region finder and curve fitter 'component finding functions')* When it gets to the bottom level the recursion unwinds and passes up the hierarchy descriptions of individual entities, using their finer properties and relations between them to establish correspondence with the model. At each level the matching function produces a set of individual entities each of which might correspond to the model, thus dealing with ambiguity essentially as would a 'back-track' or nondeteministic process.

To be more precise, the function 'match' works as followst-

```
Function mateh(d,m)
    Let c be the entity class of m.
    Let f be the special matching function of class
                                                    c.
    result = f(d,m)
```

The special matching function f may vary *from* class to class, but normally f is 'general-match', defined as follows:-

general-match : Sense data x Models -> Sets of individua;s

```
Function general-mateh(d,m)
  Let c be the entity class of m.
  Let F be the set of properties for class c.
  For each p in P, oompare f (d), the value of
    property d for the senaePdatum, with the value of
    p in the model m.  If there is too much dis-
    crepancy exit with result = empty set.
  Case 1.  m has an n-tuple of components, m1...,m. .
    Apply the component finding functions of the
    class c to d, to find sense data relevant to the
    components say d1.....,dn . As each d. is com-
    puted, match it against the oomponentJm,, thus
    let li=match(d.fm,).  If some l. is empty then
    exit Immediately with result "empty set.
    Otherwise use each element of. /  (the
    Cartesian product of the setsJof Individual
    components) to construct a new individual with
    these components.  The result is the set of
```

these individuals.

Case 2. m has a set of components, S .
Apply the component finding functions for class c to find a set of sense data relevant to the components, S.. Use the relation value finding functions of e, f for r in R, to compute the values of relations between the Sd. Compare these with the known values of the relations for the S , and use the relational structure matching algorithm (described below), together with the function match, to find the largest subsets of 3. which correspond. If these subsets are sufficiently large construct a new individual entity from eaoh correspondence among the components. The result is the set of individuals so constructed.

5 Matching Relational Structures

A set of segments forming the outline of a part can be regarded as a relational structure endowed with properties, such as length and curvature, and relations, such as adjacency, distance or relative orientation, similarly for holes or objects on the table top. Although the properties and relations usually take numerical values (length) it will simplify the discussion to talk in terms of truth valued ones (long, medium, short). In the algorithm given in the last section there is a point (case 2) where we need to put two sets (of segments, say) into correspondence on the basis of these properties and relations. TV picture processing being what it is we expect discrepancies (segments missing, two segments coalesced) but we want to match as many elements as possible. Pig. 7 shows two simple outlines and corresponding relational structures; they have several common substructures, e.g. {11',32', 54*, 43').

More precisely, by a relational structure we mean a set S of elements together with a set of properties P and a set of relations R over it (we consider only binary relations here). Given two relational structures <S,,P,R> and <S2,P,R> we define a match between them as a set T1 C s1 a set T2C S2 and an isomorphism, ~ , between T1 and T2 preserving properties and relations. Thus s1 = s2 implies p(s1) iff p(s2) for each p in P, also S1=S2 and ■' s1 imply r(S1,S1') iff r(s2,s2') for each r in R, and a match represents a common substructure in our two relational structures.

We can find matches as follows. By an assignment we mean a pair <s1,s2> with s1 in Sj and s2 in S2 such that p(s1) iff p(s2) for each p in P. In Fig. 7 the assignments are 11', 12', 14', 23', 31', 32', 34'.etc. We say two assignments <s1S2> and <s1',s'> are compatible if r(s1,s') iff r(s„,s|) for all r in R, New by definition a match is just a set of assignments such that each assignment is compatible with every other assignment in the set. Indeed we may think of the assignments as forming the nodes of a graph with compatibility as the (symmetric) relation forming the arcs. Our problem then is to find totally oonnected subsets of this graph, often called cliques.

A clique is said to be maximal if no other clique properly includes it. Finding maximal cliques is a well known problem (Karp7) A graph of n assignments may have *(n/2)* maximal oliques in a theoretical bad case, but at least we can find each maximal clique in time proportional to n (or n log n if we push it). We can do this by using a refinement of a simple binary search algorithm given by Burstall8. KnSdel? gives a similar algorithm. In fact for our recognition problem it seems adequate to generate only largest cliques (those with maximal number of elements) rather than maximal ones. Fig. 8 shows the compatibility graph for our previous example and the three maximal oliques indicated by △, □ and O. (Remember nodes in this graph represent corresponding pairs of elements of the two structures and connecting arcs represent compatibility between such pairs.) In practice we do not seem to get too many largest cliques when dealing with relational structures of a dozen or so elements, and the algorithm takes only a few seconds or so to generate them.

Our clique finding algorithm builds up cliques a node at a time. It is a recursive function, cliques: $2^{Nodes} \times 2^{Nodes} \to 2^{2^{Nodes}}$ such that cliques(X,Y) is the set of all cliques which include a clique X and are included in Y. Thus cliques(∅,Nodes) is the set of all cliques in the graph. It is defined most simply thus

cliques(X,Y) <= if no node in Y-X is connected to all elements of X then {X}
else cliques(X ∪ {y},Y) ∪ cliques(X,Y-{y})
where y is such a node.

(The reasoning is that if some y is eligible for addition to the clique X then each clique including X must either contain y or exclude y.) This generates a search tree whose nodes are labelled with a pair: set of nodes chosen, set of nodes available for choice. It starts with the empty set and all the nodes and each branch consists of including or excluding a node.

If we are looking for maximal cliques only we can cut down the search by computing at each state the set $\bar{Y}$ where $\bar{Y}$={z in Nodes: z is connected to each node in Y} and noting that any maximal clique includes $\bar{Y}$, also noting that if $\bar{Y} \not\subseteq Y$ then there are no maximal cliques included in Y. In fact our program seeks only largest cliques and runs a version of the function cliques of size k, stopping the recursion if the size of X plus the number of nodes in X-Y connected to all of X becomes less than k. We count down on k until some cliques are found.

In this section we have defined an assignment as a pair of elements, one from each structure, which have identical property values. More loosely one can demand only some degree of similarity in the properties. Our program uses as the assignments the set of all pairs <m',i'> such that i' is in match(d',m') for some m' in S_m and some d' in S_d. This gives us the nodes in our graph; the arcs are found by defining compatibility to be some suitable degree of similarity in the relations.

This relational structure matching is used three times by the layout program: to match segments in an outline, to match holes in a region and to match objects on a table top, e.g. associate some objects which it now sees with its previous knowledge. It gives us a robust matching technique which can make correspondences between observed and predicted elements in spite of imperfect data. It has some similarity in its way of working to Waltz's technique for labelling pictures (Waltz[10]). Winston[11] also does structure matching. Graph isomorphism is a well-known problem (e.g. Corneil and Gottlieb[12]; relational structure isomorphism is essentially the same. Testing whether one structure is a substructure of another is computationally harder (Barrow, Ambler and Burstall[13]) and our problem here, finding common substructures is still more onerous, but it is needed if both structures have imperfections.

Separating parts from a heap

The process of removing parts from a heap is heuristic, that is it needs a little luck. But it has never failed, if the machine is left to worry at a

heap long enough. Our program does not detect internal lines, relying on the outline of the heap. It attacks the smallest heap first.

The first tactic used is to look for a protrusion in the heap outline with a 'neck' which might indicate a part which is easily separable. A number of possible hand positions are then considered so as to pick up the protruding part or parts without fouling the rest of the heap. If it succeeds in picking something up the machine examines both this and the rest of the heap, trying to recognise an isolated part. The second, cruder, tactic is to try and pick up the whole heap, then rotate the palms to let pieces fall off. The third is to push the heap with the hand at various heights. Clearly there are subtler ways of picking objects out of a heap, but for the time being our simple tactics suffice.

Overall control aspects of the layout program

The control structure of the layout program is quite simple. At the top level there is a simple loop. We may transcribe the POP-2 code thus

```
loop: if all-tidy then goto exit;
      if do-search then explore (work-area)
      else if parts-untidy then put-all-away
      else if extra-objects then discard-objects
      else if lost-parts then search-for-missing-
                                              parts
      else if parts-needed then smash-a-heap
      else if extra-heaps then put-heaps-in-
                                  corner; goto loop
exit:
```

This snables it to cope with too many or too few parts. If one *of* the lower level routines enoounters difficulties it simply jumps out to loop. At a lower level various routines on entry assign a label to the variable pick-up-fail, and failure to pick something up satisfactorily causes a jump out to the current label.

Our recognition process is basically recursive (we have experimented with some process-swapping techniques but they are not in our program at present). This does not lead to too much rigidity, because we make rather extensive use of 'memo-functions' (Michie[14]) for example, when an analysis of the outline of a *region* is required the analysing function remembers the result and, when asked for it again, simply returns it immediately. Thus outline analysis is only done when needed and never repeated.

6. The Assembly Subprogram

This program works blind, using only hand sensing. It is written interactively at instruction time in terms of basic hand moving and sensing operations together with two higher level operations.

The basic operations include 'raise *z* centime-tea', 'move to (x,y)', 'grasp to w centimetres', 'rotate palm by O ', and functions for reading forces on the hand by means of strain guages, namely gripping force, weight of the object held and torque. There are two higher level operational constrained move and hole fitting.

Constrained move. This operation has two parameters, both force vectors, f a force opposing movement and f a constraining force, (Fig. 9). Let u and u be Tne unit vectors in these two directions. Let 6 and £ be small scalar distances. The hand attempts to move in direction -u until it is opposed by a force larger than f . ~2t the same time it keeps in contact with a surface which offers a resisting force f . The resulting movement will not necessarily be in direction -u but along the surface according to the component of $-u_m$ tangential to the surface. The operation works in detail as follows:-

1. If a force greater than f_m is felt then stop.
2. Move by $-\varepsilon_m u_m$.
3. If a force greater than f_c is felt then move by $\varepsilon_c u_c$ and goto 3, otherwise goto 1.

Fig. 9 shows an example of the pattern of movement which a single call on the constrained move operation might produce. The constraining force parameter may be left undefined giving an unconstrained move in direction -u until an opposing force is felt. This operation was suggested to us by work at HIT Draper Laboratory (Kevins et al[15]).

Hole fitting. This operation has two force vector parameters f_c and f_s. Let u_c and u_s be the corresponding unit vectors and ε_c and ε_s small scalar distances. It is used when the hand is holding an object with a protrusion which must be fitted into a hole (e.g. axle into car body) or is holding an object with a hole which is to be fitted over some protrusion (e.g. putting a ring on the peg). The hand moves in a spiral pattern normal to u_c, pushing repeatedly in the direction u_c until an opposing force greater than f_c is felt and then retreating and making a short move in the direction of the spiral before trying again. When, at some point in the spiral it succeeds in moving forward by $\varepsilon_c u_c$, it tests whether it has found the hole by moving sideways by $-\varepsilon_s u_s$. If a force greater than f_s is encountered (or after five attempts) the process stops, otherwise the spiral pattern is resumed.

The wooden car assembly gives an idea of how the program proceeds. A 'workbench' is used, fixed to one corner of the table (Fig. 10). It has a 'vice' for holding a wheel while an axle is being inserted, consisting of an L-shaped corner piece and a pivoted bar which the hand closes so that the wheel is held between the bar and the L. It also has a vertical 'wall' so that the car body can be held firmly while the second wheel is pushed onto each axle.

The sequence of events, in outline is:-

(i) *The* hand puts a wheel in the vice and inserts an axle.

(ii) It turns the car body upside down, picks up the axle with the wheel on it and inserts it into the body.

(iii) Repeat (i) and (ii) for the second wheel and axle.

(iv) Put the car body against the wall upside down with the two wheels against the wall.

(v) Push the remaining two wheels onto the protruding axles.

(vi) Pick up the assembled oar and place it on the table.

Assembly programming is still quite tedious, involving choice of numerical parameter* for distances and forces, and we have some ideas for easing it, Popplestone[16], There is clearly a lot of thinking to be done before we could make the assembly phase as versatile and easily instrucable as the layout, e.g. by replacing numerical commands with instructions using relations like 'on top of and 'fitting into' or by showing the machine intermediate assemblies. In particular our present assembly subprogram does not use the internal descriptions of the parts which have been acquired during instruction by the layout program. Such descriptions would have to be recast so as to be useful for assembly as well as recognition.

7, Concluding Remarks

Writing this program has been a valuable exercise

from the point of view of understanding- what problems are important for an integrated assembly system. By tackling a definite task but imposing the requirement of versatility, We hare raised some interesting Artificial Intelligence questions without writing a program just to justify A.I. dogmas. We hope to work in future on (i) making the learning system more coherent (il) extending the vision system to deal better with side views (essentially requiring 3-dimensional teohnigues) and to analyse half-completedl assemblies, and (iii) making it easier to program assembly manipulations, being less specific about nuaerical co-ordinates and the magnitudes of forces.

Acknowledgements

Donald Michie initiated our work on robotics and directed the' efforts to produce a working hand-eye system. Steve Salter and Gregan Crawford made the main contributions to the hardware. Ban Bobrow made helpful suggestions about the vision and recognition side, also about this paper. Ken Turner contributed the outline segmentation routines to our vision work. Valuable ideas about assembly came from MIT Draper Labs. We are grateful to Eleanor Kerse for typing. The project had support from the Science Research Council and from the General Post Office.

References

(1) Barrow, H.G. and Crawford, C.F. The Mark 1.5 Edinburgh Robot Facility. Machine Intelligence 7 (eds. B. Meltzer and D. Michie) Edinburgh: Edinburgh University Press, pp. 465-480. (1972).

(2) Michie, D., Ambler, A.P., Barrow, H.G., Burstall, R.M., Popplestone, R.J. and Turner, K. Vision and Manipulation as a Programming Problem. Proceedings of the 1st Conference on Industrial Robot Technology. Nottingham. (1937)

(3) Ejiri, M., Uno, T., Yoda, H., Goto, T. and Takeyasu, K. An intelligent robot with cognition and decision-making ability. Proceedings of Second International Joint Conference on Artificial Intelligence. Imperial College, London, pp. 350-358. (1971).

(4) Winston, P.H. Wandering About on the Top of the Robot. Vision Flash 15. Artificial Intelligence Laboratory, MIT, Cambridge, Massachusetts. (1971).

(5) Winston, P.H. The MIT Robot. Machine Intelligence 7 (eds. B. Meltzer and D. Michie) Edinburgh: Edinburgh University Press, pp. 431-463. (1972).

(6) Feldman, J.A. Private Communication. (1973).

(7) Karp, R.M. Redueibility Among Combinatorial Problems, Complexity of Computer Computations (ed. K.E. Miller and J.W, Thatcher) New York: Plenum Press, pp. 85-103. (1972).

(8) Burstall, R.M. Tree-searching methods with an application to a network design problem. Machine Intelligence 1 (eds. N.L. Collins and D. Michie) Edinburgh: Oliver and Boyd, pp. 65-85. (1967).

(9) Khodel, W. Bestimmung aller maximalen vollstandigen Teilgraphen eines Graphen G nach Stoffers. Computing. Vol. 3. Ho. 3, pp. 239-240. (1968). Correction in fbi,d. 4, p. 75.

(10) Walts", D.L. Generating Semantic Descriptions frr Drawings of Scenes with Shadows. Artificial Intelligence Laboratories Report AI TR-271. MIT, Cambridge, Mass. (1972).

(il) Winston, P.H. Learning Structural Descriptions from Examples. Artificial Intelligence Technioal Report 231, Artificial Intelligence Laboratory, MIT, Cambridge, Mass. (1970).

(12) Cornell, D.G. and Gottlieb, C.C. An efficient algorithm for graph isomorphism. J. Assoc, comput. Mach., 17., 51-64. (1970).

(13) Barrow, H.G., Ambler, A.P. and Burstall, I.M. Some Techniques for Recognizing Structures in Pictures. Frontiers of Pattern Recognition (ed. S. Watanabe) New York: Academic Press, pp. 1-29. (1972).

(14) Michie, *D,* 'Memo' functions and machine learning. Nature, 118, 19-22. (1968).

(15) Nevins, J.L., Sheridan, T.B., Whitney, D.E. and Woodin, A.E. The Multi-Moded Remote Manipulator System. E-272O. Charles Stark Draper Laboratory, MIT, Cambridge, Massachusetts. (1972).

(16) Popplestone, R.J, Solving equations involving rotations. Research Memorandum MIP-R-99. Department of Machine Intelligence, School of Artificial Intelligence, University of Edinburgh, (l973)

Reference on cliques which came to hand after paper was typed:- „

(17) Akkoyunlu, E.A. The Enumeration of Maximal Cliques of Large Graphs. SIAM Journal on Computing. Vol. 2. No. 1. March. (1973)

FIGURE 1

FIGURE 2

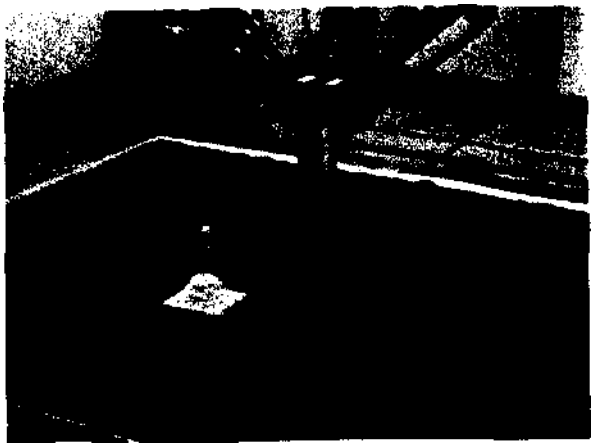


FIGURE 3

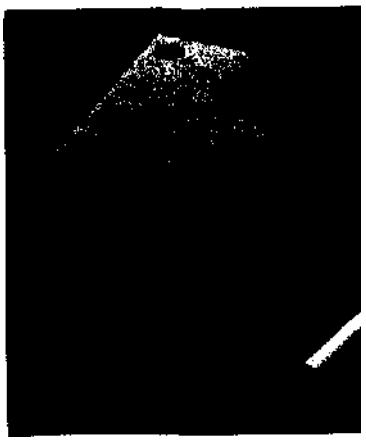


FIGURE 4

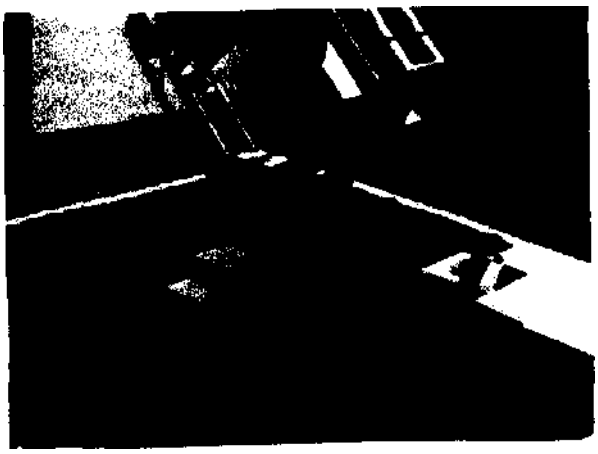


FIGURE 5

FIGURE 6

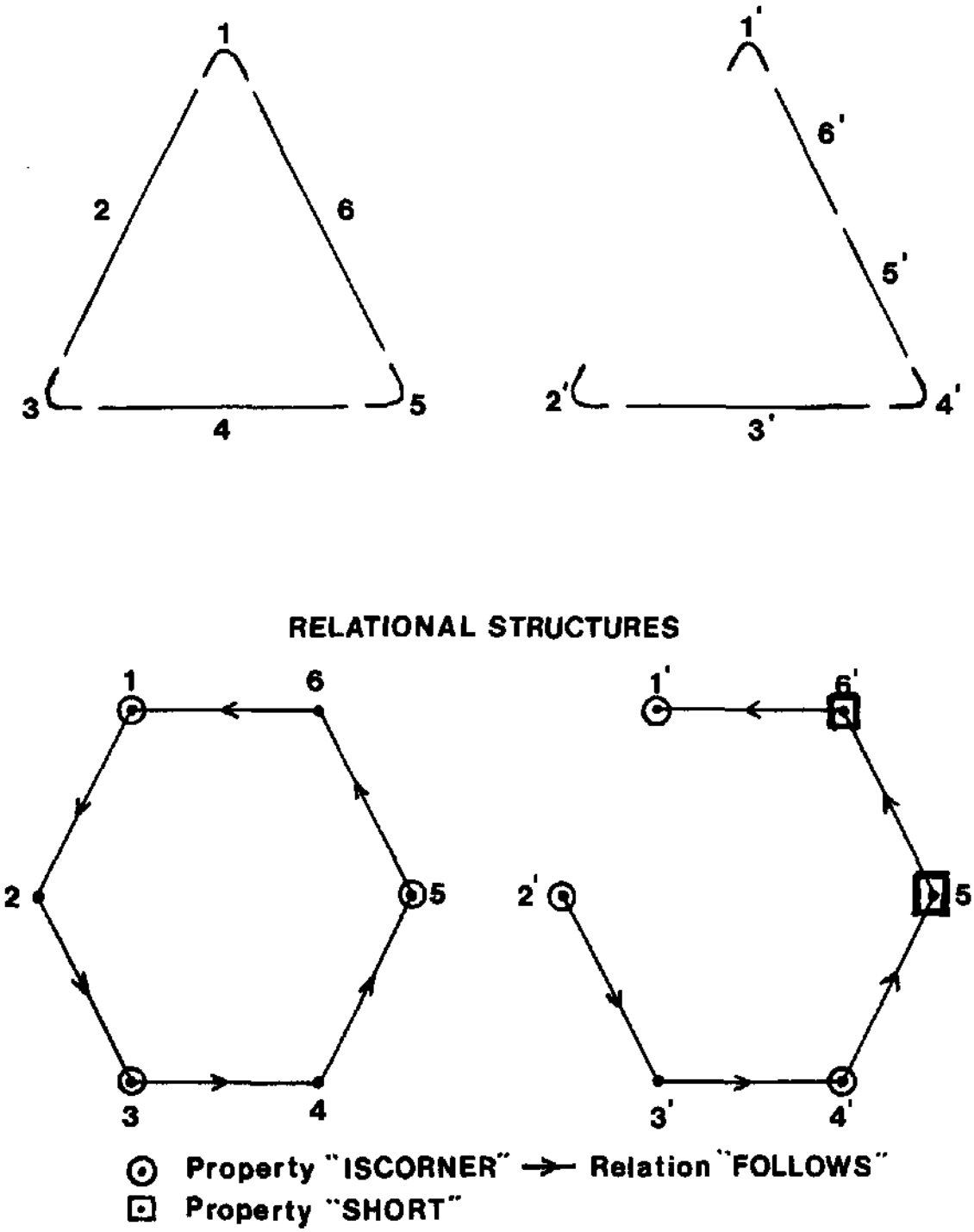

OUTLINES
1
2
6
3
4
5
1'
6'
5'
2'
3'
4'
RELATIONAL STRUCTURES
1
6
2
5
3
4
1'
6'
2'
5'
3'
4'
Property "ISCORNER" → Relation "FOLLOWS"
Property "SHORT"

FIGURE 7

MAXIMAL CLIQUES

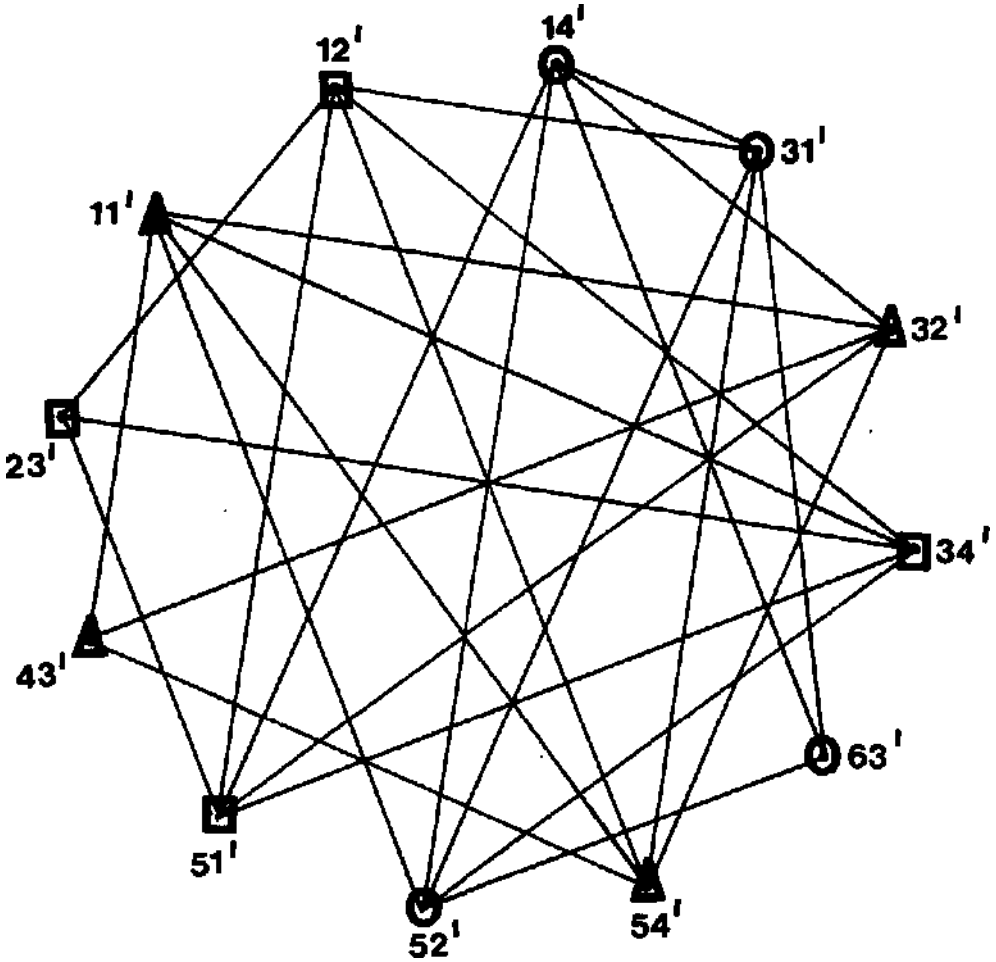


FIGURE 8

A CONSTRAINED MOVE

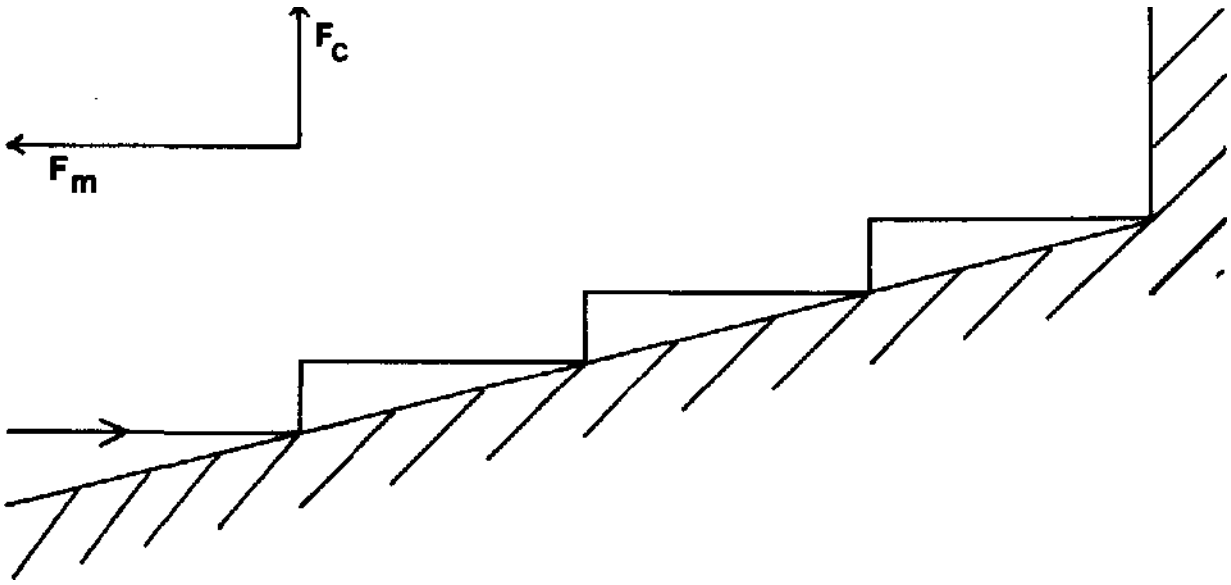


FIGURE 9

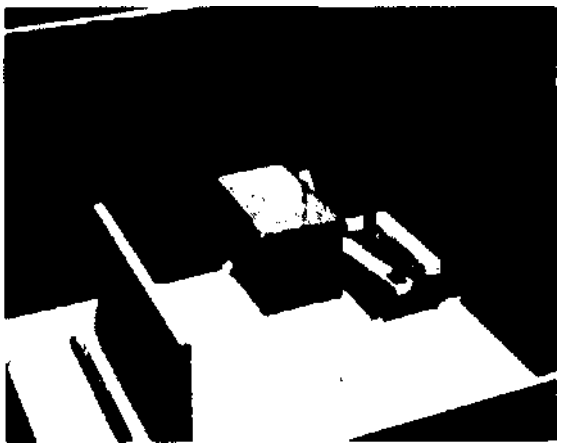


FIGURE 10

PLANNING CONSIDERATIONS FOR A ROVING ROBOT WITH ARM*

Richard A. Lewis and Antat K. Bejczy

Guidance and Control Division
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91103

Abstract

The Jet Propulsion Laboratory is engaged in a robot research program. The program is aimed at the development and demonstration of technology required to integrate a variety of robntic functions {locomotion, manipulation, sensing and perception, decision making, and man-robot interaction) into a working robot unit operating in a real world environment and dealing with both man-made and natural objects. This paper briefly describes the hardware and software system architecture of the robot breadboard and summarizes the developments *to* date. The content of the paper is focused on the unique planning considerations involved in incorporating a manipulator as part of an autonomous robot system. In particular, the effects of system architecture, arm trajectory calculations, and arm dynamics and control are discussed in the context of planning arm motion in complex and changing sensory *and* workspace environments.

KEY TERMS: Robot system; Robot system planning; Robot breadboard architecture; Arm motion planning; Arm control; Arm dynamics; Sensors for manipulation; Manipulating in natural and constrained environment.

1. 0 introduction

Autonomous goal-directed coordination of locomotion, manipulation, and sensation and perception in a semi-natural environment is the capability being sought by the JPL Robot Research Program. The initial goal of the program is to demonstrate the integration of sensory and motor functions in the autonomous performance of manipulation and locomotion tasks in response to global commands issued by an operator. The long-range goal is to develop, test, and display concepts of robot structure, system integration and operation, and machine intelligence for the design and use of adaptive autonomous machines for advanced space and planetary exploration. The JPL program utilizes results of progress obtained at other institutions engaged in robotics and *artificial intelligence work reviewed in Ref. 1.* (Ref. 1 contains an extensive list of related literature.) The robot breadboard itself is a mobile vehicle (similar to that used by the astronauts on the moon) equipped with a six degree-of-freedom manipulator (a modified version of the Stanford Electric Arm, see Ref. 2), a complement of sensors (TV, laser range finder, navigation and guidance sensors, tactile sensors, and, eventually, proximity sensors), and a local mini-computer in communication with remote computers, graphic displays, and operator consoles.

*This paper presents the results of one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract No. NAS7-100, sponsored by the National Aeronautics and Space Administration.

In this paper, we focus on the particular planning considerations involved in incorporating a manipulator as part of a total robot system operating in a complex sensory environment and dealing with both man-made and natural objects. First the architecture of the JPL robot breadboard and then the different aspects of planning manipulator motion are discussed.

2. 0 Breadboard System Architecture

The breadboard is divided into six functional subsystems: locomotion, manipulation, environment sensing and perception, computing and data handling facilities, robot executive (REX), and operator-robot interface. Each subsystem contains both hardware and software. Subsystem design is based solely on criteria of functional compatibility, performance, growth capability, and convenient interfacing. Total robot system integration will be studied experimentally and different concepts wil! be demonstrated in successive stages.

2. 1 Breadboard Hardware

The major subsystem hardware elements are shown in Fig. 1, indicating also the physica] size of the moving part of the breadboard.

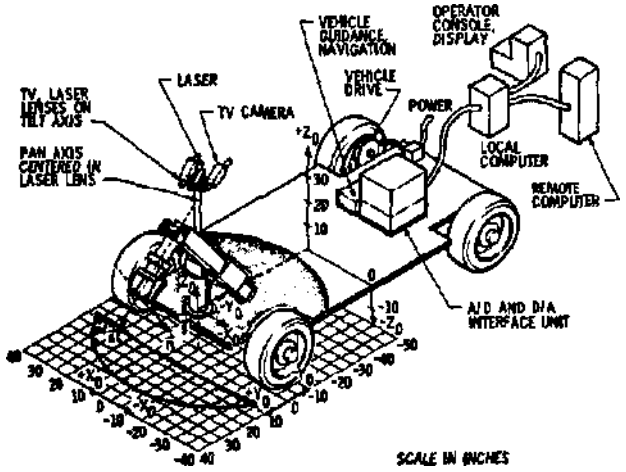


Fig. 1. Breadboard Hardware Configuration

The vehicle, on loan from Marshall Space Flight Center, provides a flat and relatively stable platform for mounting breadboard elements to be moved around in the environment. Total effective load capacity *o£* the vehicle is about 500 pounds. Travel speed will be limited to 1 mile/hour. The vehicle has Ackerman-type double steering; the two ends can be steered in the same or opposite directions. Alternatively, one or the other end only can be steered. Each wheel is independently driven by a DC torque motor. Currently, the vehicle has only dynamic braking. The suspension has a modified

independent spring action at each wheel. Inputs to the vehicle navigation, guidance, and control system are furnished by odometers mounted on the front wheels (providing vehicle center line distance travel information), a "ruggedized" directional gyro compass (providing directional reference), and wheel drive motor tachometers (providing information on vehicle velocity).

The manipulator is a modified version of the Stanford Electric Arm described in detail in Ref. 2. It has six degrees of freedom, allowing any desired hand position and orientation in an open or slightly obscured workspace. The reachable set of points (the workspace) is within a radius of 52 inches measured from the origin of the manipulator base reference frame. (See Fig. 2.) The six joints connecting

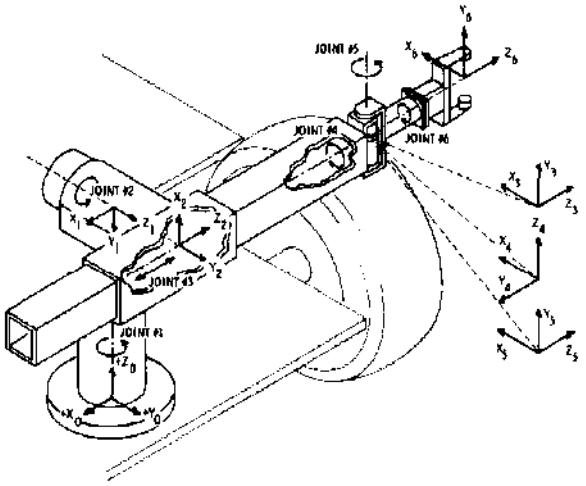


Fig. 2. Reference Frames
For .Link-Joint Pairs of Arm

the links from the base to the hand are in the following sequence: two rotary joints (providing shoulder azimuth and elevation action), a linear joint {providing in and out reach action), and three rotary joints (providing the wrist action). The hand is presently a simple parallel jaw mechanism. The joints are driven by permanent magnet DC torque motors geared directly to the corresponding links. Depending on the relative position of the links, the arm can handle loads of up to 5-8 pounds Earth weight. The arm servo control utilizes analog position measurements from the joint outputs and analog velocity measurements from the motor shafts. Holding torque at each joint is provided by electromagnetic brakes. The arm's structural stiffness and tight servo control can provide hand positioning accuracy within a few tenths of an inch. A suitable articulated and adaptively controlled hand will be added at a later date.

Environment sensing and perception is mainly obtained from two sources: TV cameras and laser ranging. The laser ranging device is a GaAs pulsed mode laser with fast pulse (~10 ns). The beam is pointed by a gimballed mirror and detected by a photomultiplier. Provisions are made for multipulse averaging using analog integration and variable averaging time. The sensing range is tentatively up to 150 feet. The design is based on previous JPL experiments (Ref. 3). Related data handling problems are treated in Ref. 4. The vision system consists of two identical and optically parallel vidicon TV cameras which will provide'digitized stereoscopic input to both scene analysis and operator display (Ref. 5). A 729 by 729 resolution sequential column digitizer furnishes video data for computer-rate digital picture processing and operator display. The TV cameras and laser are mounted on a pan and tilt mechanisms referenced to a common coordinate system and will be used as an integrated scene analysis subsystem. Arm-mounted proximity sensors {described in Ref. 6) and tactile sensors will at a later date augment the environment sensing and perception subsystem.

The proximity sensor is a small (about 0. 3 cubic inch) electro-optical device with a small ellipsoid-shaped sensitive volume permanently focused at a distance of a few inches in front of the sensor. If this proximity sensor is mounted to an appropriate place on the hand, the sensitive volume will move with and ahead of the hand at a known distance relative to a reference point on the hand. A voltage signal will appear when the sensitive volume "touches" a solid surface as the hand approaches the surface. This voltage signal can be used to guide and control the terminal motion of the hand in direct response to sensed relative hand-object position and orientation. Of course, several proximity sensors can be mounted on the hand, providing several sensitive volumes in a known pattern around the hand and facilitating the design of a versatile conditional terminal guidance and control logic for hand motion.

The computing and data handling subsystem architecture is currently based on a remote PDP-10 in the ARPA net as an off-line computer. This will be connected to a local real time computer performing realtime robot control and I/O functions. (See Fig. 3.) The remote computer system will be used

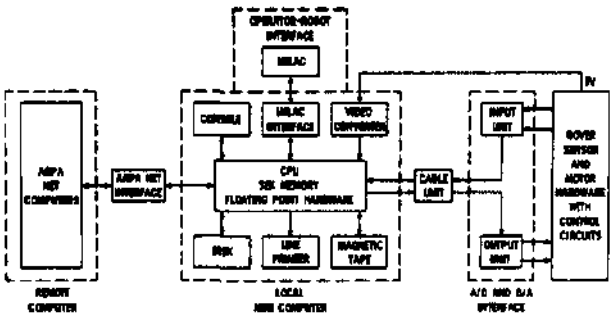


Fig. 3. Computing and Data Handling
Subsystem Architecture

to process TV and laser pictures, to construct the "world model, " to operate the different subsystem planning programs, and to execute top-level decision-making programs of the robot executive (REX). REX is described in detail below.

The realtime computer will interface with the robot through input and output units which contain A/D and D/A converters. A Cable Unit will contain the necessary logic devices allowing the robot to be tethered to the CPU via a 50-100 foot cable. TV data from the robot will interface directly (via the Video Converter) to the CPU through a separate cable. A disk storage unit will be used for fast, random access mass storage and will serve to store the operating

system routines, robot eupport programs, TV and laser pictures, and status files for the operator terminal. A magnetic tape unit will be used for performing system diagnostics and for entering operating system programs and any robot subsystem programs developed in the off-line computer. The operator console will allow operator interaction with the system and development of subsystem support programs.

The operator-robot interface functions will be performed through an IMLAC terminal connected to the realtime computer. (See Fig. 3.) During program development, the IMLAC will also be used for simulation studies. The IMLAC terminal includes a CRT display, a teletype, and a read/write cassette recorder.

2. *Z* Breadboard Software

The software system architecture is essentially hierarchical with the robot executive (REX) controlling and monitoring the various software subsystems. REX performs problem solving, interacts with the human input command structure, manages the "world model," and calls the major subsystem software modules (vehicle, arm, etc.). The major subsystems are designed to be largely independent of each other. Necessary data concerning the state of the robot and environment used by the various subsystems are furnished through the "world model, "

A Master Control Program (MCP) ties the various subsystem programs together and acts as an operating system for them. Present plans are to design the MCP in the remote computer using the mechanisms in SAIL (Stanford Artificial Intelligence Language , Refs. 7 and 8) for the creation and control of concurrent processes. Operating system modifications are also planned based on the use of the TENEX paging system (Ref. 9) which makes available interrupts of various types not incorporated into SAIL, Subsystem programs can be written in SAIL or possibly in other languages. SAIL provides easy linkages to FORTRAN and assembly languages.

The total breadboard system is of experimental nature. Thus, the software system is intended to be expandable and evolutionary.

3. 0 Planning Manipulator Motion

This task involves three separate efforts: system architecture effects, trajectory planning, and manipulator dynamics and control. These are, respectively, planning for manipulator motion, planning *o{* motion, and execution of planning. *Planning for* manipulator motion occurs at the system level in the selection, design, and placement of robot hardware. Planning of motion involves the selection and implementation of methods of specifying particular motions and motion constraints. Execution planning deals with motion control implementation schemes. We now consider each of these three separately.

3. 1 System Architecture Effects

Placement of the manipulator along the center line of the vehicle about 8 inches from the front edge of the vehicle platform allows a reasonable workspace for the manipulator on the ground (which is 18 inches below the platform) while still permitting access to tools and sample storage bins near the center of the platform. This placement does, however, give rise to several motion constraints; the manipulator can easily collide with the platform, the front edge of the platform, the wheels, and the wheel drive motors, even though the basic vehicle was modified extensively to minimize this problem.

The selection and placement of sensors give rise *to* additional motion constraints for the manipulator as well as allowing the manipulator to know its world. The manipulator makes use of both external and internal sensors. In the initial configuration of the robot, the primary external sensors to be used by the manipulator are the dual TV and laser range finder. These are used to determine a priori manipulator targets and are assumed to have sufficient resolution in the initial simplified robot environment for effective target specification. Later operation of the robot in richer and more perceptually complex environments rendering the sole *use* of *these* sensors open to question will be accompanied by the use of conditional arm control loops regulated by direct inputs from tactile and proximity sensors. Use of these latter devices in conjunction with an adaptive, articulated terminal effector will permit the arm to respond directly to relevant aspects of the environment.

Fig. 4a shows a proximity sensor mounted on the hand,, while Fig. 4b shows the concept of proximity sensor application for terminal guidance and control of hand motion in "distance seeking" and "distance keeping" *modes of* operation. A more detailed treatment of proximity sensor application to manipulator control can be found in Ref. 10.

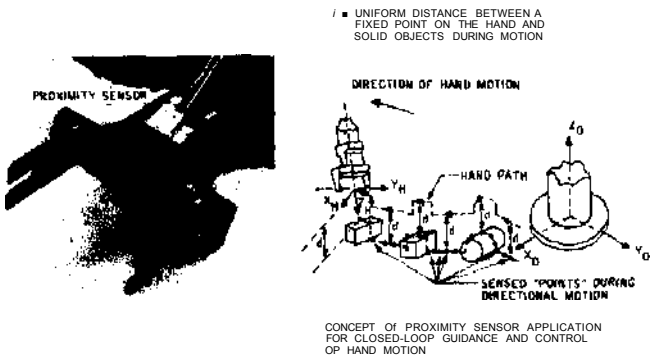


Fig- 4. Terminal Guidance and Control Of Hand Motion Using Proximity Sensing

Our robot-hand application calls for handling a variety of both regular (man-made) and irregular (natural) objects of different size and weight in various manipulative tasks. Therefore, a versatile hand-finger mechanism will ultimately be required. An articulated and adaptively controlled hand (using design and control principles similar to those applied in prosthetic hand research; see Refs. 11 and 12) will not require detailed a priori information on objects to be handled and will ease the control of many details of a grasping motion since the hand, while gripping an object and monitoring only one actuator, adapts itself "reflexively" to the shape, size, orientation, and weight of the object.

Placement of tactile and proximity sensors on the manipulator itself creates no new obstacles or additional constraints, but placement *of* the TV/laser head is a Berious problem. The present configuration places these sensors on a mast above the arm support

post and represents a tradeoff between good viewing angle and keeping out of the way of the manipulator. Even so, it is possible for the manipulator to collide with the TV/laaer head, either in normal motion or in switching from a right arm to left arm configuration. However, since the manipulator and TV/laser systems are not run simultaneously in this breadboard and the TV/laser head is mounted on a pan/tilt mechanism, these sensors can be moved out of the way.

The physical dimensions of the manipulator have been modified from the original Stanford design to permit a greater workspace on the ground. Specifically, the arm support post has been reduced by two inches and the extendable boom lengthened considerably.

The initial configuration of the robot thus provides the tools for deterministic planning of arm motion based on a priori TV/laser data with motion implementation based on feedback fronvinternal position (pot) and rate (tach) sensors. Only primitive tactile feedback in the initial configuration allows for some conditional modification of plan. In subsequent configurations, proximity sensing and an adaptive terminal effector will permit more flexible and environment-responsive manipulator motion planning. In all cases, rover hardware design constrains manipulator motion by presenting a series of permanent obstacles to the arm.

3.2 Trajectory Planning

The term "trajectory" is here meant to refer to some description, partial or complete, of the path that the arm follows.

Trajectory "planning" is the activity preceding arm motion (that is, trajectory execution), the purpose of which is to constrain or otherwise define that motion. Target and environmental obstacle information are here assumed to be provided by REX and the "world model."

The degree to which the trajectory is to respond to external sensing during trajectory execution defines a continuum of trajectory planning. At one extreme is purely deterministic trajectory planning. Any external sensing to be done is performed during the planning stage; only a catastrophe halts execution of the planned path. Internal sensing is used throughout execution to maintain the adherence of actual motion to plan. Deterministic planning assumes a static world during arm motion as well as sufficient a priori knowledge and execution accuracy capabilities, and limits adaptive control. Towards the other extreme is conditional planning, the nature of which is highly dependent on the specific external sensors used. The benefits of conditional planning are flexibility of response and possible reduction of planning time at the possible cost of increased real-time computation requirements.

In the initial configuration, deterministic planning similar to that used in the Stanford hand-eye project (Refs. 13 and 14) is to be employed. Reasons include the initial lack of proximity sensing, the adequacy of TV/laser a priori information in the initial simplified environment, and the fact that most of the obstacles to arm motion are (in the initial configuration) permanent obstacles known a priori and resulting from the placement of vehicle hardware. Conditional planning is to be implemented and interfaced with existing deterministic planning at a later time.

A related distinction concerns the manner in which the trajectory plan is specified. Either a sequence of a few points or the complete time history of the arm (that is, the path) can be planned. Arm motion in a constrained workspace involves path planning.

Deterministic path planning can be performed in joint-variable space or in 3-space. In the former case, the time history of each joint is planned; it is the combination of the time histories of the joint variables that describes the motion of the arm. In the latter case, it is the motion of a particular point on the manipulator (commonly, a point on the hand) that is planned; the required joint variable time histories are derived from the plan. The advantage of planning in joint-variable space is that the plan is formulated more directly in terms of the variables to be controlled during motion. The associated disadvantage is the difficulty in determining where the various links will be during motion, a task required to guarantee avoidance of collisions with the other parts of the robot, the natural environment, or even with the arm itself.

Constraining the fingertips to describe an elliptical arc is an example of planning in 3-space. In addition to the difficulties in finding the path described by other points on other links of the arm, there is also the problem of determining the kinematic sequence of joint variable values required to implement the plan.

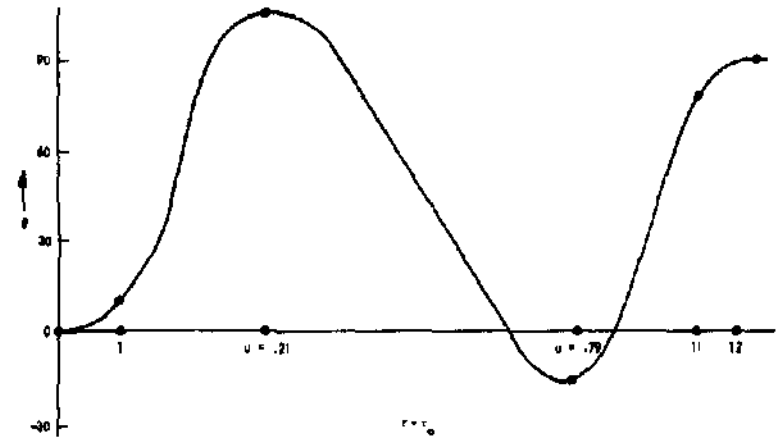
In the Stanford Hand-Eye Project (Refs. 13 and 14), the focus to date has been on deterministic path planning in joint-variable space, with some conditional planning. Specifically, the time histories of the joint variables have been specified in terms of sequences of polynomials with continuity of joint variable value and its first two derivatives guaranteed at the boundary points of polynomials in the polynomial sequence. The number of polynomial segments and specified motion contraints determine the total number of coefficients required for a complete quantitative specification of joint trajectory. The JPL robot research program uses a modified version of the Stanford planning algorithm.

Fig, 5a shows the time history of a joint variable described by a cubic, a quintic, and another cubic. It has been found that trajectories using polynomials of degree five or higher typically wander, as shown. This behavior appears in observation as gross extraneous motion of the arm.

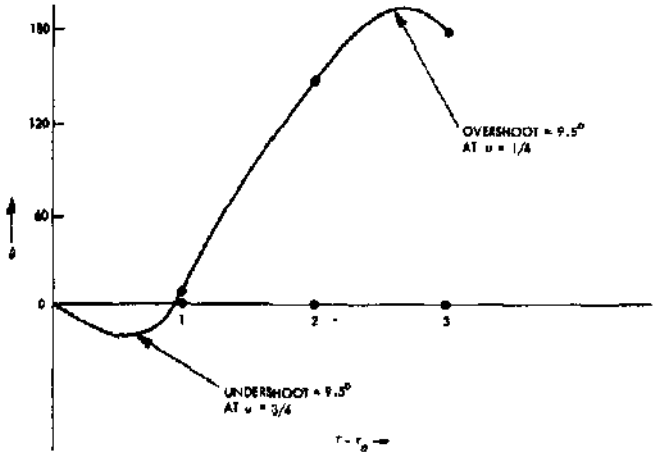
Use of a quartic-cubic-quartic trajectory reveals a somewhat different problem. As shown in Fig. 5b, the desire to assure an appropriate direction of departure and approach of the terminal effector can be thwarted by the tendency of the quartic-cubic-quartic trajectory to overshoot or undershoot its endpoint values. The Stanford Hand-Eye project used both of these polynomial sequences, eliminating overshoot by special code.

A third polynomial sequence, five cubics, is being implemented for the JPL arm. This trajectory appears to minimize the "wander" and "overshoot" problems. Typically, as in Fig. 5c, there is no overshoot; wander, when it occurs, is small.

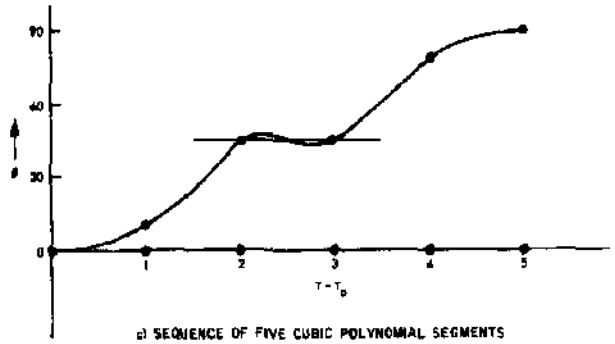
Obstacle avoidance has been implemented for the JPL arm in two ways. The first method consists of the specification of an additional safe intermediate position for the joint(s) most critically affecting arm motion in the sensitive direction. An additional



a) SEQUENCE OF CUBIC-QUINTIC-CUBIC POLYNOMINAL SEGMENTS
(showing "wandering")



b) SEQUENCE OF QUARTIC-CUBIC-QUARTIC POLYNOMINAL SEGMENTS
(showing "overshoot")



a) SEQUENCE OF FIVE CUBIC POLYNOMIAL SEGMENTS

Fig. 5. Various Polynomial Joint Trajectories

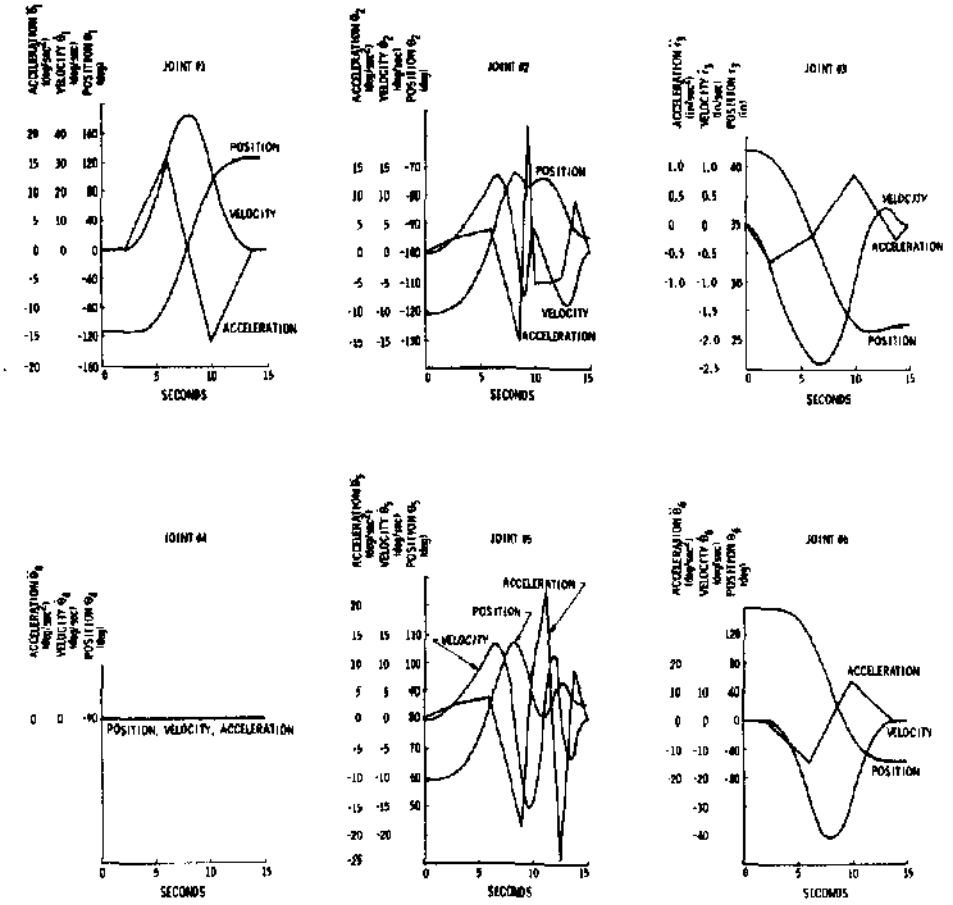


Fig. 6. Joint Variable Time Histories For a Given Task With Obstacle Avoidance

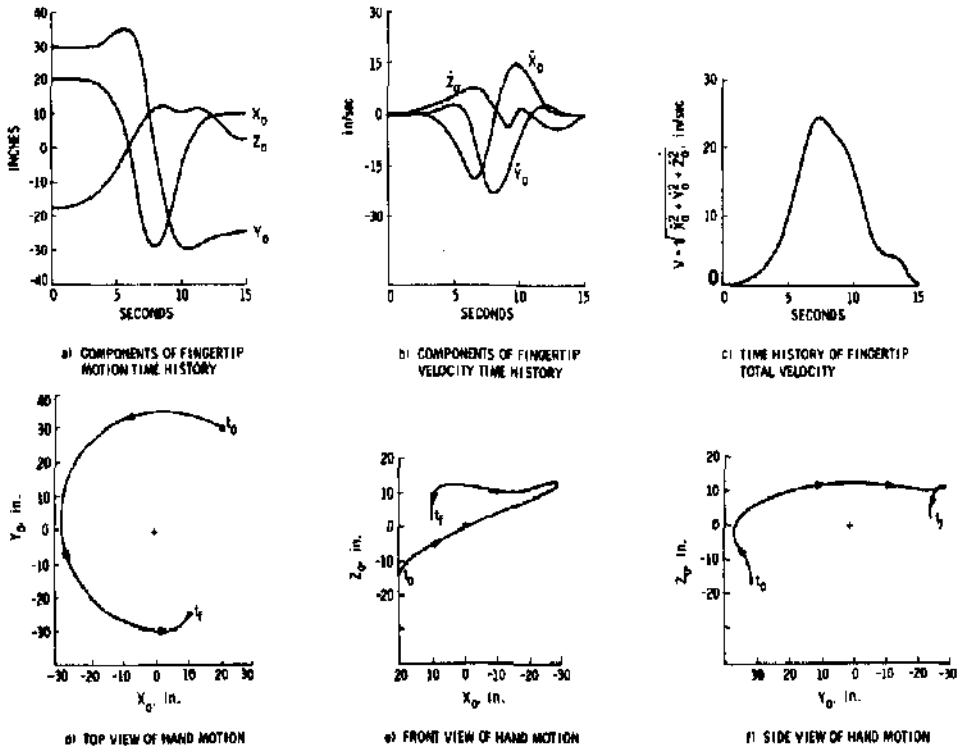


Fig. 7. Components and Projections Of Hand Motion in the Base Reference Frame (For the Same Task Shown in Fig. 6.)

cubic is required for each additional intermediate position specified. The effect of implementing this type of obstacle avoidance is shown in the trajectory illustrated in Figs. 6 and 7. Fig. 6 describes joint variable motion and Fig. 7 illustrates the projection of the same planned trajectories in the base coordinate system, as well as the associated total velocities.

As seen in Fig. 7a, the task is to lift an object from the ground (from x_Q = 20", y_O= 30", z_Q - -17" in the base reference frame) and deposit it on the vehicle platform at x_Q = 10", y_O = -25", z_O - *2".* The hand orientation is also specified at both end points of the trajectory. Further, to aid the determination of the lift-off and terminal approach phases of the joint trajectories, two intermittent hand positions are also specified. As seen in Fig- 6, joint trajectories #2 and #5 contain, respectively, 9 and 8 segments due to obstacle avoidance.

A second method of obstacle avoidance is called the "freeway method. " Precomputed safe trajectories *(called* "freeways") relating commonly *accessed* points are utilized in conjunction with entrance and exit "ramps" relating planned arm configurations to existing freeways. This freeway method is potentially useful in avoiding obstacles permanently affixed to the vehicle, such as the TV/laser head and support, the wheels and wheel motors, and the vehicle platform. Presently, there are 14 such permanent obstacles on the JPL robot. The freeway method can be used more frequently if the vehicle is positioned in a predetermined standard manner with respect to objects of intended manipulation.

Obstacle detection is performed by the relatively cumbersome method of checking for collision with all possible objects at various points along the trajectory. Of course, for many obstacles, the safety of several links mast be examined. In the case of permanent obstacles, however, the invariant property of the relationship has been exploited to produce a series of increasingly complex tests. Thus, in many cases, simple checks of joint variable values can assure safe motion,

3. 3 Manipulator Dynamics and Control

Execution planning deals with the specification of control laws and the design of control schemes whose implementation will assure that the physical motion of the arm will follow the desired motion.

Arm *motion* between distant points without prescribed continuous trajectories between the points can simply be controlled by driving each motor at some preset rate and terminating the motor drive at each joint when an appropriate signal (potentiometer or some external sensor) indicated that the joint position had reached the preset or desired terminal value, Arm trajectories planned in terms of continuous space-time coordination of joint motions, however, require that the joints be driven to comply strictly with the planned time histories of joint positions.

Several techniques are available to build a suitable position servo for each joint drive. (See Ref. 15.) An appealing computer-oriented servo technique (used *in* the Stanford Hand-Eye project) *is* to compute the required torque or force as a function of time for each joint drive, accounting also for gear ratio, efficiency, and possible nonlinearities, and construct the joint position servo loops around the computed nominal torque or force inputs. (See Fig. 8.)

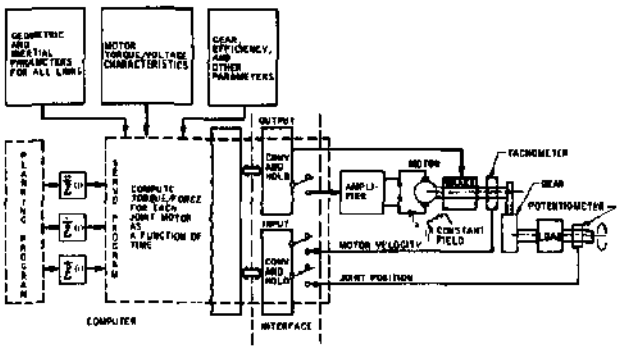


Fig. 8. Manipulator Servo Scheme

The dynamics of motion at the six joints of the arm is described by a coupled set of six second order nonlinear differential equations with time-varying (in fact, with state-varying) coefficients. There is no simple proportionality between torque {or force) acting at one joint and the acceleration of the same jyint when several joints are in motion simultaneously. Even if only one joint moves at a given time, the proportionality between torque and acceleration is a complex function of the actual configuration of all links ahead of the moving joint and any load in the hand. The total variations in link inertias as seen at the joint drives due to changes in arm link configuration or load in the hand have been calculated for the JPL arm (Ref.]6) and are shown in Fig. 9.

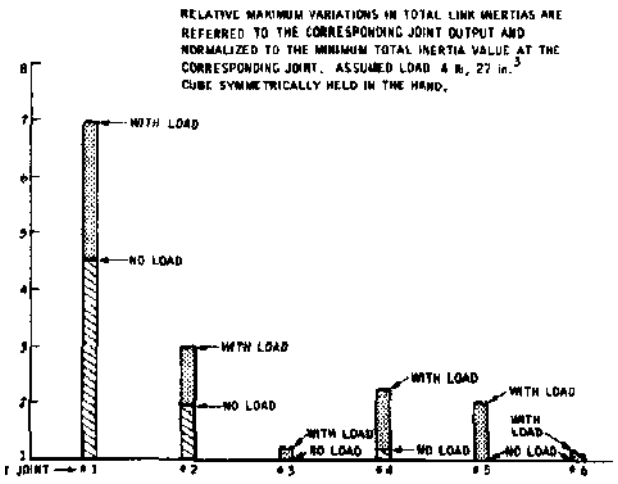


Fig. 9. Relative Maximum Variations In Total Link Inertias

In the case of simultaneous motion of several arm joints, the effective torque (or force) acting at each joint is the sum of a number of dynamic components: inertial acceleration of the joint; reaction torques or forces due to acceleration and velocity at other joints; gravity terms. The relative importance of the various dynamic components related to the planned motion displayed in'Figs. 6 and 7 and computed for the actual kinematic and inertial parameters of the JPL robot research arm is illustrated

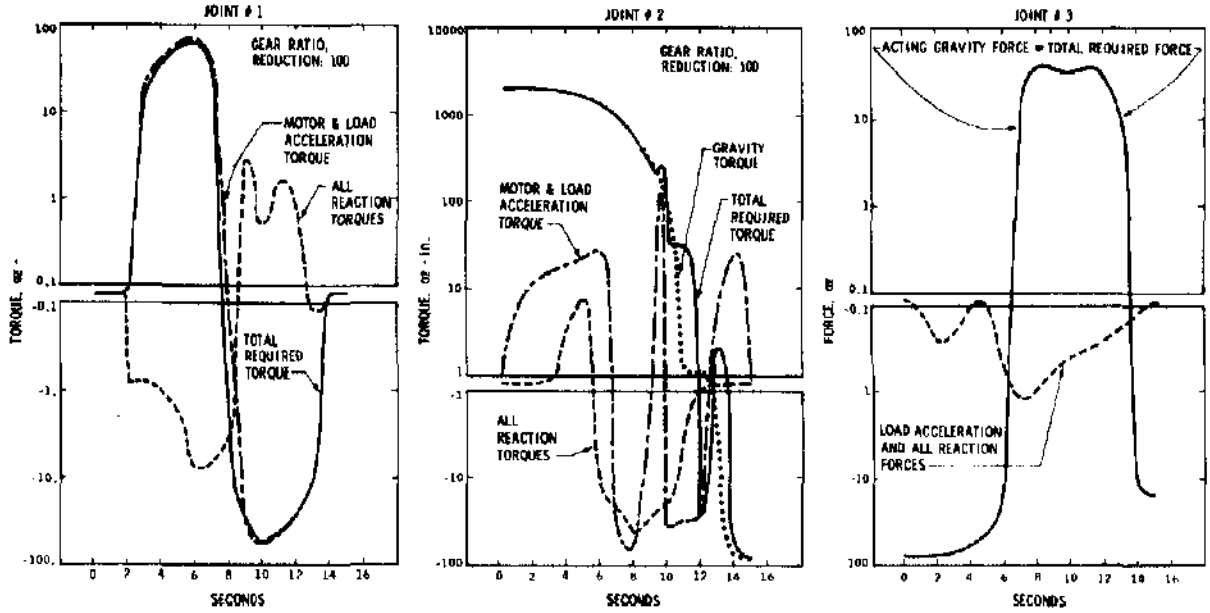


Fig. 10. Torque and Force Variations Referred to the Output (First Three Joints and for the Same Task Shown in Fig. 6)

in Fig- 10. To simplify the displayed variations, only motions and inertias of the first three joints are accounted for in the diagrams of Fig. 10. If the speed of motion (or rather, the total lime of motion) is changed by a factor n, then the acce leration and velocity dependent torque and force-components of Fig. 10 can simply be scaled by a factor n^2.

Two questions are currently investigated (Ref. 16): to what extent should the reaction components be accounted for, and in what form should the state-varying dynamic coefficients be specified to ease control scheme implementation.

4. 0 Summary

The integration of several robot subsystems into a functioning autonomous adaptive machine requires significant planning considerations on all system levels. The JPL robot research program is currently focused on subsystem planning and design and is in the process of implementing results in hardware and software. The manipulator itself has been manufactured and is undergoing final testing. Vehicle modifications are partly completed. Construction of the TV and laser systems are in progress. Selection of complementary sensors and other hardware and acquisition of a realtime controller (the local computer) are underway. The design and building of the A/D and D/A interface units are also underway. Development *of* the executive and subsystem software and the master control program have begun. The future concerns are mainly related to the implementation of subsystem execution programs, planning for subsystem interaction, and the integration of subsystems into a unified robot breadboard.

The outlined considerations and results related to the design of a self-contained planning algorithm for manipulator control, when the manipulator is part of a total robot system, suggest conducting further work in two directions: direct path planning in the object space; and truly adaptive manipulator control through unification of deterministic and conditional elements in the planning algorithm.

Acknowledgment

The preparation of the first part of the paper benefited from discussions with several members of the JPL AI working group.

References

1. Bejczy, A. K. , "Machine Intelligence for Autonomous Manipulation, " Proceedings of the First National Conference on Remotely Manned Systems, California Institute of Technology, Pasadena, California, September 13-15, 1972.

2. Scheinman, V. D. . "Design of a Computer Controlled Manipulator, " Computer Science Department, Stanford University, Artificial Intelligence Project, Memo No. 92, June 1969, and
Dobrotin, B. M. , Scheinman, V. D. , "Design of a Computer Controlled Manipulator for Robot Research, " see in this Proceedings, Third International Joint Conference on Artificial Intelligence.

3. Escobal, P. R. , et al., "3D Multilateration Cone, A Precision Geodetic Measurement System, " TM 33-605, Jet Propulsion Laboratory, California Institute of Technology, March 15, 1973.

4. Hong, J. P. , "Registration of Points, " submitted to the IEEE Transactions on Computers.

5. Levine, M. D. , O'Handley, D. A. , Yagi, G. M. , "Computer Determination of Depth Maps, " to appear in Computer Graphics and Digital Display.

6. Johnston, A. R. , "Optical Proximity Sensors for Manipulators," TM 33-612, Jet Propulsion Laboratory, California Institute of Technology, May 1, 1973.

7. Feldman, J. A., Sproull, R. F. , "System Support for the Stanford Hand-Eye System, " Proceedings of the Second International Joint Conference on Artificial Intelligence, September 1-3, 197], London, England.

8. Feldman, J. A., et al., "Recent Developments in SAIL - An ALGOL-Based Language for Artificial Intelligence," 1972 FJCC Proceedings, December 5-7, 1972, Anaheim, California.

9. Bobrow, D. G., et *al.*, "TENEX, A Paged Time Sharing System for the PDP-10," Proceedings, ACM Third Symposium on Operating Systems Principles, Stanford University, October 18-20, 1971.

10. Bejczy, A. K., Johnston, A. R., "New Techniques for Terminal Phase Control of Manipulators," TM 33- (to be published), Jet Propulsion Laboratory, California Institute *of* Technology.

11. Tomovic, R., Boni, G., "An Adaptive Artificial Hand," IRE Transactions on Automatic Control, April 1962.

12. Nightingale, J. M., Todd, R. W., "An Adaptively Controlled Prosthetic Hand," Engineering in Medicine, Vol. J, No.], October 1971.

13. Paul, R. C., "Trajectory Control of a Computer Arm," Proceedings of Second International Joint Conference *on* Artificial Intelligence, London, England, September 197 1.

14. Paul, R. C, "Modelling, Trajectory Calculation and Servoing of a Computer Controlled Arm," Ph.D. Thesis, Computer Science Department, Stanford University, August 1972.

15. Markiewicz, B. R., "Analysis *of* the *Computed* Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator," TM 33-601, Jet Propulsion Laboratory, California Institute *of* Technology, March]5, 1973.

16. Bejczy, A. K., "Robot Arm Dynamics and Control," TM 33- (to be published), Jet Propulsion Laboratory, *California* Institute *of* Technology.

CONTROL ALGORITHM OP THE WALKER CLIMBING OVER OBSTACLES

D.E. Okhotsimski, A.K, Platonov
U S S R

Abstract. The paper deals with the problem of development the multilevel control algorithms for six-legged automatic walker, which provide the walker with the possibility to analyse the terrain profile before it while moving over rough terrain, and to synthesize adequate, rather reasonable kinematics of body and legs for walker's locomotion along the route and climbing over obstacles on its way. DC simulation and analysis of walker's model moving image on DC display screen make it possible to evaluate the algorithms developed and to find ways for their improvement.

Key words: six-legged walker, DC simulation, control algorithm, data processing, obstacle overcoming.

The paper deals with the problem of control algorithm synthesis for a six-legged walker. It is supposed that the walker is supplied with an onboard digital computer. Rather a complicated algorithm may be used, which provides walking over rough terrain and climbing over some isolated obstacles. It is also supposed that the walker is equipped with a measurement system giving information about the terrain relief. Measurement data are processed by DC and used when making decision.

An effective method of testing the algorithms is their simulation on a digital computer with a display unit. It is possible to simulate the walker itself, terrain relief, measurement system functioning! data processing, decision making and walker controlling. Observing on the CRT screen the moving image of the vehicle walking over the terrain, it is possible to check the functioning of the algorithms, to estimate their effectiveness and to find ways for their improvement .

This paper deals with the algorithms in the range from the environment information (input) to the vehicle kinematics (output).. The problem of terrain measurement data processing and measurement controlling are also investigated. The simulation results are discussed.

On the first stage of the control algorithm synthesis it was assumed that all necessary information about the terrain relief was got and processed and was kept in the computer memory in the form convenient for its further use in the decision-making algorithm.

Several types of six-legged walking system were investigated. Schematic image of one of them is seen in Pig. I. All six legs of the walker have equal geometrical parameters and equal orientation of the joint axes. Each leg has three degrees of freedom in the joints: two in the hip joint and one in the knee. The first *hip-joint* axis is perpendicular to the plane of the vehicle body, while the second one is parallel to the body plane and perpendicular to the thigh. The knee axis is parallel to the second hip-joint axis. The total number of degrees of freedom in six legs amounts to eighteen. The vehicle body has no kinematic constraints, and therefore it may have six degrees of freedom in its motion relative to the supporting surface.

The walker of this type has rather rich kinematic feasibilities which may be used to provide the vehicle's adaptivity to the terrain. The problem is to synthesize appropriate control algorithms which, could organize the walker kinematic in a reasonable way for the effective solving of different locomotion tasks.

It was reasonable to design control algorithms as a multilevel hierarchical system. The following 5 levels were adopted:

1. Leg. This level is the lowest one.It is necessary to synthesize leg motion during the support and swing phases and to avoid small-size obstacles.

2. Leg coordination. This level is higher than the previous one. The leg-coordination algorithms provide support scheduling of the legs, i.e. they generate sequences of "up" and "down" times for all legs. The condition must be satisfied: The stability margin of the vehicle should be always *no* less than a given value.[18]

3. Standpoint sequence. This level fixes in advance several supporting points on the support surface. In a simple case, if the terrain relief allows it, the level generates a regular standpoint sequence described by two parameters: the gauge width and the stride length. In more complicated cases it is necessary to plan an irregular standpoint sequence, e.g. for some cases of climbing over obstacles.

4. Body. The output of this level is the parameters of motion of the walker's centre of mass both along the route and in vertical direction, and the parameters of body rotation (pitch, jaw, roll).

5. Route. The route planning level is the highest one. Up to now the route of the walker has been planned by an operator.

Pig. 2 shows interlevel information flow. The complex of control algorithms is dash-lined. Dotted lines indicate the flow of terrain information to different levels.

It was reasonable to begin designing the algorithms from lower levels and then pass on to the higher ones. When testing the algorithms the outputs of higher levels were imitated.

The initial stage of investigation dealt with the leg-control algorithm in the simple case of regular gait of the walker moving along the regular standpoint sequence. The body moved with constant velocity. The imitation of the levels higher to the leg-control level was, in this case, rather simple.

The leg-control algorithm provided vertical legs adaptation to small-scale terrain roughness.

A special block was designed for synthesizing leg-tip motion during the swing phase in the case of complicated small-scale relief. The ordinates of the leg-tip trajectory (Fig. 3) were calculated as the sum of the ordinates of the convex envelope of the relief (dashed line in Fig. 3) and of the ordinates of a parabola with vertical axis. The parabola was chosen in such a way that itB ordinates were equal to zero both in the initial and final points. It was assumed that the horizontal component of the leg-tip velocity was constant during the whole swing phase.

For the second level of leg coordination - the algorithm for support scheduling with prescribed stability margin was designed in a general case for irregular standpoint sequence.

Two types of gait were investigated:

1. Tripod gait. Each of the two tripods consists of foreleg and hind leg of one size and of middle leg of another size. Three legs of the tripod swing simultaneously.Two tripods swing alternately. Fig. 4a illu - strates the adopted logics of calculating "up" and "down" times of the tripod in the case when all legs of the same side use the same standpoint sequence ("step-in-step" type of locomotion). The swing phase of the tripod coincides with the time interval when the projection of the centre of mass of the walker moves between two dashed lines inside the supporting triangle formed by the legs of the other tripod (Fig. 4a). This logics provides stability margin of prescribed value.

2. Wave gait.[18] The idea of this type of gait was taken from one of the enthomological papers by D. Wilson.14 The swing waves propagate along the legs of each side of the walker beginning from the hind legs. The hind legs of both sides start alternately.

Support scheduling logics is shown in Fig. 4b, The time interval between the start of the hind leg and the standing of the foreleg (wave propagation time) was calculated under condition of prescribed stability margin. Two equal intervals of simultaneous support of hind and middle legs, and of middle and front legs were subtracted from the wave propagation time. The rest of the time was devided among three legs proportional to their strides (the rule of constant leg-tip horizontal velocity.

It should be noted that in special case of regular standpoint sequence the gaits generated both by wave and by tripod algorithms may coincide. But in. general case of irregular standpoint sequence algorithms synthesize different gaits.

The designed algorithms of this level generated support schedule for both constant and variable velocity of the body in general case of curve route. The body rotation and the vertical component of body velocity might be taken into consideration.

On the third level two versions of standpoint planning algorithms were designed which were able to generate standpoint sequences for arbitrary curve route on the support surface with small-scale roughness. It was assumed that each point of the surface might be used as a standpoint.

Some algorithms were designed for generating special irregular standpoint sequences in case of overcoming obstacles.

The fourth-level algorithms formed body motion for curve route under the above mentioned condition relative to the support surface. Some cased of overcoming obstacles were considered.

Fig. 5 presents an example of the walker's locomotion along the curve route. The vehicle moved at first along the rectilinear segment AB. Then, at point B, it changed its route and began walking along the circle of the prescribed radius around the object located inside the circle (part BOB). At point B the walker continued its previous route (segment BD).

The problem of overcoming isolated obstacles of some types was investigated. An obstacle may be considered as an isolated one when it is located on the support surface all points of which might be used as standpoints. For some obstacles it appears undesirable or impossible to use points of the support surface in the vicinity of the obstacle due to geometrical restrictions associated with the neighbourhood of the obstacle.

Some types of isolated obstacles are shown in Fig. 6. One-parameter obstacle "cleft" (Fig. 6a) is functionally equivalent to the domain forbidden for standing the legs. There are no geometrical restrictions in the vicinity of the "cleft".

Two-parameter obstacle "boulder" (Fig. 6b), on the contrary, creates two restricted spots close to it. The spot before the boulder is undesirable because of the possibility of contacting the boulder in the support phase. The body of the boulder may make it impossible to stand leg tip in the spot behind the obstacle. It is permissible to stand legs of the walker on the boulder; it is even desirable.

The bottom of the three-parameter obstacle "pit" (Fig. 6c) may be used to stand legs on it except two spots near the walls.

It should be noted that "cleft", "boulder" and "pit" from the geometrical point of view may be regarded as a combination of more simple obstacles of the types "step-in" and "step-down" (Fig. 6d, e). If the longitudinal dimensions of the upper part of the boulder or these of the pit bottom are large enough, the boulder and the pit may be interpreted as two separate isolated obstacles of the "step" type. If the "steps" are positioned rather close one after another, there exists interference between them, and it is, apparently, more reasonable to treat such a combination as a special type of obstacle with its own special method of overcoming.

Some algorithms were designed for decision-making concerning the reasonable actions of the walker overcoming the obstacle. It was assumed that all necessary information about the type and geometrical parameters of the obstacle are available and may be used by decision-making algorithm.

As to the methods of overcoming obstacles, the basic principle was assumed that the higher level might be involved only in case of real need. For instance, if adaptation to email scale obstacles can be made by means of level "leg", this must be done. If this appears impossible, the special standpoint sequence and appropriate support schedule must be generated. If necessary, the special body motion has to be used.

The algorithms for overcoming the cleft-type obstacle were designed in greater details A special classification block estimated the situation: standpoint sequence parameters, cleft width and its position relative to the walker. Depending on the situation analysis results the following decisions about the regime could be made:

1. Nothing has to be changes.

2. It is necessary to make longer one stride before the cleft by changing the position of two standpoints and shifting them in such a way that one of them, the nearest to the cleft, would be positioned on the brink. The further development of standpoint sequence may be regular, as before the cleft.

3. It is necessary to position four standpoints on the brinks of the cleft (two *on* each brink) and to rearrange some other standpoints.

4. To apply regime 3 but to shift standpoints on the brink closer to the axis of the standpoint sequence.

5. The body of the walker must be lowered, and regime 4 must be applied.

The standpoint sequences in Fig. 7 correspond to regime 2, while those in Fig. 8 correspond to regimes 4 and 5.

The regimes 1-5 are listed in order of growth of their complicacy and their feasibilities. According to the basic principle the classification block tried to find out subsequently the possibility to use regimes 1-5, beginning from regime I, and adopted the first of them which provided successful overcoming the cleft.

Such an approach is evidently applicable to designing reasonable methods of overcoming other types of obstacles. It should be noted that for a pit rather deep, or for a boulder rather high, or for an obstacle like the one in Fig. 10 it may be necessary to tilt the body of the walker and change its pitch angle in an appropriate way as a function of time (Fig. 9, 10). It is evident that when analysing the obstacle, this regime, as the most complicated one, has to be tested in the last turn.

Some problems connected with measurements were investigated: measurement data processing, obstacle identification, measurement control.

It was assumed that the measurement system was able to estimate the distance between the fixed point of the vehicle and the point of intersection of the measuring beam and the support surface. The direction of the beam may be constant. When the vehicle walkes, the beam slides over the terrain and measures its profile. But this may be insufficient. The angle between the beam and horizon must be small enough for the vehicle could get terrain relief information beforehand and has possibility of planning its actions in a reasonable way. On the other hand, it is clear that for small beam-horizon angle rather long zones after obstacles are inaccessible to relief measurements. The increasing of the beam-horizon angle diminishes the inaccessible zones but diminishes simultaneously the distance between the vehicle and the measured points of the terrain.

Under the circumstances it was reasonable to control the beam direction for more effective use of measurement system. One of the adopted rules was as follows. All the time when it is possible, some "small" beam-horizon constant angle is used. This regime is used as long as the size of inaccessible zones is no more than a given value and each zone can be "overstepped", i.e. overcomed without placing any standpoint inside the zone. If not, the additional measurements must be carried out when approaching nearer to the obstacle. The measuring beam must be inclined steeper to horizon.

If measuring results indicate that it is Impossible to place standpoints inside the zone after the obstacle in an appropriate way, the further locomotion is excluded. If appropriate placing the standpoints is possible, the walker uses these points for standing its legs and walks on.

The investigation carried out confirmed that observing on the display screen the moving image of the vehicle walking on the terrain is a very effective method for testing the control algorithms and estimating their properties. The motion picture made from the CRT screen of the display unit gives an idea of the walker control algorithms effectiveness.

References:

I. Muybridge, E., Animals in Motion, Dover Co., New York, I957.

2. Бернштейн Н.А., Общая биомеханика, М., I926.

3. Бернштейн Н.А., О построении движений, Медгиз, I947.

4. Бернштейн Н.А., Очерки по физиологии движений и физиологии активности, М., I966.

5. Hildebrand, M., Symmetrical Gaits of Horses, Science, V. I50, N 3697, pp. 70I-708, November I956.

6. Hildebrand, M., Analysis of the Symmetrical Gaits of Tetrapods, Folia Biotheoretica, V. VI, I966, pp. I-22.

7. Томович, Р., О синтезе самодвижущихся автоматов, Автоматика и телемеханика, т. 26, № 2, I965.

8. Tomovic, R., McGhee, R.B., A Finite State Approach to the Synthesis of Bioengineering Control Systems, IEEE Transactions

on Human Factors in Electronics, V. HFE-7, N 2, pp. 65-69, June, I966.
9. McGhee, R.B., Finite State Control of Quadruped Locomotion, Simulation, V. 9, N 3, pp. I35-I40, September I967.
IO. McGhee, R.B., Some Finite State Aspects of Legged Locomotion, Mathematical Biosciences, pp. 67-85, February I968.
II. Frank, A.A., McGhee, R.B., Some Considerations Relating to the Design of Autopilots for Legged Vehicles, Journal of Terramechanics, V. 6, N I, pp. 23-35, I969.
I2. Frank, A.A., Automatic Control of Legged Locomotion Machines, Ph.D. Dissertation, University of Southern California, May I968.
I3. McGhee, R.B., Frank, A.A., Optimum Quadruped Creeping Gaits, University of Southern California, I969.
I4. Wilson, D.M., Insect Walking, Annual Review of Entomology, V. II, I966, pp.I03-I22.
I5. Wilson, D.M., Stepping Patterns in Tarantula Spiders, Journal of Experimental Biology, V. 47, I967, pp. I33-I5I.
I6. Wendler, G., The Coordination of Walking Movements in Arthropods, Symp. Soc. Exp. Biol., V. 20, I966, pp. 229-250.
I7. I7. Игнатьев М.Б., Кулаков Ф.М., Михайлов А.А., Михайлов В.В., Юревич Е.И., Алгоритмы управления адаптивной шагающей машиной, доклад, представленный на IУ Симпозиум ИФАК, Дубровник (Югославия), I97I.
I8. Охоцимский Д.Е., Платонов А.К., Боровин Г.К., Карпов И.И., Моделирование на ЦВМ движения шагающего аппарата. Известия Академии наук СССР, Техническая кибернетика, I972, вып. 3, стр. 47-59.
I9. Охоцимский Д.Е., Платонов А.К., Боровин Г.К., Карпов И.И., Лазутин Ю.М., Павловский В.Е., Ярошевский В.С. Алгоритмы управления движением шагающего аппарата. Институт прикладной математики АН СССР, Препринт № 63 за I972 год, Москва.

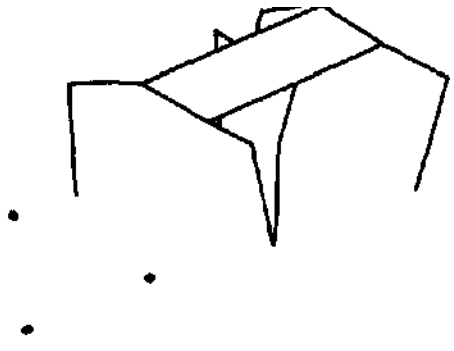


Fig. I. Schematic image of six-legged walker.

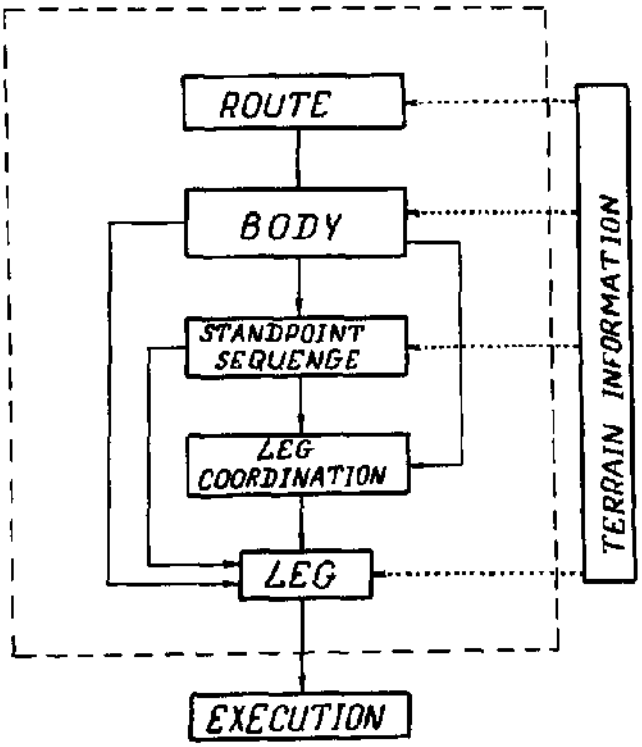


Fig. 2. Information flow in the control algorithm.

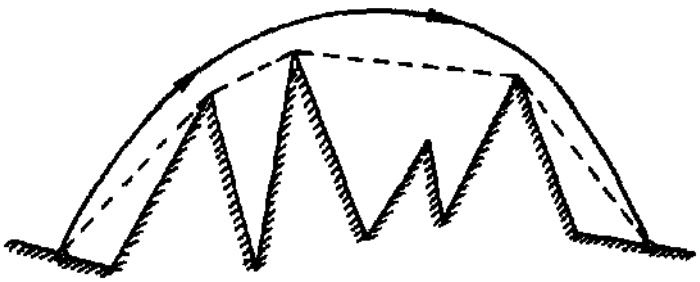


Fig. 3. Leg-tip trajectory in case of complicated relief.

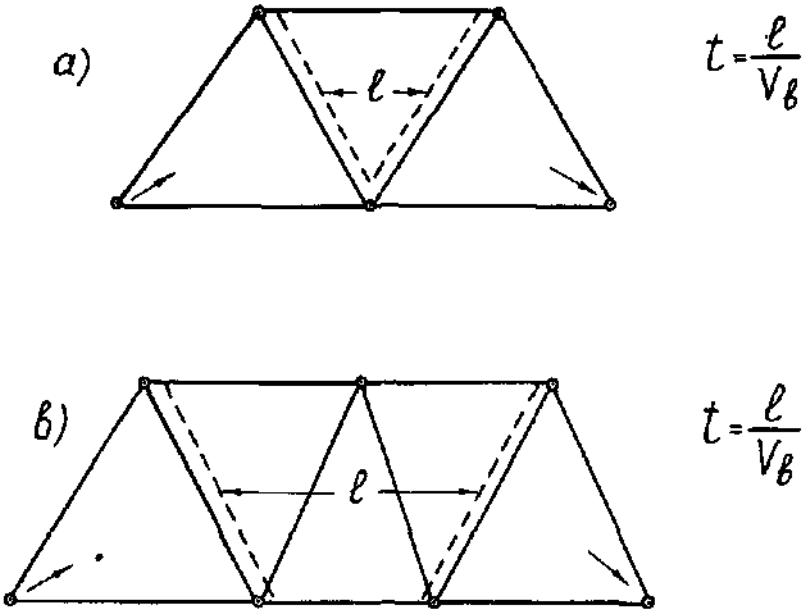


Fig. 4. Support scheduling logics:
a) tripod gait, b) wave gait.

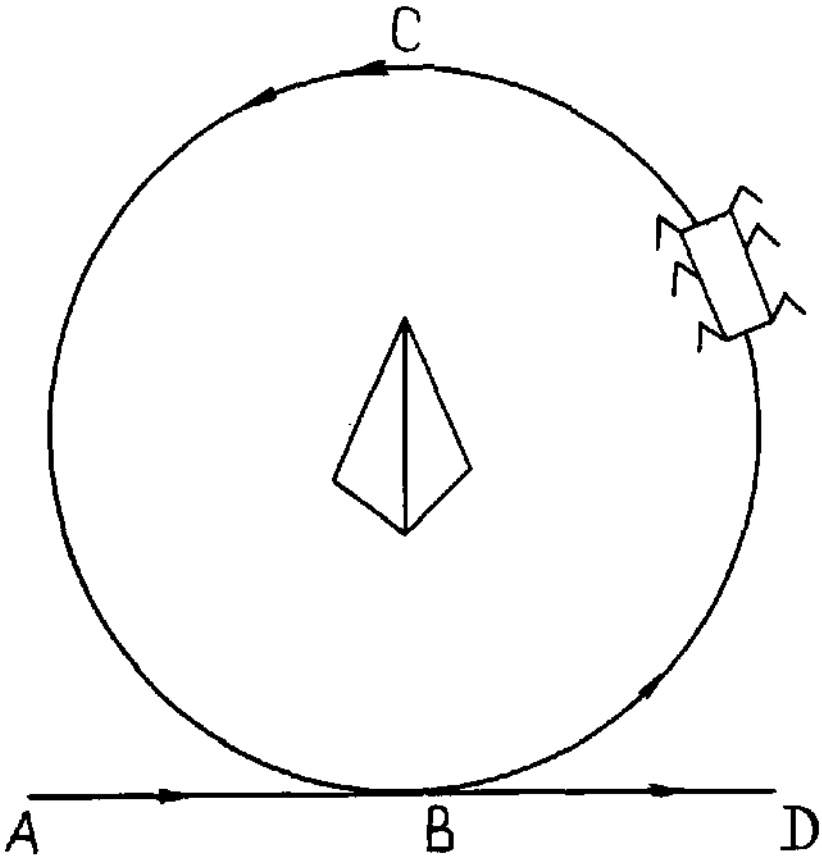


Fig. 5. An example of locomotion along the curve route.

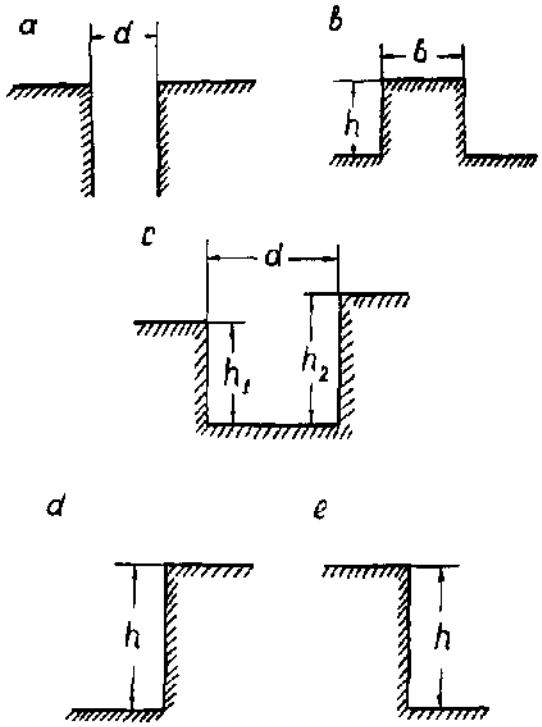


Fig. 6. Some types of isolated obstacles: a) cleft, b) boulder, c) pit, d) step-up, e) step-down.

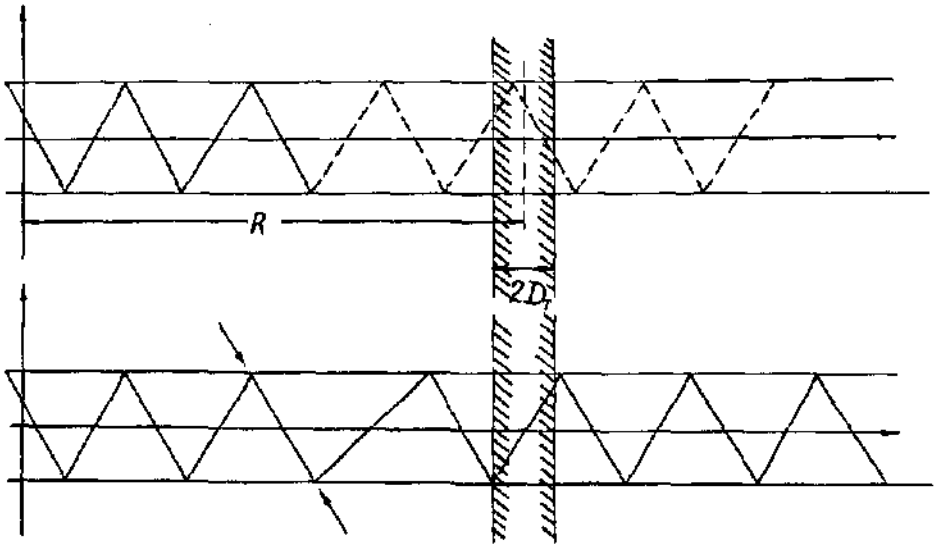


Fig. 7. Modification of standpoint sequence by shifting two standpoints.

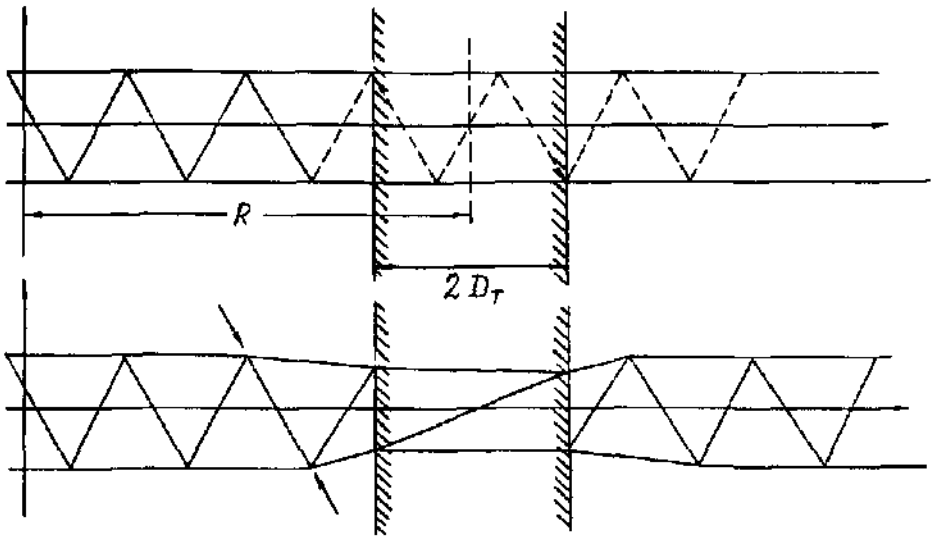


Fig. 8. Modification of standpoint sequence by diminishing its width.

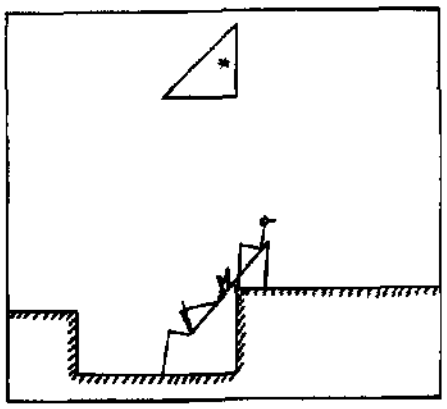


Fig. 9. Overcoming pit with body tilting.

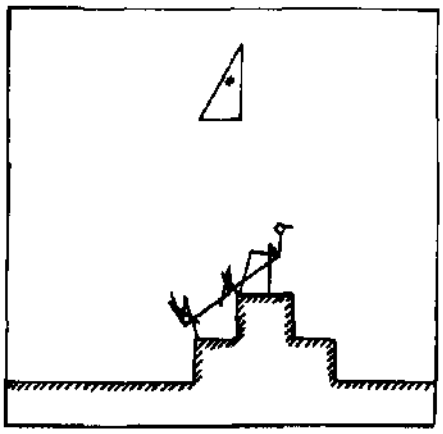


Fig. 10. Climbing over a complicated obstacle.