SPIE 1986 [illegible]

Hierarchical robot control in a multisensor environment

Bir Bhanu, Nils Thune, Jih Kun Lee, and Mari Thune

Department of Computer Science, University of Utah
Salt Lake City, Utah 84112

Abstract

Automatic recognition, inspection, manipulation and assembly of objects will be a common denominator in most of tomorrow's highly automated factories. These tasks will be handled by intelligent computer controlled robots with multisensor capabilities which contribute to desired flexibility and adaptability. The control of a robot in such a multisensor environment becomes of crucial importance as the complexity of the problem grows exponentially with the number of sensors, tasks, commands and objects.

In this paper we present an approach which uses CAD (Computer-Aided Design) based geometric and functional models of objects together with action oriented neuroschemas to recognize and manipulate objects by a robot in a multisensor environment. The hierarchical robot control system is being implemented on a BBN Butterfly multi processor.

Index terms: CAD, Hierarchical Control, Hypothesis Generation and Verification, Parallel Processing, Schemas

Introduction

Factories of the future will make extensive use of automatic recognition, inspection, manipulation and assembly of objects, which calls for intelligent computer controlled robots. These tasks are complex to control and the complexity increases rapidly as the tasks get more involved. In addition, growth in the number of sensors and the types of sensors a robot is equipped with also contributes to increasing complexity resulting in many times more data to process than in a single sensor system. Consequently, a control system needs to be flexible, adaptable, and able to learn from past experience.

Many of the tasks in automated factories need to be done in real time. Therefore, it is natural to bring parallel processing into the picture enabling considerable speedup in the execution of tasks compared to sequential processing on a conventional processor. For example, low level image processing, involving large amounts of data, is possible to accomplish in real time. Furthermore, the control can also experience speedup by running independent parts in parallel.

Knowledge about the intelligent aspects of a control system can be drawn from the neurosciences where studies of the brain and the nervous system indicate how basic building blocks are organized and how they are capable of distributed processing. Using this knowledge we present an approach to robot control called a *hierarchical* control system which uses *action oriented schemas*. Such schemas are termed *neuroschemas* because of their similarity to *neurons*[1], which are the basic building blocks of the brain, and *schemas*[2,3,4] which are a basic kind of representation. These neuroschemas are the basic building blocks of the control system.

The purpose of the control system is to achieve high level goals, specified by a user, through planning and action. The goals which can be achieved depend upon the system's knowledge base, and are restricted by existing programs which are compiled into the system. Currently, the system's knowledge is sufficient to locate and recognize polyhedral three-dimensional objects in range images. The system also has the capability to learn by experience and by user interface.

In addition to the robot control system, we also present a new approach to three-dimensional object recognition. It uses CAD based geometric models of the objects together with a *feature indexed hypothesis* strategy to recognize 3-D objects in range images.

Most of the earlier work using feature indexed hypotheses for recognition has been limited to two-dimensional objects[5,6]. Three-dimensional cases are much more complex. One needs to consider the three-dimensional model acquisition and complicated orientation problems. Models of the objects can be in the form of pictures from different viewing angles. However, it is expensive and not practical to store more than a thousand pictures taken from different viewing angles[7]. CAD based geometric models provide a good solution. The key to recognizing three-dimensional objects using feature indexed hypotheses is to find the correct transformation matrix for the model. Using this matrix, the model can be transformed to the location in the range image which corresponds to the position of the object being matched. The feature indexed hypotheses method is then used to determine if the model and the object actually match.

An early version of the control system and the 3-D recognition algorithm is implemented on a VAX 11/780 and the control system is currently being transported to a BBN Butterfly parallel processor with a 19 node configuration.

Aspects of robot control

To make an "intelligent" control system, it is natural to borrow ideas from the most intelligent system we know, the human brain and the nervous system. Studies of the brain indicate some important and basic factors of our intelligence[8], which are also important for a robot control system:

1. The brain is made of basic building blocks, called neurons.
2. The brain is structured in a hierarchical manner.
3. The brain operates in parallel.

Basic building blocks (neurons) gather, process, and produce information which is used to make intelligent decisions about tasks to be done. Even though there are many categories of neurons, like motor neurons and sensory neurons, almost all of them have the same general structure: many *dendrites* carry the input to the *cell body* where the information is processed. A single *axon* carries the output to other neurons in the nervous system. All the dendrites and axons are organized in a complex network which probably is the key to our intelligence, since it provides the necessary links between parts of our brain[8,9].

It is believed that the neurons, with their complex network of interconnections, are organized in a hierarchical fashion[8]. Main "commands" are issued at the top, and are split into subgoals as they propagate down the hierarchy. Neurons receive their "commands" from neurons at a higher level and also accept feedback from the neurons at lower levels.

In addition to the hierarchical organization, it is clear that the brain makes extensive use of parallelism in carrying out its tasks[8,9,10,11]. When an action is initiated, many neurons are involved in receiving input, processing it and propagating the result. It is known that the brain is quite slow compared to digital computers, being able to carry out only about 100 serial time steps per second[9,11]. Since normal reaction time for a human being is approximately 0.5 to 1.0 seconds, and the tasks which the brain is carrying out during this time often requires a substantially higher number of computations than 100, it must be concluded that parallelism is essential for our ability to react as fast as we do.

In developing an intelligent control system for robots, it is desirable to include the three important aspects of the brain already discussed. Using knowledge about the neurons in the brain, and ideas from schemas[2,3,4], the neuroschemas have emerged as the basic building blocks of the control system. These are organized in a hierarchical manner for each goal the system can achieve. Hence, three of the main aspects of the brain have their analogs in the robot control system: basic building blocks, hierarchical organization, and parallel processing.

The basic building blocks of the robot control system

Figure 1 shows the three main parts included in the control system: the hierarchical control structure, the global knowledge base, and the global data base. The hierarchical control system receives all its input from the user or from multisensors, and achieves goals according to the information found in the knowledge base. In case the information does not already reside in the knowledge base and a new experience is gained, the knowledge base is updated with this new information. The data base is important for recognition of objects because all models known to the system are stored here. It can also be updated.

The hierarchical control structure

The purpose of the robot control system is to achieve a main goal issued by the user. To achieve such a goal, the system uses a structure in the form of a tree organized as a hierarchy which controls how the goal will be obtained. This tree, which exists while the main goal is being obtained, is an analog to the short term memory in the human brain[12,13]. Figure 2 shows an example on how the tree grows both in the serial version and the parallel version, respectively. The G's in Figure 2 denote Goal, the S's Subgoal, and the P's denote the processor on which the goal is being controlled/processed. The rectangular boxes denote that a (sub)goal is active. If the boxes are filled, it means the goals have been achieved. The oval boxes denote (sub)goals to be obtained. The root of the tree is created when a main goal is given to the system. The control flows in a top-down manner, like the backward chaining in a production system[12].

The basic unit in the control structure is the *neuroschema*. Each node in the trees in Figure 2 is controlled by a neuroshcema. They can only be activated by other neuroschemas, which are already active, so the system is therefore action oriented and goal driven. In addition to activate new neuroschemas, the parent neuroschemas provide decision making (using probability measures) at their respective nodes in the hierarchy, thus deriving a sequence of steps. This planning is based upon previous experience, and achieves a goal with least risk and cost.

The neuroschema consists of three sections: an *Activation Section*, an *Event Section*, and a *Learning Section*. Each neuroschema is actually a procedure, as shown in Figure 3 (encoded in the C-language), simulating the general functions of a neuron by receiving input, processing it, and producing output. Depending on the input received, different processes are activated by the neuroschema to produce an output. A neuron can take on any number of inputs and produces an output. The same is true for the neuroschema. In addition, any type of input and output can flow through it, making it flexible. The neuroschema makes use of the knowledge it has been activated with to determine how to process both the input and the output.

The Activation Section checks the status of the goal - if it has been achieved or not - and indicates the result. This section thereby functions similar to the electric potentials which can be measured in a neuron when it is active or inactive.

The Event Section determines how the goal is to be achieved. This corresponds to the function a neuron carries out on its input to produce its output. The neuroschema, which is activated with a goal, obtains this goal by achieving its subgoals. The subgoals are obtained by either activating a new neuroschema at the level below in the hierarchy, or if the subgoal is a program it is executed. One particular goal can sometimes be obtained by achieving either one of several different alternative subgoals (Sg) (an OR-branch: achieve one subgoal OR the other). Using statistics about previous successes in obtaining different goals, the goal with the highest probability (P) is chosen. Success (S) means that the (sub)goal has been obtained with satisfactory output. Failure indicates that the (sub)goal was not achieved. The probability values are calculated as follows:

$$P(S) = 0.5 \tag{1}$$

It is an unbiased value if the (sub)goal has never been achieved before. If it has been achieved before, but never with this particular input

$$P(S|Sg) = \frac{Successes}{Trials} \tag{2}$$

Here, the numerator is the number of times the subgoal has been achieved with any kind of input, and the denominator is the total number of trials (failures + successes). If the subgoal has been achieved with exactly this input (I) before

$$P(S|Sg \cap I) = \frac{Successes}{Trials} \tag{3}$$

In this case, the numerator is the total number of successes, so far, for obtaining the subgoal with exactly this input. The denominator is the total number of trials made with this input. To enable calculation of these probabilities, the control system records the number of trials, successes and failures. For every trial, the number of successes and failures are updated and stored in the knowledge base.

The Learning Section is activated when no information about how to obtain the goal can be found in the knowledge base, indicated by the Activation Section. It is also activated if none of the alternative subgoals of the main goal can be obtained. In both cases the Learning Section has to obtain the necessary information from the user.

The structure of the neuroschema resembles the schema of Iberall and Lyon, and Overton[2,3,4]. In contrast to their schemas, however, which have preconstructed plans for achieving a goal, the neuroschema is a control environment which can be activated with any plans for goal achievement from the knowledge base. Another difference is found in the approach to learning, which in the case of schemas, is accomplished by instantiating schemas which better fit a new situation. When our system is learning, a new neuroschema is not instantiated; instead, new information (in the form of probability measures and alternative subgoals) about how to achieve the goal is used to update the knowledge base, and it is used to achieve the goal next time.

The knowledge base

The knowledge base is the *world model* of the control system and can be viewed as an analog to the *long term memory*[3,12] of the human brain. *Short term memory*[12], in contrast, uses information found in long term memory to obtain a goal and then disappears. The knowledge base includes programs, core system goals and any knowledge obtained as a result of learning. Basically, we are using a world model where knowledge about how to obtain a goal is stored as procedures (procedural knowledge). Also, with each goal in the knowledge base, an indication of how probable it is to obtain this goal, is stored.

The programs are a part of the knowledge base. They are the core of the system, enabling it to take actions on the environment. These actions takes place at the lowest level of any branch of the control hierarchy. At the moment, programs for recognizing polyhedral objects exist in the knowledge base. However, there is no limitation on the number or kind of programs which it can consist of. Figure 4 shows the structure where the programs are shown as goals with no subgoals.

Goals which are contained in the knowledge base are usually not programs. Instead, they contain information about how to further divide themselves into subgoals (see Figure 4). At the lowest level in the hierarchy, however, the goals are programs which result in action. When the system is started up, there exist some predefined goals, which together with the programs constitute the knowledge base. Figure 5 shows an example, where one subgoal is further divided into new subgoals and some subgoals are programs.

The data base is described in a later section.

Control of multisensors

Intelligent robots in automated environments are required to be equipped with many sensors of multiple types. TV Cameras, range finders, tactile, force , and torque sensors are important in tasks such as recognition, assembly, inspection, and manipulation of objects[14]. Integration of multisensors in a robot control system is concerned with both control of the sensors and efficient and effective utilization of diverse information from multiple sensors to achieve the goal.

One of the key advantages with the control system we have presented is that there is no limit on the number or type of programs being incorporated into the system. This implies that any number and type of sensors can be controlled. In our work, a multisensor environment consist of two types of sensors: a TV camera providing intensity data, and a laser range finder providing range data. The objective is to recognize 3-D objects and manipulate them. Sensor fusion will enable more accurate recognition, since the data obtained from the two types of sensors complement each other.

Parallelism

Our robot control system (serial version) is being transported from a VAX 11/780 to a BBN Butterfly Parallel Processor (parallel version). It is a multiple instruction, multiple data (MIMD) machine, and is connected to a host machine which in our case is a VAX 11/780.

Each processor runs one copy of the Chrysalis operating system . This operating system is written in C and supports communication and synchronization between processes running on different processors. This is done by means of dual queues which pass messages between these processes, and an event mechanism (similar to *signals* in UNIX). Chrysalis does not provide automatic resource allocation, load balancing or process migration, however[11]. Each user-developed program has to set up the data, create all necessary processes, and decide on which node(s) they will run. In contrast to Chrysalis, the Uniform System approach to programming the Butterfly provides the user with easier resource management. The Uniform System is built on top of Chrysalis and consists of several subroutines which take care of, for example, allocation of memory and processors, and generation of new tasks (processes). The user does not allocate memory space or processors explicitly, since the Uniform System takes care of distribution of tasks on processors and provides special memory allocation routines. The Uniform System is especially suitable for homogeneous problems often found in low level computer vision.

Parallelism in the system can be divided into two categories according to the complexity involved. First, there is the most complex task of implementing the actual control and high level programs, which involves running different programs on different processors, requiring communication and synchronization between various processes. This is done using Chrysalis. Second, it is the relatively simpler task of implementing low level computer vision programs with inherent parallelism and no complex control aspects, making the Uniform System the best programming approach.

The first, and more complex category of hierarchical control uses the hierarchically organized tree, described earlier (see Figure 2), to decide if subgoals can be started up in parallel. This includes situations where different alternative subgoals can achieve the same goal with approximately the same probability of success. In addition, subgoals can be started up in parallel when all needed inputs are provided, and any use of end effectors will not result in conflicts. High level programs are also started up in parallel if all the inputs are available and no conflict with end effectors will arise.

One of the advantages of using multiple processors to execute alternative ways of achieving goals simultaneously is to prevent time delay due to an alternative's failure to obtain the goal. If one of the alternatives fails, or the results are not satisfactory, the result of another can be used instead. If the alternatives were not executed in parallel and the first alternative failed, it would take longer to achieve a goal; the next alternative would be executed only after the first had failed.

When the hierarchical control allows parallelism, the parent neuroschema has to check if there are any processors available on which to start up "child processes" (new neuroschemas). If this is the case, the parent also has to set up all the necessary data on the respective processors before it can initiate any child processes. Parent and child will communicate using a dual queue, on which messages are posted. When a child is done, a special message informs the parent[13]. If no processor is available, the child process must be started up on the same processor as the parent is running on. Moreover, if there is only one way of obtaining a goal, the child is started up on the same processor node as the parent is running on, since there is no alternative which can be started up in parallel. In the last case, when there is only one way of achieving the goal, the child's work can be distributed on several processors, however.

The second category of parallelism in the system involves programs with inherent parallelism such as low level image processing programs. These programs deal with image data which requires extensive and time consuming processing, but are usually of a much simpler type than the control programs. One example is edge detection. In this case, the data (the image) can be split into several "chunks" and put onto the available processors, which all run the same edge detection program on their part of the image[13] (a homogeneous problem). There is no complex control aspects, like starting up different programs on different processors and taking care of dual queues for message passing between the processes.

The two above mentioned categories of parallelism in the system could use as many parallel processors as there are possible processes. However, there is a limit on the number of processors: 19 in our case. Therefore, the question of processor utilization arises. There has to be a balance between the number of processors the two categories are allowed to occupy. Obviously, the most time consuming processes should use the maximum number of processors, thus reducing the number of "bottle necks" in the system. Since the low level image analysis will be the most expensive part with regard to execution time, it is preferable that these processes occupy most of the processors on the Butterfly, so as to prevent unecessary serial execution. The total execution time for achieving a goal will then be minimized. If the hierarchical control program occupies just a few nodes, it will not hurt the overall performance significantly, since even the serial version of the control does not take much execution time.

The 3-D object recognition algorithm

This section describes a new algorithm to recognize three-dimensional objects in range images using feature indexed hypotheses. The models used in object recognition are obtained from Computer Aided Design (CAD). Most of the earlier work using feature indexed hypotheses for recognition has been limited to two-dimensional objects[5,6]. Three-dimensional cases are much more complex. One needs to consider the three-dimensional model acquisition and complicated orientation problems. The key to recognizing three-dimensional objects using feature indexed hypotheses is to find the correct transformation matrix for the model. Using this matrix the model can be transformed to the location in the range image which corresponds to the position of the object being matched. The feature indexed hypotheses method will then be used to determine if the model and the object actually match. The basic principle of the feature indexed hypotheses[6] consists of the following:

1. Selecting pertinent features
2. Building a data base
3. Generating hypotheses
4. Verifying hypotheses

The pertinent features

The first step of the feature indexed hypothesis method is to select the pertinent features which are the most important features for recognition. The main criterion when selecting is that the pertinent features should be invariant to the object orientation. In an intensity image, the pertinent features of two-dimensional objects could be regions, corners or boundaries[5,6]. However, these types of features are not suitable for three-dimensional object recognition using intensity based methods. The intensity sensor (TV camera) projects a three-dimensional scene onto a two-dimensional image plane, and thus the shapes of regions, corners or boundaries are totally dependent on the orientation of a given object. The depth information used to represent three-dimensional objects is also lost. Therefore, to use intensity data only, for three-dimensional object recognition, is more difficult. With the advent of devices that can directly sense and provide the coordinates of points in space, researchers have been pursuing the integration of multisensors to provide three-dimensional information for recognition. A specialized ranging device, 100A White Laser Scanner, is used in our Robotics Laboratory to sense and provide the depth information. The 100A White Laser Scanner is a sophisticated laser scanning device that provides a contour map (range image) of the objects being measured by using Cartesian coordinates (X, Y, and Z). Three-dimensional Cartesian coordinates of points in space are provided directly by this ranging sensor without complex perspective projection problems[16].

In the current implementation, we concentrate on polyhedral object recognition using range data. The vertex and its three angles (Figure 6) are selected to be the types of features which are invariant to the object orientation. Magee et al.[17] select the length between two vertices as the type of feature, but the length will not be correct when objects are occluded. In this paper, the algorithm to extract features for polyhedral objects occurs in four steps:

1. Apply a gradient edge operator[18] to the range image to extract edges (see Fig. 7).
2. Perform Hough transform on the thinned edges to get line equations.
3. Compute the intersections of these line equations. Let the intersections be the feature vertices.
4. Calculate the three angles (θ_1, θ_2, θ_3) of the feature vertices. Junctions of regions in segmented images tend to be trihedral[12]. (θ_1, θ_2, θ_3) are sorted in descending order.

These low level algorithms can be efficiently implemented on the BBN Butterfly parallel processor, thus reducing computation time.

The data base

The data base shows which objects match each of the selected features. It contains the three angles (θ_1, θ_2, θ_3) of all the vertices in an object, matched model numbers, the vertex coordinate V and three neighbor vertex coordinates (V_1, V_2, V_3) (see Figure 6-(a)). For each model there may be several instances of one feature (identical features). They will appear as one feature only, in the feature table. Since this reduces the number of features it also reduces computation time. The data base will be searched when generating hypotheses.

Table 1 shows one example of the data base which contains three objects: Poly_1, Poly_2 and Poly_3 (see Figure 8). Poly_1 has five feature types: (136, 102, 94), (90, 78, 43), (139, 90, 90),(90, 90, 41) and (90, 90, 90). Features (136, 102, 94) and (90, 78, 43) have one match. Features (139, 90, 90) and (90, 90, 41) have five matches respectively (1 from Poly_1, 4 from Poly_2). Feature (90, 90, 90) has 11 matches (4 from Poly_1, 7 from Poly_3). Poly_2 has two feature types: (139, 90, 90) and (90, 90, 41). Poly_3 has four feature types: (180, 111, 69) with one match, (135, 111, 75) with one match, (159, 159, 29) with one match and (90, 90, 90) with nine matches.

Hypothesis generation

After the features are extracted, using the algorithm described in the "Pertinent Features" section, each feature of the unknown object (image) is compared with those of the data base. The model of an object will be considered as the *possible* indentity of the unknown object if the feature information fits the model. The features are compared using angle difference measures.

$$AngleDiff=\sum_{i=1}^{3} |\theta'_i-\theta_i| \qquad (4)$$

Here, (θ_1, θ_2, θ_3) is the feature in the image, and (θ'_1, θ'_2, θ'_3) is the selected feature in the model base. If the AngleDiff is less than a threshold , the hypotheses is generated. An example feature, and its matches in the possible model base, are shown in Figure 9. In the Figure 9 example, the feature type is (139, 90, 90). This feature matches six times in the data base. Four of the matches of the feature are shown in

Figure 9-(a). Two of the matches for the feature are shown in Figure 9-(b).

The hypothesis verification

The features which represent many identical matches could match several possible models when the AngleDiff is small enough to be considered as a match. The purpose of the hypothesis verification is to find which of the models is the real identity. The technique of verifying the hypotheses is that the image is translated and rotated to the same location and orientation as the possible matching models by applying a homogeneous transformation to the image. Homogeneous transformation consists of a homogeneous translation matrix and a homogeneous rotation matrix. An identity will be found *if all edges of the image object fit the corresponding edges in the possible model.* We choose the feature vertex coordinate (A,B,C) in the image as the translation position. A translation by (A,B,C) and a rotation by the nine unknowns (a,b,c,d,e,f,g,h,i) have these respective homogeneous matrices:

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ A & B & C & 1 \end{matrix} \qquad \begin{matrix} a & b & c & 0 \\ d & e & f & 0 \\ g & h & i & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

For solving the nine unknowns, we first need to find *three non-colinear points* in the image, which correspond to three points in the model. Recall the pertinent features where each feature consists of the vertex V and its three neighboring vertices V_1, V_2, and V_3 (thereby defining the three edges). Primed vertices refer to vertices in the model and unprimed vertices refer to vertices in the image. The vertex set (V, V') is the first set of corresponding points to be found when feature matching is done. In the real word, at least one of the three edges (VV_i, $1 <= i <= 3$) are not occluded, and one of these edges VV_i (choose the longest one if more than one) should then be equal to one of the three edges in the model ($V'V'_j$, $1 <= j <= 3$). The vertex set (V_i, V'_j) is the second set of corresponding points. The criterion for choosing the third set of corresponding points is to: first, select the edge which has the largest θ angle between VV_i and itself (say VV_k, $k <> i$, $1 <= k <= 3$), second, find the corresponding edge in the model with the same angle between it and $V'V'_j$, third, compute the point x' in the model by requiring $VV_k = x'V'$. To illustrate the rotation matrix solution procedure, consider Figure 11. (V,V') is the first set of corresponding points since the features are matched. VV_1 is the largest edge in the image and $VV_1 = V'V'_2$, so (V_1,V'_2) is the second set of corresponding points. θ_1 is the largest angle, so we find x' on $V'V'_1$ by requiring $x'V' = VV_2$. When we process this information, θ_1, θ_2, θ_3 could be equal. The lengths, VV_1, VV_2, VV_3 could also be equal. Therefore, all possible combinations of corresponding points must be computed in turn. The correct rotation matrix is found by checking that the three edges fit those of the model. Figure 10 shows an example of a rejected hypothesis. Some of the edges in the image (Poly_1) do not fit any of the edges in the possible model (Poly_2).

Conclusion

We have presented our ongoing work in developing a hierarchical robot control system based on three important aspects of the human brain: basic building blocks (neuroschemas), hierarchical organization, and parallel processing. The system consists of three basic parts which include the knowledge base, the data base, and the hierarchical control structure. The hierarchical control uses the information in the knowledge base and the models in the database to achieve any goals given to the system. Currently, these goals involve a method for three-dimensional object recognition. The main aspects of this method are the CAD-based geometric models for objects, and feature indexed hypotheses for recognition of these objects.

In the near future we will complete the implementation on the Butterfly parallel processor and obtain results for speedup when comparing parallel versus serial processing. We will also study processor utilization and process synchronization. We will continue expanding the scope of the tasks which the control system can handle, including recognition of more complex 3-D objects and their assembly.

Acknowledgements

This work was supported in part by NSF Grants DCR-8506393, DMC-8502115, ECS-8307483 and MCS-8221750

References

1. E.R. Lewis, "The Elements of Single Neurons: A Review", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 5, September/October 1983, pp 702-710.
2. M.A. Arbib, T. Iberall and D. Lyons, "Coordinated Control Programs for Movements of the Hand", University of Massachusetts, COINS Technical Report 83-25, August 1983.
3. T. Iberall and D. Lyons, "Towards Perceptual Robotics", University of Massachusetts, COINS Technical Report 84-17, August 1984.
4. K.J. Overton, "The Acquisition, Processing, and use of Tactile Sensor Data in Robot Control", University of Massachusetts, COINS Technical Report 84-08, May 1984.
5. R.C. Bolles and R.A. Cain, "Recognizing and Locating Partially Visible Objects: The Local-Feature-Focus Method", The International Journal of Robotics Research, Vol. 1, No. 3, Fall 1982, pp 57-82.
6. T.F. Knoll and R.C. Jain, "Recognizing Partially Visible Objects Using Feature Indexed Hypotheses", IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, March 1986, pp 3-13.

7. F.P. Kuhl, O.R. Mitchell, M.E. Glenn and D.J. Charpentier, "Global Shape Recognition of 3-D Objects Using a Differential Library Storage", Computer Vision, Graphics, and Image Processing, No. 27, 1984, pp 97-114.
8. J.S. Albus, Brains, Behavior, & Robotics, BYTE Books, Subsidary of McGraw-Hill, 1981.
9. J.A. Feldman, "Connections", BYTE, April 1985, pp 277-284.
10. J.A. Anderson, "Cognitive and Psychological Computation with Neural Models", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 5, September/October 1983, pp 799-815.
11. C.M. Brown, C.S. Ellis, J.A.Feldman, T.J. LeBlanc and G.L. Peterson, "Research with the Butterfly Multicomputer", Report from Computer Science Dept., University of Rochester, 1986.
12. D.H. Ballard and C.M. Brown, Computer Vision, Prentice-Hall, 1982.
13. (a) "Butterfly Parallel Processor Overview", BBN Report Number 6148, Version 1, March 6, 1986. (b) "Chrysalis Programmers Manual, Version 2.3", BBN Report Number 6191, May 1, 1986. (c) "Butterfly Parallel Processor Tutorial for Programming in the C Language", BBN Report Number 6190, March 10, 1986. (d) "The Uniform System Approach to Programming the Butterfly Parallel Processor", BBN Report Number 6149, Version 1, March 6, 1986.
14. C. Jacobus, W.D. Lee and J. Norton, "Flexible Assembly and Inspection of a small Electric Fuel Pump", SPIE Vol. 579 Intelligent Robots and Computer Vision, 1985, pp 528-536.
15. L.G. Roberts, "Machine Perception of Three-Dimensional Solids", Optical and Electro-Optical Information Processing, J.T. Tippet et al., Eds., 1965, pp 159-197.
16. R.O. Duda and P.E. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
17. M.J. Magee, B.A. Boyter, C.H. Chien and J.K. Aggarwal, "Experiments in Intensity Guided Range Sensing Recognition of Three-Dimensional Objects", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 6, November 1985, pp 629-637.
18. B. Bhanu, S.K. Lee, C.C. Ho and T.C. Henderson, "Range Data Processing: Representation of Surfaces by Edges", Eight International Conference on Pattern Recognition, Paris, France, October 27-31, 1986.

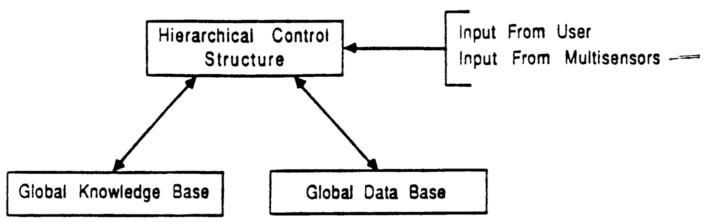

Figure 1: The Basic Building Blocks of The Control System

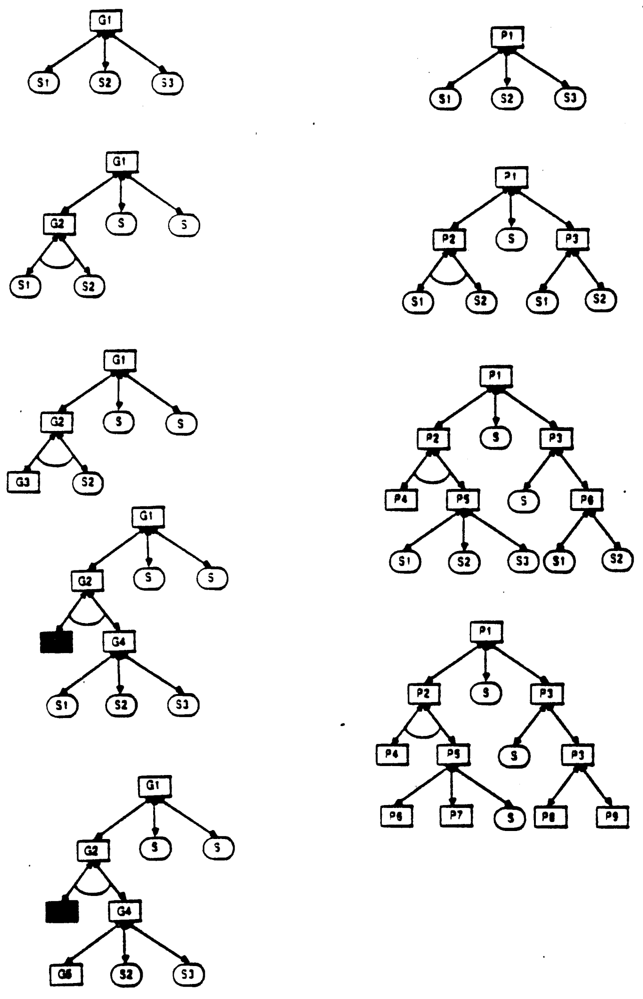

Figure 2: Hierarchical Control Tree: Serial and Parallel Versions

```
Achieve_Goal(Goal,Goal_Information)
char       *Goal;
subgoal    *Goal_Information;
{
     while (STATUS != DONE)
     {
          /* Activation Section */
          STATUS = Check_Goal_Status(Goal_Information)

          /* Event Section */
          if (STATUS == GO) Achieve_SubGoals(Goal_Information)

          /* Learning Section */
          if (STATUS == LEARN) Learn_From_User(Goal,Goal_Information)
     }
}
```

Figure 3: The Neuroschema

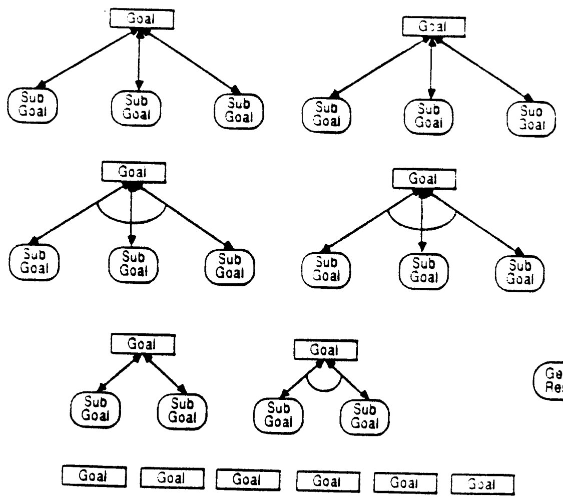

Figure 4: The Knowledge Base

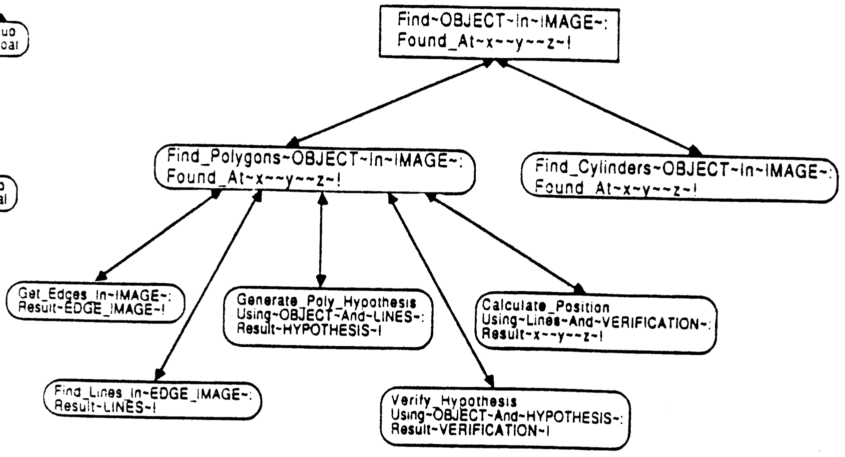
Find~OBJECT~In~IMAGE~: Found_At~x~~y~~z~!
Find_Polygons~OBJECT~In~IMAGE~: Found_At~x~~y~~z~!
Find_Cylinders~OBJECT~In~IMAGE~: Found_At~x~y~~z~!
Get_Edges_In~IMAGE~: Result~EDGE_IMAGE~!
Generate_Poly_Hypothesis Using~OBJECT~And~LINES~: Result~HYPOTHESIS~!
Calculate_Position Using~Lines~And~VERIFICATION~: Result~x~~y~~z~!
Find_Lines_In~EDGE_IMAGE~: Result~LINES~!
Verify_Hypothesis Using~OBJECT~And~HYPOTHESIS~: Result~VERIFICATION~!

Figure 5: Goal With Subgoals

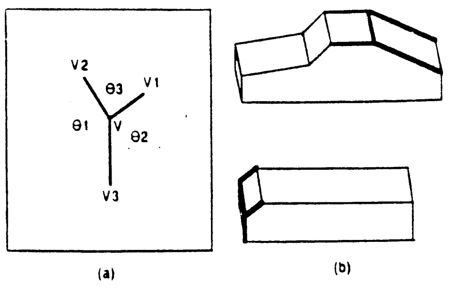

Figure 6: Example of the feature indexed hypotheses method. (a) Selceting a vertex and its three angles as the pertinent feature. (b) Occurrences of feature in data base.

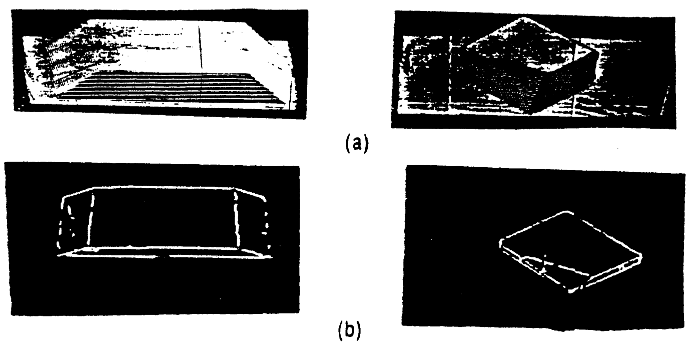

Figure 7: Test images: Poly_2 & Poly_3. (a) Range image. (b) Edge detection result.

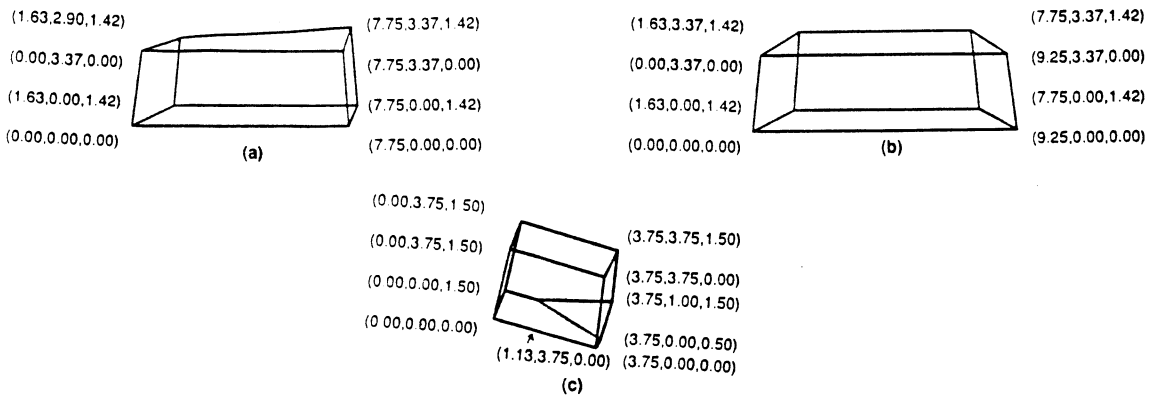

Figure 8: Models: (a) Poly_1. (b) Poly_2. (c) Poly_3.

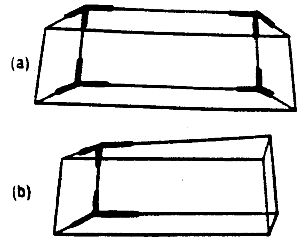

Figure 9: The feature type is (139,90,90). This feature matches six times in the data base. Four of the matches are shown in (a). Two of the matches are shown in (b).

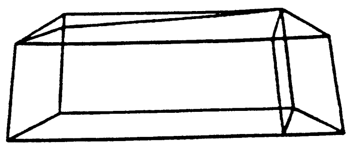

Figure 10: Some of the edges in the image (Poly_1) do not fit any of the edges in the possible model (Poly_2).

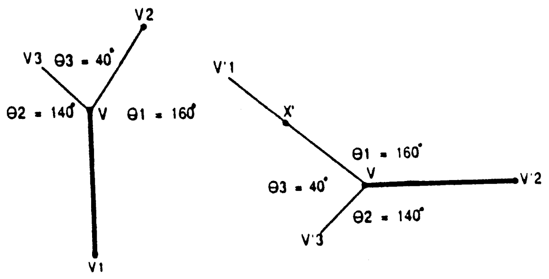

Figure 11: Select corresponding points example

```
(01,02,03)=(136,102, 94)

Poly_1
 V =(1.63, 2.90, 1.42)
 V1=(0.00, 3.37, 0.00)
 V2=(7.75, 3.37, 1.42)
 V3=(1.63, 0.00, 1.42)


(01,02,03)=( 90, 78, 43)

Poly_1
 V =(0.00, 3.37, 0.00)
 V1=(7.75, 3.37, 0.00)
 V2=(1.63, 2.90, 1.42)
 V3=(0.00, 0.00, 0.00)


(01,02,03)=(139, 90, 90)

Poly_1
 V =(1.63, 0.00, 1.42)
 V1=(1.63, 2.90, 1.42)
 V2=(0.00, 0.00, 0.00)
 V3=(7.75, 0.00, 1.42)

Poly_2
 V =(7.75, 3.37, 1.42)
 V1=(9.25, 3.37, 0.00)
 V2=(7.75, 0.00, 1.42)
 V3=(1.63, 3.37, 1.42)

Poly_2
 V =(1.63, 3.37, 1.42)
 V1=(0.00, 3.37, 0.00)
 V2=(7.75, 3.37, 1.42)
 V3=(1.63, 0.00, 1.42)

Poly_2
 V =(7.75, 0.00, 1.42)
 V1=(9.25, 0.00, 0.00)
 V2=(7.75, 3.37, 1.42)
 V3=(1.63, 0.00, 1.42)

Poly_2
 V =(1.63, 0.00, 1.42)
 V1=(1.63, 3.37, 1.42)
 V2=(0.00, 0.00, 0.00)
 V3=(7.75, 0.00, 1.42)


(01,02,03)=( 90, 90, 41)

Poly_1
 V =(0.00, 0.00, 0.00)
 V1=(7.75, 0.00, 0.00)
 V2=(0.00, 3.37, 0.00)
 V3=(1.63, 0.00, 1.42)

Poly_2
 V =(9.25, 3.37, 0.00)
 V1=(7.75, 3.37, 1.42)
 V2=(0.00, 3.37, 0.00)
 V3=(9.25, 0.00, 0.00)

Poly_2
 V =(0.00, 3.37, 0.00)
 V1=(9.25, 3.37, 0.00)
 V2=(1.63, 3.37, 1.42)
 V3=(0.00, 0.00, 0.00)

Poly_2
 V =(9.25, 0.00, 0.00)
 V1=(9.25, 3.37, 0.00)
 V2=(7.75, 0.00, 1.42)
 V3=(0.00, 0.00, 0.00)

Poly_2
 V =(0.00, 0.00, 0.00)
 V1=(9.25, 0.00, 0.00)
 V2=(0.00, 3.37, 0.00)
 V3=(1.63, 0.00, 1.42)


(01,02,03)=(180,111, 69)

Poly_3
 V =(3.75, 1.00, 1.50)
 V1=(3.75, 0.00, 1.50)
 V2=(3.75, 3.75, 1.50)
 V3=(1.13, 0.00, 1.50)


(01,02,03)=(135,111, 75)

Poly_3
 V =(3.75, 0.00, 0.50)
 V1=(3.75, 0.00, 0.00)
 V2=(3.75, 1.00, 1.50)
 V3=(1.13, 0.00, 1.50)


(01,02,03)=(159,159, 29)

Poly_3
 V =(1.13, 0.00, 1.50)
 V1=(3.75, 1.00, 1.50)
 V2=(3.75, 0.00, 0.50)
 V3=(0.00, 0.00, 1.50)


(01,02,03)=( 90, 90, 90)

Poly_1
 V =(7.75, 3.37, 0.00)
 V1=(7.75, 3.37, 1.42)
 V2=(0.00, 3.37, 0.00)
 V3=(7.75, 0.00, 0.00)

Poly_1
 V =(7.75, 3.37, 1.42)
 V1=(7.75, 3.37, 0.00)
 V2=(7.75, 0.00, 1.42)
 V3=(1.63, 2.90, 1.42)

Poly_1
 V =(7.75, 0.00, 0.00)
 V1=(7.75, 3.37, 0.00)
 V2=(7.75, 0.00, 1.42)
 V3=(0.00, 0.00, 0.00)

Poly_1
 V =(7.75, 0.00, 1.42)
 V1=(7.75, 0.00, 0.00)
 V2=(7.75, 3.37, 1.42)
 V3=(1.63, 0.00, 1.42)

Poly_3
 V =(0.00, 3.75, 1.50)
 V1=(0.00, 0.00, 1.50)
 V2=(0.00, 3.75, 0.00)
 V3=(3.75, 3.75, 1.50)

Poly_3
 V =(0.00, 3.75, 0.00)
 V1=(0.00, 0.00, 0.00)
 V2=(0.00, 3.75, 1.50)
 V3=(3.75, 3.75, 0.00)

Poly_3
 V =(3.75, 3.75, 0.00)
 V1=(3.75, 3.75, 1.50)
 V2=(0.00, 3.75, 0.00)
 V3=(3.75, 0.00, 0.00)

Poly_3
 V =(3.75, 3.75, 1.50)
 V1=(3.75, 3.75, 0.00)
 V2=(0.00, 3.75, 1.50)
 V3=(3.75, 1.00, 1.50)

Poly_3
 V =(3.75, 0.00, 0.00)
 V1=(3.75, 0.00, 0.50)
 V2=(3.75, 3.75, 0.00)
 V3=(0.00, 0.00, 0.00)

Poly_3
 V =(0.00, 0.00, 1.50)
 V1=(0.00, 0.00, 0.00)
 V2=(0.00, 3.75, 1.50)
 V3=(1.13, 0.00, 1.50)

Poly_3
 V =(0.00, 0.00, 0.00)
 V1=(0.00, 0.00, 1.50)
 V2=(3.75, 0.00, 0.00)
 V3=(0.00, 3.75, 0.00)
```

Table 1: Data Base Selected for Features