EIS-TR-79-11

HOW DOES A MOBILE ROBOT UNDERSTAND ITS WORLD?

by

Yutaka Kanayama

Jun'ichi Iijima

Shin'ichi Yuta

December 7, 1979

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS

UNIVERSITY OF TSUKUBA

EIS-TR-79-11

HOW DOES A MOBILE ROBOT UNDERSTAND ITS WORLD?

Yutaka Kanayama

Institute of Information
Sciences and Electronics
University of Tsukuba
Sakura-mura, Niihari-gun
Ibaraki-ken 305
Japan

Jun'ichi Iijima

Department of Computer Science
University of Electro-Communi-
cations
Chofu, Tokyo 182
Japan

Shin'ichi Yuta

Institute of Information
Sciences and Electronics
University of Tsukuba
Sakura-mura, Niihari-gun
Ibaraki-ken 305
Japan

Abstruct

A world-describing method for mobile robots is described. The fundamental idea is that the positional relation between any two points can be easily determined if we fix a reference direction in the world. First a skeleton which is the set of representative points of all objects is constructed. The skeleton is expressed in a form of a list, the R-list of elements, which describes relations between pairs of representative points. The relation between each representative point and the corresponding object is described in another list, the I-list. The current position of the robot in the world is dynamically described in a similar manner.

1. Introduction

A large number of intelligent mobile robots have been constructed for research into artificial intelligence at Stanford Research Institute [1], at the University of California, Berkeley [2, 3], at the California Institute of Technology [4, 5], at LAAS [6], at MIT and elsewhere. We also have constructed a mobile robot "Yamabiko" whose remarkable features are self-containedness and independence [7].

An interesting problem is "How should a mobile robot understand its two-dimentional world?" The problem can be divided into these two questions:

(1) How can the world be described?
(2) How can the position of the robot in its world be described?

A grid model has been adopted by many authors [1, 3, 5], in which a world is divided into many tiny squares and the presence or absence of an object in each square in expressed by 1 bit of information. This method is quite natural, but seems to have the following disadvantages:

(1) A large memory space is needed if a high precision is desired. In the case that the world is not bounded, this procedure needs almost unlimited working space.
(2) The structure of the data is uniform and structured description of the world is difficult.

Another method is proposed by G. Giralt and others [6]. They decompose the world into cellular areas, some of which include obstacles.

In this paper we describe a new method to describe the world; this method has more flexibility and efficiency than the grid model and the cellular area method.

2. Self-Contained Robots

The authors have implemented this method on a self-contained robot "Yamabiko III" whose features and functions are described in this section (See Figure 1). A robot which satisfies the following conditions is said to be self-contained:

(1) One that can move in an two-dimensional environment (mobility).
(2) One whose intelligence and power sources are contained in a closed body (self-containedness).
(3) One that can recognize its outer world and its position in the world (understanding its world).

Robots of this type interest us because they can work in a variety of environments with freedom in movement. They are useful, for example, in a street, in a fire, under water, in a nuclear reactor, or in a mine. Yamabiko III also works as a "robot cameraman" [8].

Another feature of Yamabiko is that it is a multiprocessing system. Since it works in the real world, operations by the brain, by the legs and by the eye have to be executed simultaneously. Therefore Yamabiko has three processors, for the brain, the legs and the eye.

The size of Yamabiko III is 35 × 40 × 52 cm; the weight is about 15 Kg; the processors are 6802's. It has 2KB ROM and 16KB RAM. It has a supersonic eye, a TV camera eye, direct current motor legs, a floating arithmetic and NiCd storage batteries. Figure 2 shows its block diagram. Yamabiko can get the distance and the angle to an object with its supersonic eye, can move forward or backward, and can revolve around its center with the legs. Thus the robot has the functions that are essential for understanding the world.

3. Describing the World

In general, an environment of a robot consists of many objects,

for example, desks, columns, walls, chairs, blocks, rooms and hallways. (See Figure 3(a).) Let us restrict the world of robots to be two dimensional. We can assume that its configuration does not vary in a short period. Hence we call the descriprion of the world, excluding the robot itself static data. Besides this information we need dynamic data which tells us the current position of the robot in its world. This is not time-invariant because the robot may change its position.

One of the key points of our way of description is that a reference axis is determined and every data segment is expressed by means of this direction. This is sometimes called north, but it does not mean that this is identical with magnetic north.

3.1 Static Data

Consider the world of Figure 3(a). First we determine a representative point for each object. For example, rectangular and circular objects are represented by their centers, and a wall by one of the endpoints. Assume each object has an identifier P_i and each representative point is designated by the identifier of the object it represents. The set of all representative points in a world is called a skeleton. The skeleton of the world of Figure 3(a) is shown in Figure 3(b).

Second, we make two lists of data:

(1) An R-list which describes the whole structure of the skeleton and is a list of R-elements.(#)

(2) An I-list is a list of I-elements; each I-element refers to a representative point and describes the object which the point represents.

(#) R and I are abbreviations for "relative" and "individual" respectively.

An R-element describes the relative relation of two representative points. For example, the skeleton shown in Figure 4 is described by three R-elements corresponding to three pairs of points (P_1, P_2), (P_2, P_3) and (P_1, P_4). A reference axis is given in this world and is fixed. Vector P_1P_2 is expressed by the distance l_1 and the angle t_1 with the reference axis. They form an R-element (P_1, P_2, l_1, t_1). Thus the R-list of this skeleton is $((P_1, P_2, t_1, l_1), (P_2, P_3, t_2, l_2), (P_1, P_4, t_3, l_3))$.

Clearly an R-segment (P, P', t, l) is equivalent to $(P', P, t+\pi, l)$. All pairs of representative points are connected in the R-list; for any pair (P, P') there exists a sequence of R-elements $(P, Q_1, t_0, l_0), (Q_1, Q_2, t_1, l_1), \cdots, (Q_n, P', t_n, l_n)$ in the R-list. This list may be redundant. It means that there may exist a loop in R-list. In the case of a world consisting of rectangles, if its reference axis is properly chosen, most of the third terms in R-elements should be 0, $\pm\frac{\pi}{2}$ or $\pm\pi$.

Static data can have a hierarchical structure. Consider the world depicted in Figure 5(a). This world can be described in two levels. The top level describes only three rooms and a hallway, as in Figure 5(b). Low level data are divided into three parts, each of which describes the positions of furniture in one of the rooms. This type of structured description permits efficient processing in many of the problems stated in Section 4.

Some examples of I-elements are

$(P_1$, CIRCLE, 50)
$(P_2$, SQUARE, 100, 0)
$(P_3$, WALL, 250, $-\frac{\pi}{4})$,

where the first terms are identifiers, the second types, the third sizes and the fourth directions of the objects.

3.2 Dynamic Data

If a robot cannot move, all it needs is the static data of its environment because it knows the position of its eye(s) in the map and it can guess the position of what it sees. If the robot is mobile, however, it must know its position in its world and have some formal means to represent relative positions in the static world.

Let us define its position by means of the skeleton. It is assumed that the robot is in a position from which it can look at least two of the representative points in its world. In Figure 6, the robot Y can see P_1 and P_2, among others, with angles s_1, s_2 and lengths m_1, m_2, and it is heading in direction s. Thus the dynamic data are $(s, (P_1, s_1, m_1), (P_2, s_2, m_2), \ldots, (P_r, s_r, m_r))$, where r is the number of representative points which the robot can look at. Clearly, the position of Y is uniquely determined from the data above.

These values have to be updated as the robot changes its position or direction. The rules are as follows:

(1) In case Y rotates through an angle s',
    $s := s + s'$.
    If Y rotates clockwise, s' is negative.
(2) In case Y moves forward by m, then
    for i := 1 to r do begin $m_i' := \sqrt{m^2 + m_i^2 - 2mm_i\cos(s-s_i)}$;

$$s_i := s_i + \cos^{-1}\left(\frac{m_i^2+m_i'^2-m^2}{2m_im_i'}\right)$$

$$m_i := m_i'$$

    end

The Second Cosine Formula in trigenometry gives the evaluation of new m_i and s_i (See Figure 7).

When a mobile robot walks, it always updates the dynamic data. One problem is that its evaluation of walking distance and

rotation angles inevitally has some errors. Therefore it also has to estimate the error, and when the value exceeds a permissible threshold value, it must look around and determine exact m_i and s_i values.

It is apparent that the reference axis plays an extremely important role in this method. If the robot is given precise static data and a good eye system, a mobile robot can easily know the reference axis at any point in its world. Some hardware to tell the magnetic north, however, might be useful.

This method resembles the way people understand our world. When we walk or drive in a town we know the direction in which we are heading. When we have lost this knowledge of direction, we have lost our way.

4. Algorithms

Since self-contained robots are special purpose computers, they operate with programs. We will demonstrate some algorithms to manipulate the data base described above. The simplicity of these programs shows the validity of this method of describing the world.

4.1 Evaluation of distance between representative points P_1 and P_2.

If an R-element (P_1, P_2, t, l) is in the R-list, the answer l is immediate. An element (P_2, P_1, t, l) is regarded as (P_1, P_2, $t+\pi$, l) as mentioned earlier.

If R-elements (P_1, P_3, t_1, l_1) and (P_3, P_2, t_2, l_2) are in the R-list, a new R-element (P_1, P_2, t, l) is generated and added, where t and l are evaluated as follows (See Figure 8):

$$l = \sqrt{l_1^2 + l_2^2 - 2l_1 l_2 \cos(\pi - t_2 + t_1)}$$

$$= \sqrt{l_1^2 + l_2^2 + 2l_1 l_2 \cos(t_1 - t_2)}$$

$$t = t_1 + \cos^{-1}\left(\frac{l^2 + l_1^2 - l_2^2}{2ll_1}\right)$$

In the case that $(P_1, Q_1, \cdots)$, $(Q_1, Q_2, \cdots)$, $\cdots$, $(Q_n, P_2, \cdots)$ are in the R-list, the evaluation is successively made using the above process. A path from P_1 to P_2 can always be generated in this way because the P-list is connected.

4.2 Finding a route to a destination

Suppose the destination is one of the representative points. Otherwise you can add an R-element to the R-list and can regard it as a representative point. Let P_1 be a point that the robot is looking at and P_2 be the destination. Then the distance and the direction of vector YP_2 can be obtained in a similar way to 4.1, where Y denotes the position of the robot Yamabiko.

On the route YP_2, however, there may be some obstacles. We can avoid collisions by finding a safe path in a way similar to one suggested in [5], [9], and will omit details here.

4.3 Generating Static Data

Suppose a robot has no data about its world. Then it has to initially discover the whole map of the world. We assume the following conditions over the world W:

(1) There exist no walls in the world.
(2) The robot can pass between any two objects.
(3) There are no isolated objects; for any object O_1 in W, there exist two objects O_2, O_3 and a position Q such that the robot at Q can look at O_1, O_2 and O_3.

(4) At the initial configuration, there exists at least one object in sight of the robot.

A simplified algorithm for this problem is as follows:

R-list: = ϕ; look around;
repeat begin R-list: = R-list $\cup$ {new R-element}; walk along the convex path around the current world and seek objects
end
until no new objects are found.

Acknowdedgment

The authors thank Professor J. W. Higgins for his useful suggestions and critical reading of the manuscript.

References

[1] Nilsson, N. J., A Mobile Automation: An Application of Artificial Intelligence Techniques, Proc. of 1st IJCAI, pp. 509-520, 1969.

[2] Smith, M. H. and L. S. Coles, Design a Low Cost General Purpose Robot, Proc. of 3rd IJCAI, pp. 324-335, 1973 .

[3] Coles, L. S., A. M. Robb, P. L. Sinclair, M. H. Smith and R. R. Sobek, Decision Analysis for an Experimental Robot with an Unreliable Sensors, Proc. of 4th IJCAI, pp. 749-757, 1975.

[4] Dobrotin, B. and R. Lewis, A Practical Manipulator System, Proc. of 5th IJCAI, pp. 723-732, 1979.

[5] Thompson, A. M., The Navigation System of the JPL Robot, Proc. of 5th IJCAI, pp. 749-757, 1977.

[6] Giralt, G., R. Sobek and R. Chatila, A Multi-Level Planning and Navigation System for a Mobile Robot, Proc. of 6th IJCAI, pp. 335-337, 1979.

[7] Kanayama, Y., J. Iijima, H. Ochiai, H. Watarai and K. Ohkawa, A Self-Contained Robot "Yamabiko", Proc. of 3rd UJCC, pp. 246-250, 1978.

[8] Iijima, J., S. Yuta and Y. Kanayama, Robot Cameraman, Proc. 10th Image Engineering Conference, pp. 155-158, 1979 (in Japanese).

[9] Lozano-Perez, T. and M. A. Wesley, An Algorithm for Planning Collision-Free Paths Amongst Polyhedral Obstacles, Research Report of IBM T. J. Watson Research Center, RC 7171, 1978.

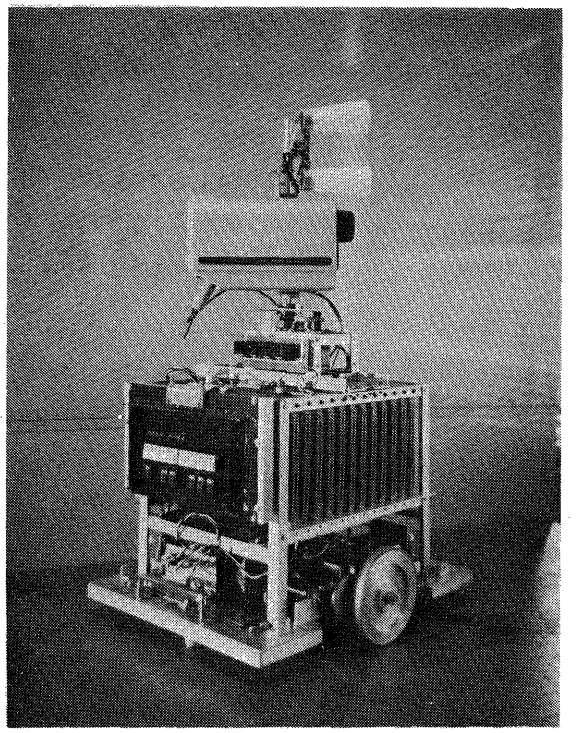


Figure. 1 Yamabiko III

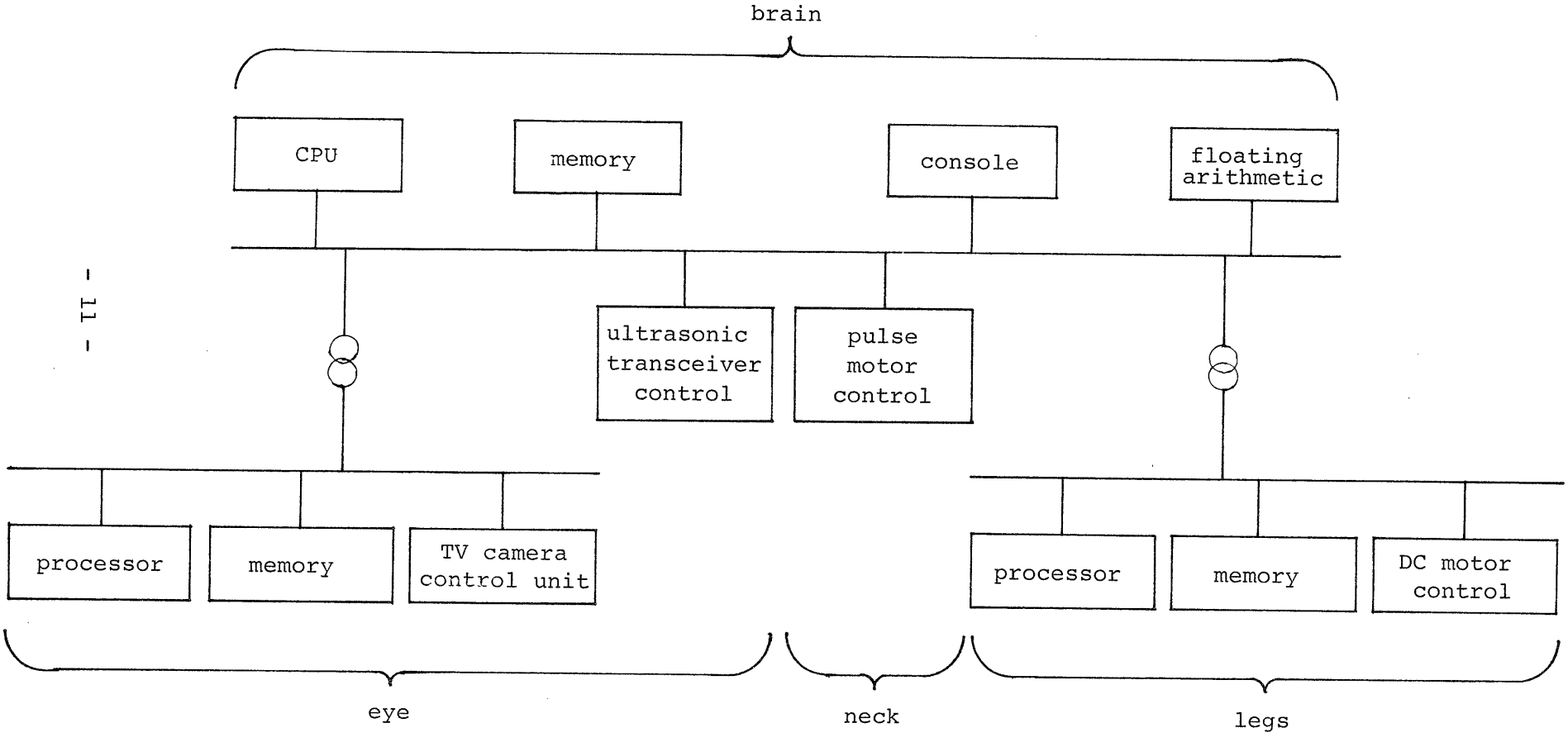
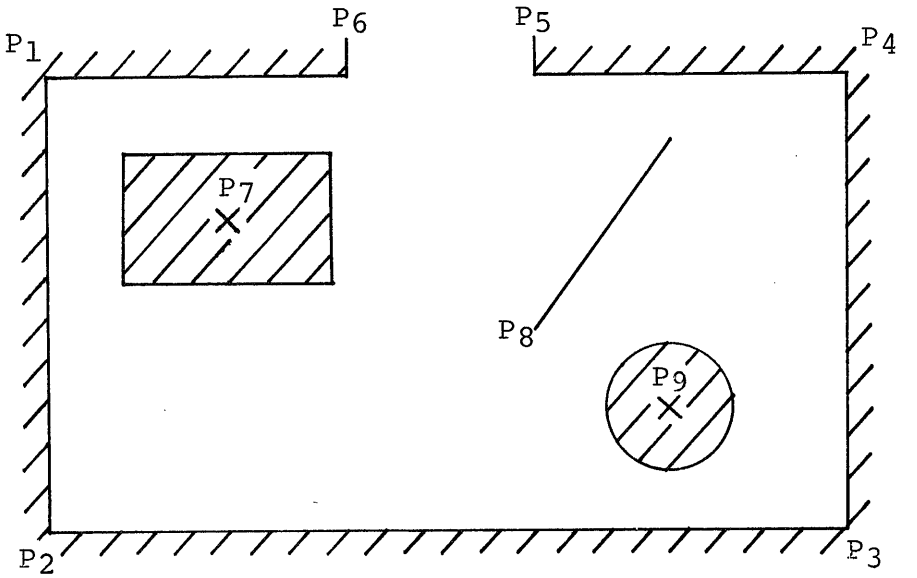
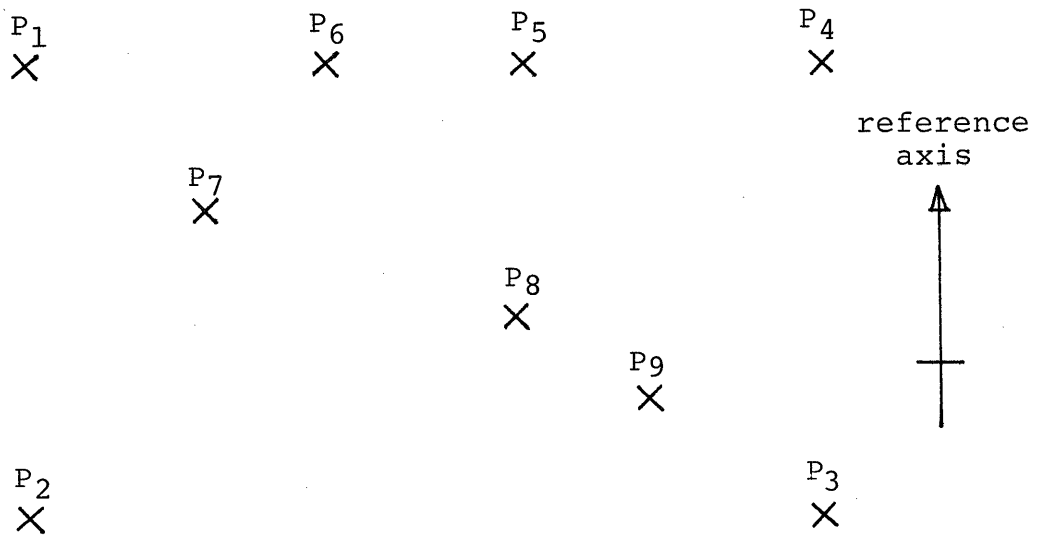


Figure 2. Configuration of Yamabiko III



(a) World W



(b) The skeleton of the world W

Figure 3. A simple world

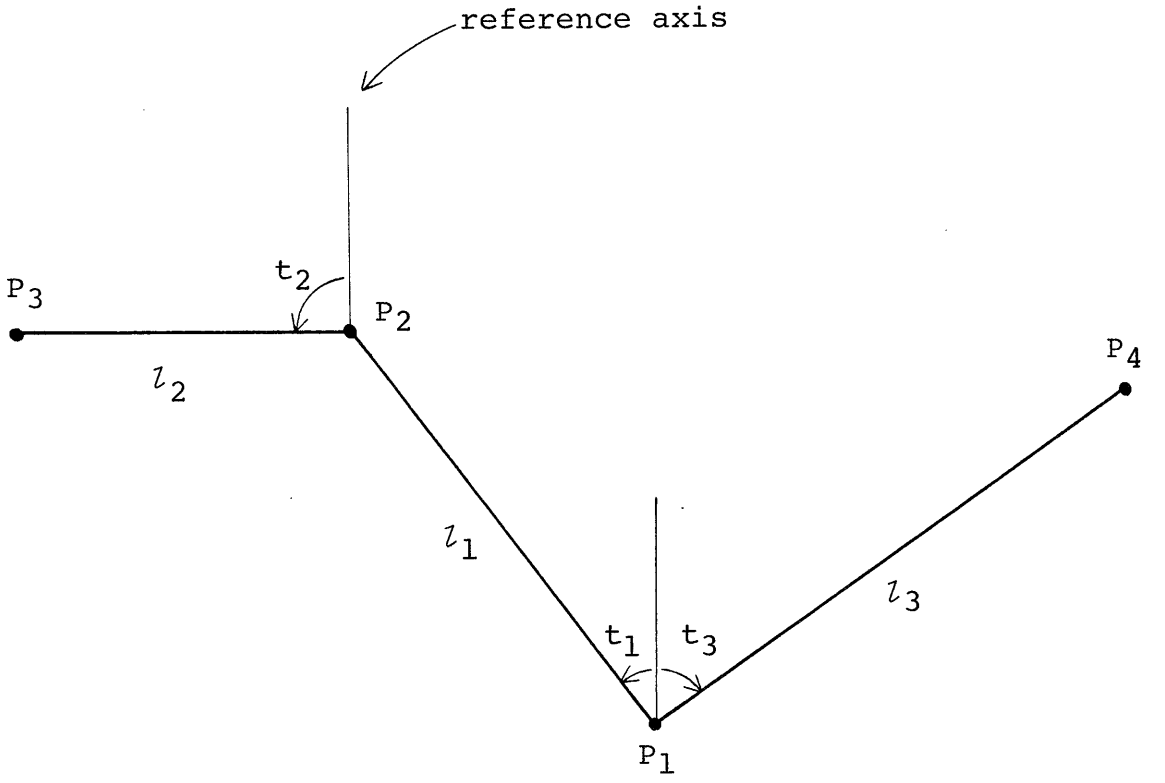
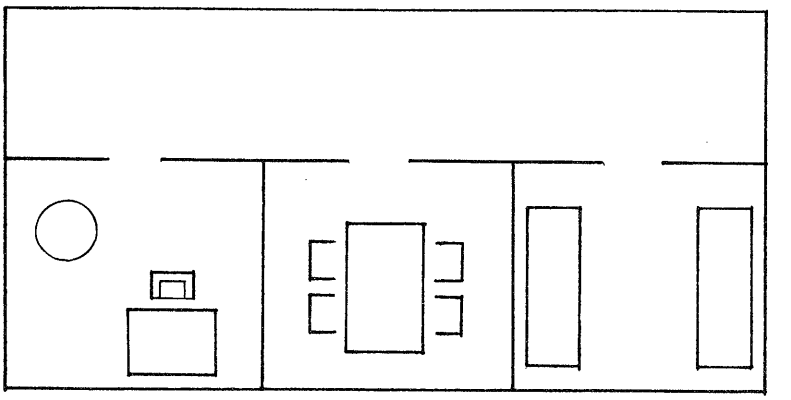
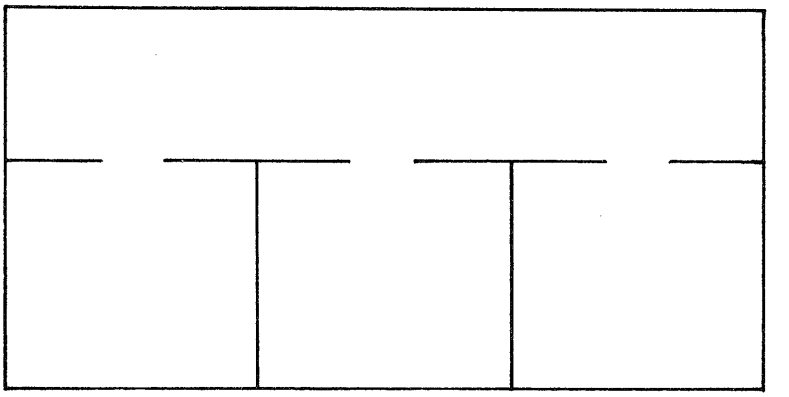


Figure 4. Describing a skeleton



(a) Original world



(b) Top level description

Figure 5. Structured Description

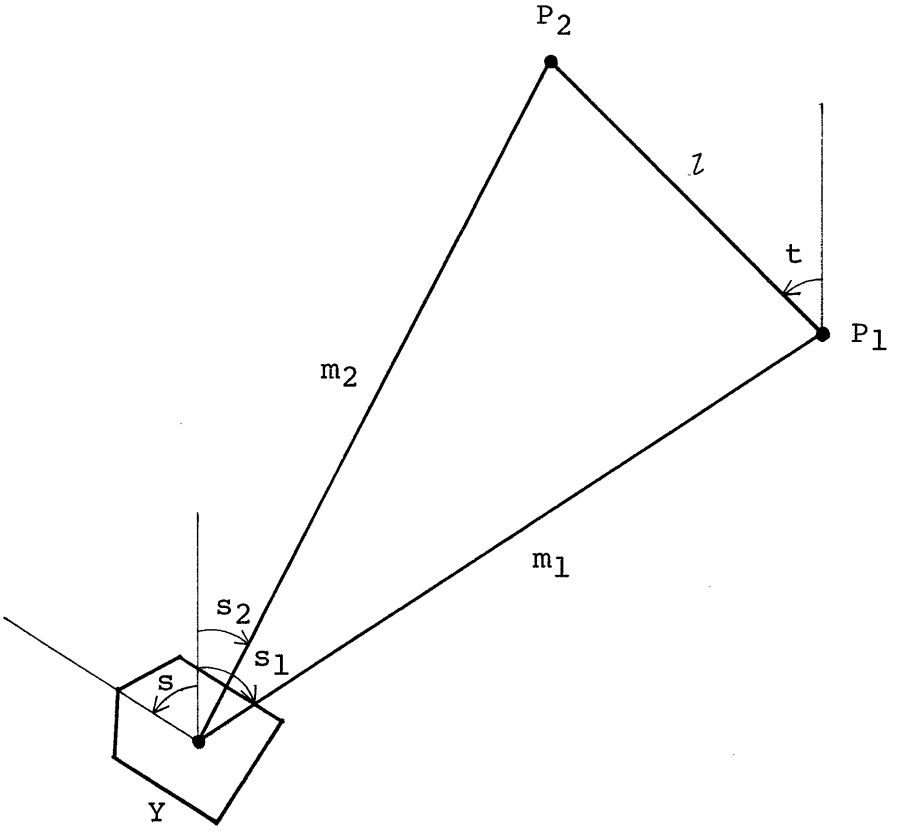


Figure 6. Dynamic data

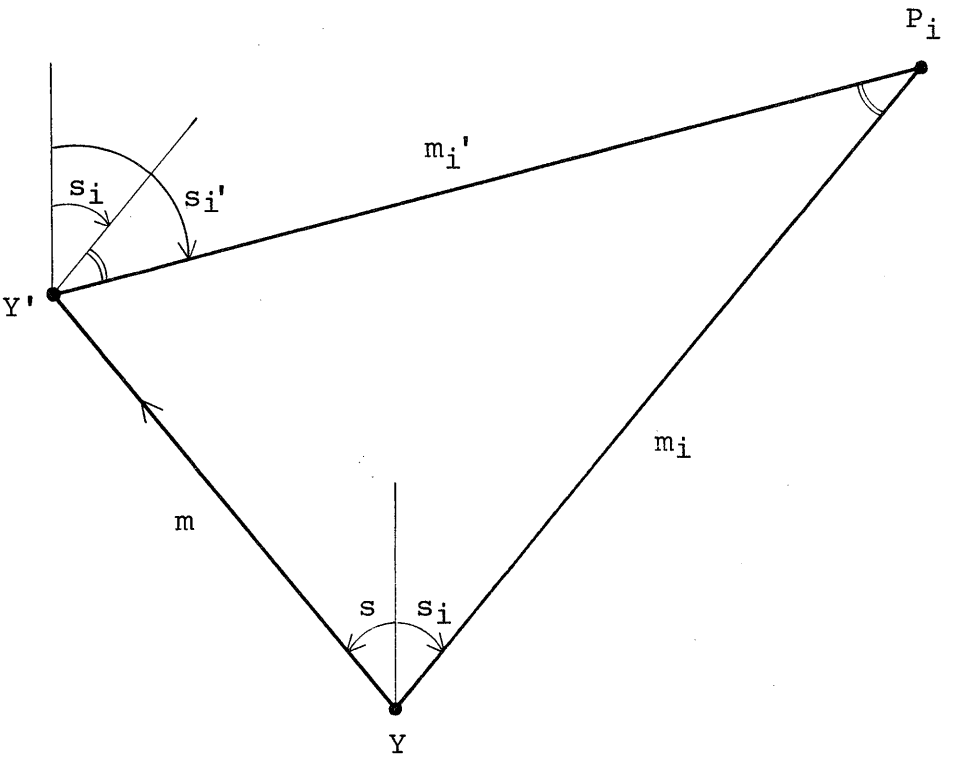


Figure 7. Updating m_i and s_i

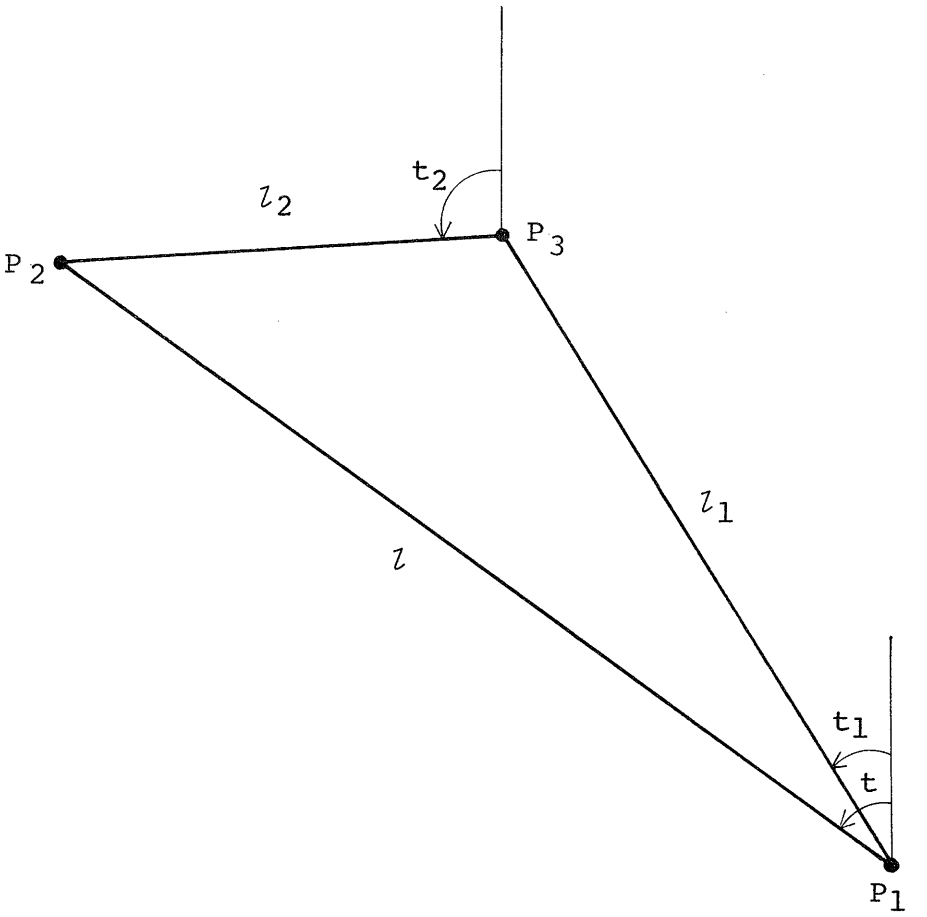


Figure 8. Evaluating distance