Comp. & Maths. with Appls. Vol. 11, No. 5, pp. 505–515, 1985
Printed in Great Britain.

0097-4943/85 $3.00 + .00
© 1985 Pergamon Press Ltd.

SYMBIOSIS IN COMPUTATIONAL VISION SYSTEMS

JAY GLICKSMAN
Computer Science Laboratory, Texas Instruments, P.O. Box 226015, MS 238, Dallas, TX 75266, U.S.A.

(*Received September* 1984)

Abstract—While the goal of computational vision systems is the totally automated understanding of images, it is not necessary for this goal to be achieved before practical vision systems can be developed. In particular, systems that require some amount of human intervention can be applied to real problems with beneficial results. In "symbiotic" vision systems the computer brings something to bear on the problem that either replaces or supplements a subtask that would otherwise be done by a human expert.

In symbiotic systems there is a large range of possibilities for the type and amount of interaction that the human expert must provide. This can range from minor aid to the computer system, when it reaches an ambiguity that it cannot resolve, to major control of the processing in complicated regions of the image that are of primary interest. Symbiotic vision systems must allow the user to access the system at the point in the range that is suitable. In addition, the system must have facilities to both present information and accept "advice" from the expert in a way that is natural and convenient.

Two example vision systems will illustrate different ways in which these problems have been solved. The first, MISSEE, uses a cycle of perception combined with a schema-based system architecture to provide a flexible framework in which the user can select the amount of interaction he wishes to undertake. The second, which carries out the normal moveout phase of seismic data processing, has a more limited focus, but provides a natural means of communication.

1. INTRODUCTION

The goal of producing an automated vision system has proved to be a difficult one to accomplish. The ambiguity inherent in digitized perceptual data has inhibited solutions to the mapping from image to scene in all but the most restricted cases. One response to this difficulty has been the emphasis on "early" visual processes where the results stop short of the boundary between image and scene. An alternative methodology is to provide expert (human) assistance to the vision system to guide it towards appropriate solutions.

Symbiosis can be defined as "the living together of two dissimilar organisms in a close association that is advantageous to both"[1]. Ignoring the fact that a computer program is not a living organism, man and machine can work together profitably. Computers are a tool and as such are helpful to their user (if they are appropriate for the task at hand). Conversely, a user that provides advice to a program could be considered to be helping that system. However, if the human has to do too much of the work, then the benefit of using the machine decreases. Consequently, the form and amount of the symbiotic interaction must be suitable for the man, not the machine.

A simple form of man–machine interaction is selection, whereby the machine selects images that could be of interest to a human expert. An example of the use of this type of interaction would be to automatically select satellite images of a particular locale where important features were not obscured by cloud cover.

The user can aid the machine during the actual interpretation process by disambiguating image features. A small amount of human intervention could produce significantly better results if the information provided was used at critical points in the analysis. Furthermore, if a truly symbiotic relationship can be formed between man and machine, then it would be possible for more difficult problems to be solved through cooperation than could be handled by man or machine alone.

Several researchers have advocated the usefulness of man–machine interaction in vision systems[2–5], expert systems[6], and theorem provers[7]. One reason is that

> The substantial amount of ad hoc world knowledge required to plan perceptual strategies is most reasonably acquired in an incremental fashion. The system should thus be designed to request additional information from a user at times of failure, indecision, or on encountering

a new object and to incorporate this information immediately in a revised strategy.[Ref. 8, pp. 8–9]

Human interaction would be most useful in current vision systems to resolve problems due to ambiguity or indecision. If a vision program has reduced the number of possibilities for an interpretation to a small number, then very little "information" is required from the user to resolve the difficulty. Furthermore, interaction can be useful in combating the other problems of model-based vision: inconsistency, unsatisfiability, and incompleteness (cf. [9]).

One of the criteria for the success of a computer vision system is that it display graceful degradation. A program is said to degrade gracefully if it is able to produce partial results when all of its input requirements have not been met. Hence, if the user chooses not to participate in the interpretation process, the system should still be robust enough to continue without the additional guidance. The more useful information the user provides, the better the results one would expect. If the user wishes to input information, a vision system should accept it.

In a symbiotic system the person and the machine play different roles, each doing what they are good at. The person would give high-level direction to specify what in the image is "interesting"; the computer system would do the mindless bookkeeping (e.g. counting pixels). The man could act as teacher or advisor; the machine, as learner and doer. In his role as mentor, the human determines the form that the interaction should take.

2. THE PROBLEM

There are two basic problems in building symbiotic vision systems. First, there is the question of how to allow the user to address the machine at a suitable level of abstraction. An artificial intelligence system should not force the user into a constrained, low-level dialogue. Rather, the user should be able to direct the system in a manner of his own choosing. In addition, systems that are to learn must be able to accept generalizations that span several conceptual levels. The second issue for the designer of a symbiotic system concerns the form of the input and output: that is, what information to provide the user, when it is appropriate to display it, how it should be presented and, conversely, what and when information should be received.

Man–machine communication is best achieved when the computer is able to relate to the user in a natural manner. Thus, natural language is to be preferred over obscure programming languages. In image understanding, where pictorial information is so important, graphic input and output are often the most direct ways for exchanging information.

Graphic output translates the internal data structures manipulated by computer programs to images that correspond to the original inputs, exaggerate features of interest, or show the results of interpretation in specific regions. This can be done using overlays, pseudocolouring, diagrams, and schematics. The format of the display will depend on the amount of data to be displayed, the domain, the application, and the type of user interaction to follow.

Graphic input translates user intentions into actions to be carried out on internal data. It usually consists of pointing at parts of a displayed image, creating geometric entities (e.g. splines, circles), or freehand drawing. An example of a type of drawing from the geographic domain is the sketch map. Sketch maps depict certain standard symbols that are abstractions of what might appear in a scene. They contain information concerning what is in a scene as well as the topological relationships among the entities.

More direct information might be input to a vision system via text. Whereas a sketch map retains some analogic properties of the intensity image, textual input does not. It is also usually much more abstract than an intensity image. Examples include specifying the existence, location, or characteristics of an object.

An interactive vision system should only make requests from the user at times when critical decisions must be made, or else it would lose its effectiveness as a tool. However, how is the system to know what the user considers critical? One way is for the system to have a user model that has knowledge of the preferences and habits of the individuals that use the system[10]. Then the system can refer to the model in the context of the current decision when determining if (and how) the user should be queried.

The next two sections describe particular solutions to the problems outlined above. The first, MISSEE, can be characterized as having limited input/output capabilities, but offering a wide range of possibilities in the continuum of the amount of interaction required of the user. The second, which removes normal moveout from raw seismic traces, features a more sophisticated interaction environment, but has a more restricted control mechanism.

3. EXAMPLE 1: THE MISSEE SYSTEM

MISSEE is a vision system that can interpret aerial photographs of small urban scenes. Its performance is enhanced by the acceptance and use of information from the user in several distinct forms. The ability to utilize a wide range of interaction types is a result of the schema-based architecture the system is built around[11]. Advice concerning particular aspects of the image can be accommodated by the appropriate node in the semantic network. Interaction is a function of the major control paradigm utilized in the system: a cycle of perception[12].

MISSEE can accept from one to three sources of input information. The first source, the only required one, is the digitized image that is to be interpreted. A second source is a sketch map that the user can draw on top of the image to give an overview of the major objects in the scene and their geographic relationships. The third information source is textual input entered by the user.

The three different information sources provide MISSEE with very different sorts of data. The digitized image is processed to yield information in the form of edges, regions, and pixels. The sketch map is analyzed by a program called MAPSEE[13], which generates a semantic network of MAYA instances[14] that relate to the sketch in terms of chains and regions. The textual user interface deals with much more abstract concepts, such as roads and river systems. Thus the system must be able to exchange a variety of types of information with the user.

To be able to accommodate disparate types of knowledge requires a method of representation independent of the nature of the images. MISSEE employs a schema-based system called MAIDS[9], which is an extension of MAYA. The schemata correspond to (possibly hypothetical) instances of scene entities, such as roads, that are found in images. These instances are arranged into a semantic network based on the various relationships among them: specialization, composition, and neighbour. Procedures attached to schemata are the means by which the instances and semantic network can be generated from the various facts as they become available. The control mechanism utilized by MISSEE determines how the interpretation and the interaction take place.

Control proceeds by means of a cycle of perception[15,16]. Figure 1 shows the version used in MISSEE. Its three phases are "schema" directing "exploration" that samples "objects" in the image that modifies the "schema." The cycle of perception provides convenient points for communication with the user. At each node in the cycle there is a particular, useful kind of information that can be exchanged.

Outputs

Global. When a schema is invoked, the model type (e.g. river) can be described for the user so that he knows what the current focus of attention is. Furthermore, as the schema is instantiated—by having its slots filled with values—the information can be output.

Local. The information derived from the image can be output when the image is sampled. It can be in the form of verifications of hypotheses directed by the schema, facts concerning the image features that may be relevant to the schema, or data that can cause a shift in attention.

Planning. This is a list of possibilities for further processing. As they are executing, the schema-specific procedures can make inferences about other models that should be invoked. They will then send a message to alert the appropriate schema. Those messages plus the user's textual input make up the possibilities list.

The possibilities list is a priority queue. Its entries are ranked by the schema that sent the message. The value given by the schema will generally be a factor of its own confidence and the certainty of the inference that caused the message to be sent. The rankings are shown to the user so that he can see how processing will continue and can assert his priorities by rearranging the queue. The latter is done via priorities input.

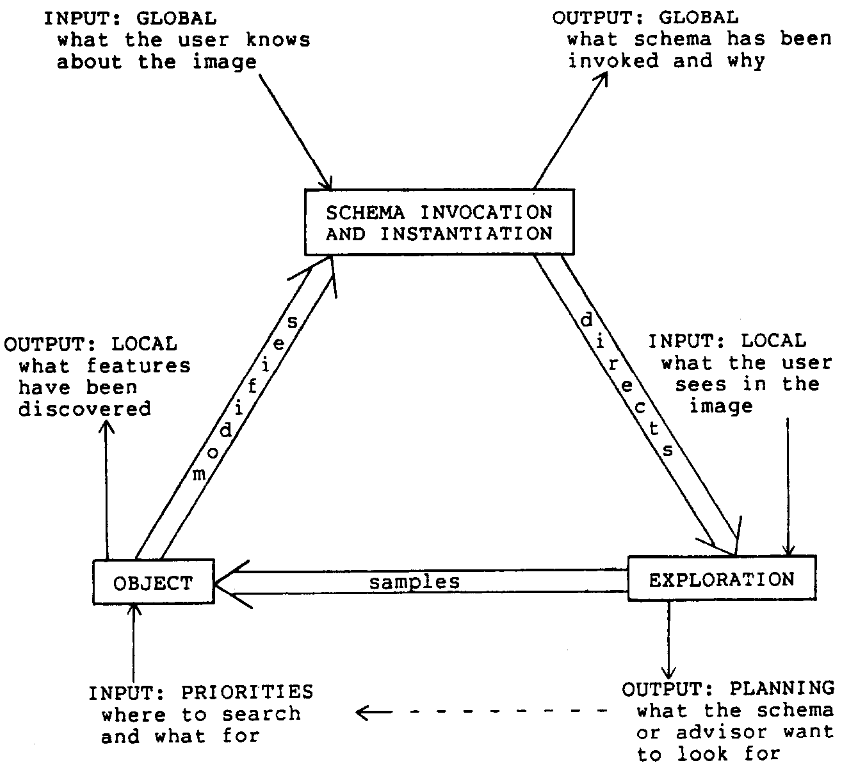
INPUT: GLOBAL
what the user knows about the image
OUTPUT: GLOBAL
what schema has been invoked and why
SCHEMA INVOCATION AND INSTANTIATION
OUTPUT: LOCAL
what features have been discovered
modifies
directs
INPUT: LOCAL
what the user sees in the image
OBJECT
samples
EXPLORATION
INPUT: PRIORITIES
where to search and what for
OUTPUT: PLANNING
what the schema or advisor want to look for

Fig. 1.

Inputs

Global. The user can enter general or specific information about the scene. General data includes parameters that affect all levels of processing, for example, the scale of the aerial photograph. Specific information pertains to specific objects. For instance, one can indicate that there is a road in the picture. This information can cause entries to be added to the possibilities list or might cause global schemata or variables to be modified.

Local. More specific information can also be introduced concerning the interpretation that is taking place. For instance, the user is able to redirect the attention of a schema's procedure to a different location in the image.

Priorities. The user can influence subsequent processing by modifying the priority queue. Rearranging the entries changes the search and instantiation priorities of objects. Objects that are of no interest or that are perceived to be incorrectly identified can be removed from the queue.

The cycle of perception is well suited to a schema-based, object-oriented vision system. It allows feedback from previously instantiated schemata to provide a more informed context for subsequent processing. Also, interaction is facilitated by having clearly distinguished points in the cycle where different types of information can be exchanged.

Control is exercised through the cycle of perception by means of a global priority queue. Each entry on the queue designates a schema, how it is to be entered, and a possible context for the evaluation of one of its attached procedures. This is analogous to, but more limited than, pattern-directed invocation in languages such as MAYA and achieves a similar modularity.

Messages are sent from schemata to redirect attention to other schemata as the results of interpretation become available. The user can also initiate messages either to impart new information or to make known his requirements. A message is made up of four parts: (1) the name of the generic schema it is directed to; (2) how the schema is being entered (top-down or bottom-up); (3) a priority number; and (4) parameters that are used to establish the context when the appropriate procedure attached to the schema is evaluated. Consequently, the sender of the message (a schema or the user) must "know" two aspects of the receiver. The name of the recipient schema must be known, as well as where it sits in the hierarchy relative to the sender. Lateral messages (i.e. heterarchical rather than hierarchical communication) are always sent top-down. A priority value is included to rank the message. The user can provide a value to reflect the importance of his message relative to those already in the queue. A schema will generally use its confidence value plus or minus some small number to reflect the importance of the message. Finally, some information useful to the procedure, such as an image location, may be provided.

The execution cycle shows where the opportunities for interaction occur in the system as implemented (see Fig. 2). Interaction with the user can be easily controlled and varied. In terse mode the user is not prompted for information as interpretation progresses. On the other hand, if terse mode is off, then during each cycle the user has a chance to query the system for detailed information (over and above what is provided in the displays and in the protocol) and to influence processing in various ways: A new control loop is entered that permits the user to escape to LISP (to change global values, examine schemata, interact with the environment) or modify the priority queue. When finished, execution is resumed or the cycle is broken.

This execution cycle appears to be quite different from the cycle of perception (Fig. 1). However, it controls interpretation in the same manner, and each aspect of the cycle of perception can be found somewhere in the execution loop. Both provide for the timely instantiation of relevant schemata and maintain a focus of attention.

An example of the interchange that occurs in MISSEE is shown in Fig. 3. This figure contains the interaction phase of the execution cycle and, subsequently, the protocol output as %town-1 is instantiated. Figure 4 shows the display that might be shown at the end of this step of the processing.

In MISSEE the user is able to do as much or as little as desired to influence the progress

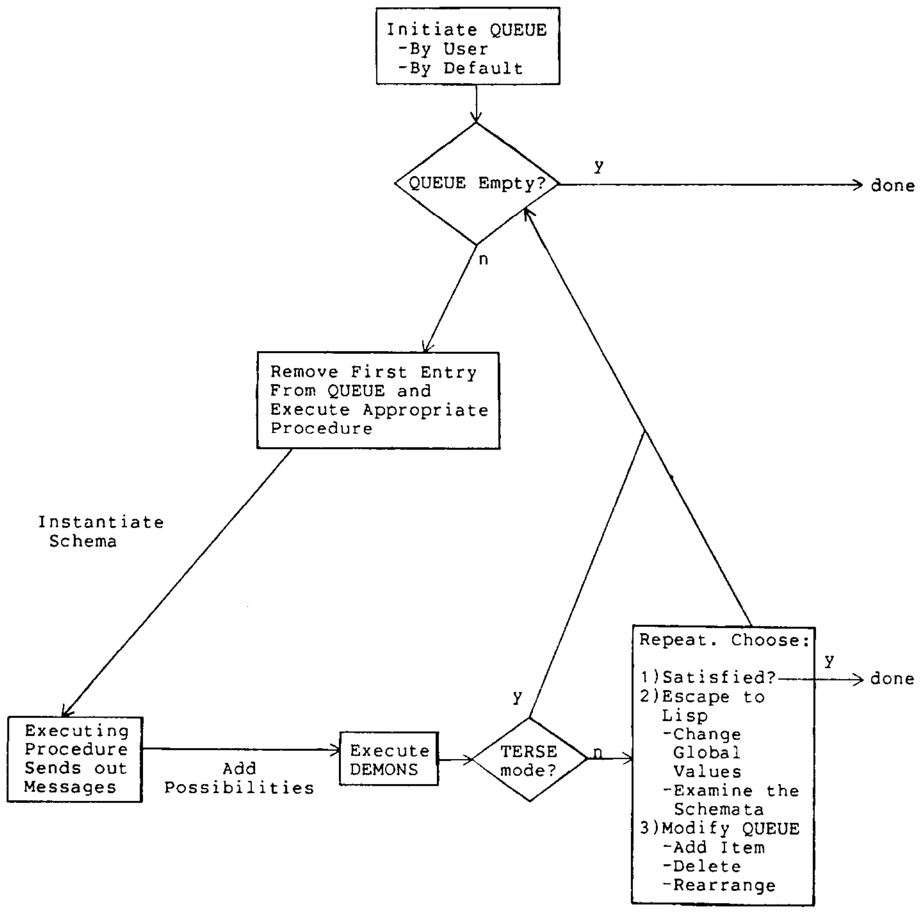

Fig. 2.

```
        ** STEP 23. ***************************************

  SCHEDULER: Here is the priority queue:

  ;; The first line indicates that a top down message has been sent to
  ;; the town schema with priority 50 to try to instantiate a town at
  ;; region 738 in the image.
  ;; *town-1 is an instance of a town found by Mapsee in the sketch map.

  %QUEUE = ((50 town-top-down ( 738 region))          [1]
            (50 town-top-down (  19 region))          [2]
            (50 road-top-down (  18 region))          [3]
            (50 road-top-down (1629 region))          [4]
            (50 town-bottom-up-sketch-map (*town-1))) [5]

    Do you want to modify it(y), be in lisp(l), or quit(q)? y

  Commands: d num   --Delete element num
            a s-exp --Add s-exp to the Queue using its priority
            m n1 n2 --Move element n1 to after n2
            q       --Quit processing
            e       --End modifying Queue, continue

  Command: m 5 0
  SCHEDULER: What is the new priority for 5? 70

  1. -> (70 town-bottom-up-sketch-map (*town-1))
  2. -> (50 town-top-down ( 738 region))
  3. -> (50 town-top-down (  19 region))
  4. -> (50 road-top-down (  18 region))
  5. -> (50 road-top-down (1629 region))

  Command: e

  OBJECT: town-bottom-up-sketch-map for *town-1
  town-bottom-up-sketch-map:
          Create a new town instance %town-1
          *** Model Consistency for town from sketch map ***
          regions list =  (9)
          Interpretations are (URBAN HILLS)
          One interpretation must be consistent with "urban" or fail
          %town-1 is model-consistent
          Centre of town at (51 . 81)
          Neighbouring road regions are ()
  %town:  value added to regions = (9)
  %town:  value added to sketch-map-item = *town-1
  SCHEDULER: Adding to Queue (50 road-top-down (   8 region))
  SCHEDULER: Adding to Queue (50 road-top-down (1006 region))
  town-bottom-up-sketch-map: %town-1 must-be-part of some landmass
  SCHEDULER: Adding to Queue (53 landmass-bottom-up (%town-1))
  DEMON:  Adding neighbours link from %road-1 to %town-1
```

Fig. 3.

of interpretation. By taking an active role the user can guarantee that the results will be to his liking. Depending on how well the automatic system is performing, this may take more or less effort on his part.

At one extreme the user may choose to let the system run automatically. It is not required that he draw a sketch map. In the current implementation he must help train the intensity categorizer to assign possible classes to regions in the image, but it is possible to automate this phase. In terse mode MISSEE will not interrupt its execution cycle and will continue until the priority queue is empty. MISSEE will have automatically produced a protocol and images. If that output is insufficient, then after the execution cycle is finished, the user may choose to examine the semantic network in greater detail.

The results from this type of processing will be somewhat inaccurate. In an experiment with six images, MISSEE's error rate (in terms of regions incorrectly identified over the total number of regions) was 24%. The user can help the situation significantly by drawing a sketch map of the scene over the image. In the experiment this reduced the error to 9%. Even better results will follow if the user takes a more active role in the interpretation process.

The user can provide three types of textual input to the system: global, local, and priorities. In terms of improving the accuracy of the results, it may be necessary only to make minor

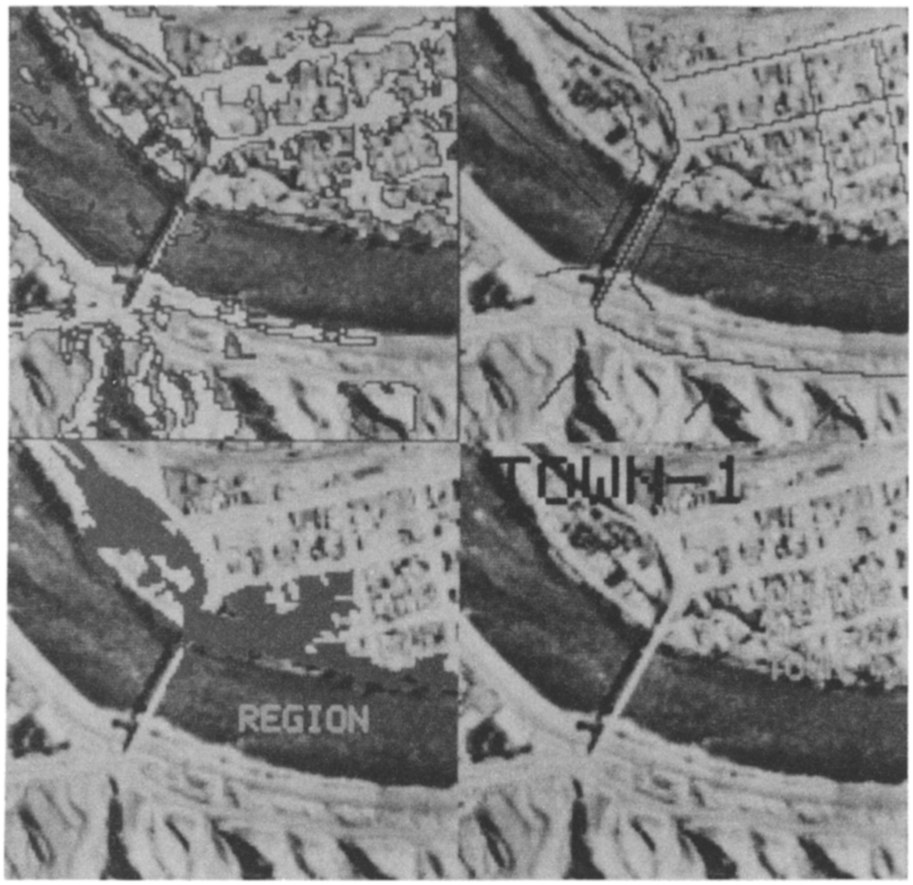

Fig. 4.

changes in the parameters of messages on the possibilities list. More drastic measures would include adding or deleting messages. In extreme cases where scene objects have not been instantiated or nonexisting entities have been, it is possible to modify the semantic network directly by creating and destroying instances. Thus, the user can ensure that the end results reflect his perception of the scene.

With regard to task priorities, the user can narrow the system's focus. If the user is only interested in objects of a certain type, such as roads or river systems, then MISSEE can be directed to look for only those objects (and their components and specializations) and to ignore irrelevant messages. In addition, the user can be very specific about where to find the items of interest. Subsequently, specific parts of the resulting schemata can be examined to obtain the desired information about the objects.

The user also controls the environment so that the interaction fulfills his needs. Global input determines how much is included in the protocols, if and where graphic output appears, and whether the user wants to have a chance to interact in each loop of the execution cycle.

In summary, the user can influence the what, when, where, and how of interpretation to get the desired results. The amount of interaction can range from almost none to a step-by-step dialogue. The user would generally vary the amount of interaction, depending on the accuracy of processing, priorities, or both. This provides a very flexible adjunct to automatic processing.

4. EXAMPLE 2: INTERACTIVE NORMAL MOVEOUT

Whereas MISSEE was designed to recognize a wide range of objects, the system to remove the effects of normal moveout (NMO) has a more limited purpose. To reduce noise in seismic reflection studies of the subsurface, several readings at a common depth point are combined. However, due to the way that these surveys are shot, the geometry between sound emitter, reflectors, and receivers is different for each receiver. The farther away the receiver is from the emitter, the longer it will take for the signals to arrive, and it will appear that a reflection event is "deeper" (as measured by time) in the earth. This is called normal moveout and can be compensated for if the distance between emitter and receiver are known as well as the velocity of the signal through the subsurface media that it passes. However, the velocity is only approximately known at best, so that the process is generally an interactive one where different velocity maps are tried in turn until an "acceptable" image results.

Figure 5 shows a copy of a LISP machine screen that contains several window panes. The pane labeled "pre-NMO traces" shows the raw data. The top of the screen represents the surface of the Earth, and the emitter in this case was at the far left side. The signal received by each sensor is called a trace, and the traces are displayed from the top in increasing time (the axis is actually part of the "velocities" pane), which varies monotonically but nonlinearly with depth because the velocity also varies nonlinearly with depth. Amplitude changes are called

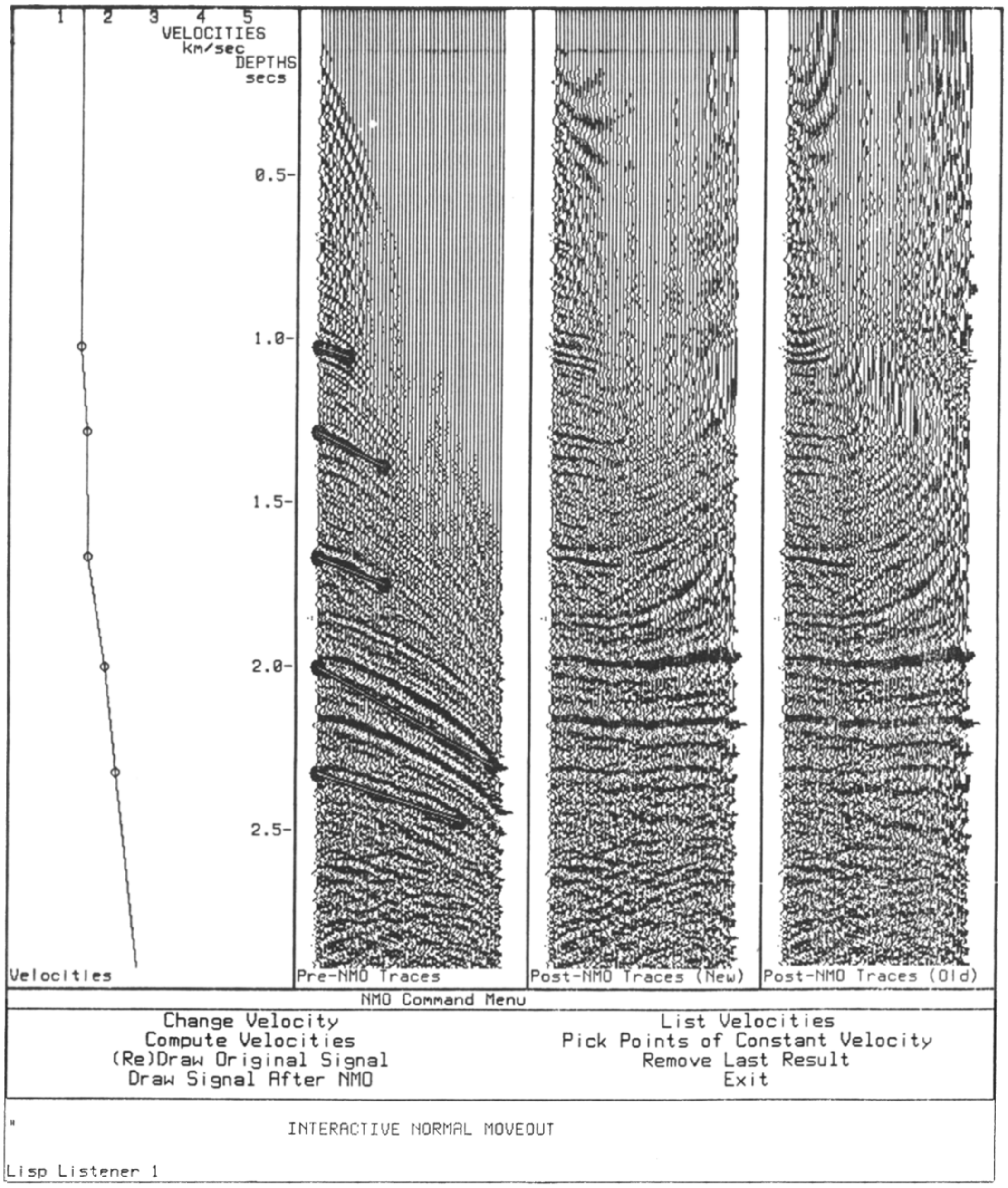

Fig. 5.

events, and, in this form of display, positive amplitudes are darkened so that they stand out. Coincident events in neighbouring traces "move out" due to the increasing distance from the receiver to the emitter. It is the task of the NMO process to "straighten out" these events.

Signals are stored as a generic data type within this system. They consist of a data array (amplitude values for a given trace and time) plus associated information about the data, such as its units of measurement and axes values. They are an abstract object within an object-oriented environment and as such can be used in a wide variety of domains (vision, speech, seismic), since the actions that can be carried out upon them are not limited to specific applications. In terms of the data structures, the result of NMO processing will be a new signal that contains the corresponding, corrected values.

There are two approaches to solving this problem. First, one can use whatever knowledge is available, from sources such as well logs or geology, to determine reasonable values for the velocities. One would then interpolate between the given values, remove the effects of NMO, and examine the results. Where the events had been moved too much, the velocity value should be lowered, and vice versa. The process could be iterated until the user was satisfied.

The other method is to examine the traces themselves and determine visually which points should be on the same straight (horizontal) line. From the two points it is possible to calculate the velocity that would be necessary for them to have occurred at the same time. Then the process becomes similar to the first one. After NMO compensation, the results could be modified by editing the placement of the two endpoints.

This second method lends itself to an automatic solution. The high-amplitude events move across the traces like a line across an image. In fact, they are restricted to always move "down" the image as they move away from the emitter. Thus a fairly straightforward line follower is sufficient to connect the strong events across traces. For the purposes of this application, these images can be "understood" by determining the positions and strengths of the events. The endpoints can be used as in the second method above to generate the appropriate velocities to carry out the NMO.

In the implemented system the control is left to the user. Choices are made by selecting items from a command menu. There are three initial operations, and they connect to each other directly. When the user has some knowledge of the area, he can input the velocities (*change velocity*). Using the LISP machine's pointing device (a mouse), he can add, delete, and modify depth/velocity values in the "velocities" pane. For feedback, numeric values are displayed, and rubber-band lines are drawn as the mouse is moved. At the next step the user can use visual (and geophysical) knowledge to pick points of constant velocity in the "pre-NMO traces" pane. Endpoints can be added, deleted, and modified using rubber-band lines, as was the case with the velocities. When the user is finished with this phase, the velocities that correspond to the constant velocity pairs are calculated and displayed in the "velocities" pane. The user can then graphically edit them if he desires. Finally, the system can automatically compute velocities. This results in lines being drawn in the "pre-NMO traces" pane and velocities being drawn in the "velocities" pane. Either can be edited.

After the input phase the effects of NMO can actually be removed (*draw signal after NMO*), a new signal created, and the visual results displayed in the "Post-NMO traces (New)" pane. The user can then go back and graphically make any changes that are desired. After a second NMO is carried out, the first results are moved into the "post-NMO traces (Old)" pane so that one can see whether they are converging towards a satisfactory result. If the new results are not as good as the previous one, the user can remove the last result and back up one step. When finally satisfied, the resulting signal can be passed on to the next stage of seismic data processing.

The NMO system provides a flexible interactive environment so that the user can submit information and see the results of his changes in a natural way. However, the range of advice given by the system to the user is very limited, and the control is handled by the user, not the system, as was the case for MISSEE. The NMO program can offer one potential solution that the user can then modify, but it does not maintain a "deep" understanding of the data. However, for this limited application an initial solution is a good starting point for the subsequent iterations that will converge to a final result.

5. DISCUSSION

Although the goal of computational vision is to produce totally automated systems, the assumption made in this paper is that interactive, symbiotic systems are a worthwhile stepping stone along the way. The obvious results of symbiosis are (1) the computer system need not implement all the modules necessary to reach a solution, and/or (2) the accuracy of the results will be improved over those that could be obtained by the automated system alone. The first result implies that vision systems can be built incrementally. The second result holds promise that practical applications may be found that can be economically handled by such systems.

If symbiotic systems are to be an aid in solving the larger vision problem, then they must not be more difficult to solve than the original problem. This requires finding a suitable method for exchanging information between the program and the user. There are two major issues involved: the form that the information should take, and the means by which it is to be exchanged.

The form of the information

The human visual system is a complex information processor that makes available to consciousness a wide range of data. However, much of the early visual processes, such as edge detection, motion detection, and colour perception, are not available to consciousness. People perceive scenes not images. On the other hand, machine vision systems start from images and must work towards scenes through several levels of abstraction. A symbiotic system must make the proper connections between the disparate types of information possessed by the human and the machine. The symbiotic system must move up several levels of abstraction to be able to properly communicate with a human user. This is also necessary if one hopes to incorporate expert knowledge in a system (for example, to be able to recognize trees in an aerial photograph as spruce trees), since experts generally work at more levels of abstraction than novices.

Of the two systems described, MISSEE maintains a richer knowledge base and allows communication at more levels than does the NMO program. It generates a semantic network that has an explicit specialization hierarchy. Schemata control the interpretation process, via the cycle of perception, and because they reflect their positions within the hierarchy, they can interact with the user at an appropriate level within the range of possibilities. The system for removing the effects of NMO has only a single level of abstraction, that of the signal. Since the task is concerned with an aspect of seismic data processing rather than seismic image understanding, that is sufficient.

Information exchange

The medium for communication in this area is often pictures. But the issue of what to depict, how best to highlight important features, when to change the display, and how to let the user impart information is not well understood. Hardware capabilities are constantly improving and advances are being made in ergonomics, but less work has been done on information content and how best to mediate its flow between man and machine. Part of this issue is the question of control of the processing: Is it being directed by the program or the user, or do they take turns?

MISSEE provides for several forms of input, but the interaction is generally restricted. The user can draw an initial sketch map, but all interactive communication is textual. The user responds to graphic displays and protocols by manipulating the messages on the priority queue or by accessing the semantic network through the schemata manipulation routines. The user is unable to interact with the graphic objects. In the NMO system, however, the user has a large range of graphics interaction capabilities available. Structured manipulation of graphics entities with feedback is the standard mode of operation in that system. Picking and positioning enable the user to create graphs and overlays that communicate the necessary processing parameters to the system. A command menu lets the user choose the way that processing should proceed. It is a more user-oriented form of communication than is found in MISSEE.

The two issues raised above are comparable to questions of descriptive and procedural adequacy: (1) Is the data description adequate to capture the knowledge supplied by the user, and is it sufficient to properly incorporate it with the information derived from the sensory input? (2) Can the system adequately manipulate the data so that it can present the appropriate facts to the user and then make use of the advice provided in return?

There are many approaches that may be appropriate for symbiotic vision systems. This paper presented examples of a schema-based and an object-oriented system. Rule-based and constraint-based systems are other AI architectures that could be used. From a system design viewpoint the crucial aspect is not the architecture but the power and usefulness of the knowledge representation scheme and its ability to capture the necessary forms of knowledge for a given application. Both of the example systems described, although quite different in nature, are adequate for the tasks they were designed to handle.

REFERENCES

1. D. B. Guralnik (Editor), *Webster's New World Dictionary* concise edition, p. 752. Nelson, Foster & Scott, Toronto (1964).
2. H. G. Barrow and J. M. Tenenbaum, Representation and use of knowledge in vision. *Sigart Newsletter* **52**, 2–8 (1975).
3. G. J. Agin and R. O. Duda, SRI vision research for advanced industrial automation. *Proc. Second USA–Japan Computer Conf.*, pp. 113–117. Tokyo (1975).
4. H. G. Barrow and J. M. Tenenbaum, *Msys: A System for Reasoning About Scenes*. SRI, AI Center, TN 121 (1976).
5. G. Stockman, *Toward Automatic Extraction of Cartographic Features*. L.N.K. Corp., ETL-0153 (1978).
6. H. P. Nii, E. A. Feigenbaum, J. J. Anton and A. J. Rockmore, Signal-to-symbol transformation: HASP-SIAP case study. *AI Mag.* **3**, 23–25 (1982).
7. R. W. Weyrauch, Prolegomena to a theory of mechanized formal reasoning, in *Readings in AI* (Edited by B. L. Webber and N. J. Nilsson), pp. 173–191. Tioga, Palo Alto (1981).
8. J. M. Tenenbaum, *On Locating Objects by Their Distinguishing Features in Multisensory Images*. SRI, AI Center, TN 84 (1973).
9. J. Glicksman, *A Cooperative Scheme for Image Understanding Using Multiple Sources of Information*. U.B.C., Dept. of Computer Science, TN 82–13, November (1982).
10. E. Rich, *Building and Exploiting User Models*. CMU, Dept. of Computer Science, CMU-CS-79-119 (1979).
11. J. Glicksman, A schemata-based system for utilizing cooperating knowledge sources in computer vision. *Proc. Fourth Conf. CSCSI*, pp. 33–40. Saskatoon, Canada (1982).
12. J. Glicksman, Procedural adequacy in an image understanding system. *Proc. Fifth Conf. CSCSI*, pp. 44–49. London, Canada (1984).
13. W. S. Havens and A. K. Mackworth, Schemata-based understanding of hand-drawn sketch maps. *Proc. Third Conf. CSCSI*, pp. 172–178. Victoria, Canada, May (1980).
14. W. S. Havens, *A Procedural Model of Recognition for Machine Perception*. U.B.C., Dept. of Computer Science, TR-78-3, March (1978).
15. U. Neisser, *Cognition and Reality: Principles and Implications of Cognitive Psychology*. W. H. Freeman, San Francisco (1976).
16. A. K. Mackworth, Vision research strategy: black magic, metaphors, mechanisms, miniworlds and maps, in *Computer Vision Systems* (Edited by A. R. Hanson and E. M. Riseman), pp. 53–61. Academic Press, New York (1978).