State-space search for high-level control of machine vision

Shu-Yuen Hwang
National Chiao Tung University
Department of Computer Science and Information Engineering
1001 Ta Hsueh Road
Hsinchu 300, Taiwan

Abstract. Computer vision is a task of information processing that can be modeled as a sequence of subtasks. A complete vision process can be constructed by synthesizing individual operators performing the subtasks. Previous work in computer vision has emphasized the development of individual operators for a specific subtask. However, the lack of knowledge about other levels of processing, while developing the operators for a specific level, makes the development of a robust operator and thus a robust system unlikely. To obtain vision problem-solving methods that are robust in the face of variations in image lighting, arrangements of objects, viewing parameters, etc., we can simply incorporate all possible sequences of image-processing operators, each of which deals with a specific situation of input images; then an adaptive control mechanism such as a state-space search procedure can be built into the methods. Such a procedure dynamically determines an optimal sequence of image-processing operators to classify an image or to put its parts into correspondence with a model or set of models. One critical problem in solving vision problems with a state-space search model is how to decide the costs of paths. This paper details the state-space search model of computer vision as well as the design of cost functions in terms of information distortions. A vision system, VISTAS, has been constructed under the state-space search model and its parallel version has been simulated.

Subject terms: computer vision; image processing; state-space search; vision algorithm synthesis.

Optical Engineering 31(6), 1264–1276 (June 1992).

1 Introduction

Computer vision is a task of information processing. The goal of the task, as described by Marr,[1] is discovering from images what is present in the world, and where it is. Because of the existence of variations in the world, such as noises, image lighting, arrangements of objects, and viewing parameters, it is usually difficult to model the distributions of images, and therefore hard to construct a theory for practitioners to follow. The lack of theory in computer vision implies that there is no optimal or robust solution for most problems. One can justify the validity of a method by its empirical but not its theoretical behavior. Furthermore, any empirically optimal or near-optimal operator at a specific level of processing does not guarantee to give its successive level of processing a best input. For example, almost all existing methods for thresholding are based on heuristics which usually ignore the knowledge about other levels of processing.[2] How useful these heuristics are must be observed during experiments on certain domains of images, and the observation is applicable only to these domains. Thus, the best method for thresholding is unknown.

Although the literature has covered various methods, algorithms, and heuristics for solving particular steps of vision problems, any application of computer vision generally involves more than one step of processing. Developing a reliable algorithm for any application is therefore more difficult than for a specific aspect of vision because the theory and heuristics necessary for coordinated multiple-step sequences of vision operations are not fully explored, in addition to the lack of robust operators.

This paper presents a methodology for solving vision problems. Instead of searching optimal or robust operators at each step of processing, the methodology uses state-space search procedure to dynamically determine an optimal sequence of image-processing operators to classify an image or to put its parts into correspondence with a model or set of models. The optimality is defined in terms of least information distortion. The robustness can be achieved by accommodating all possible sequences of image-processing operators into the search space, each of them can deal with a specific situation. By providing guidelines for the use of state-space search in computer vision, we hope to simplify the development of vision algorithms and broaden the scope of problems these methods can handle.

The structure of this paper is as follows. Section 2 discusses the state-space search model of computer vision. Section 3 discusses the design of cost functions based on information distortions, which is the heart of our approach.

Paper 2989 received Oct. 4, 1990; revised manuscript received Oct. 24, 1991; accepted for publication Nov. 11, 1991. This paper is a revision of a paper presented at the SPIE conference Machine Vision Systems Integration in Industry, November 1990, Boston, Mass. The paper presented there appears (unrefereed) in SPIE Proceedings Vol. 1386

Section 4 discusses an experimental system VISTAS which implements the ideas discussed in the previous sections. Section 5 is devoted to parallelizing the state-space search within the context of VISTAS. The last section is the conclusion.

2 State-Space Search Model of Computer Vision

2.1 *Vision Problems and Algorithms*

The process of computer vision can be modeled as a canonical sequence of subtasks[3–5]:

1. *Preprocessing*. The preprocessing phase removes uninteresting patterns and noise from the image. Preprocessing is performed under the assumption that an image is composed of an informative pattern modified by uninteresting variations. Examples of operators in this phase are filters.
2. *Segmentation*. The segmentation phase can be further divided into two sub-steps.
   Labeling. The labeling phase decides the attributes of pixels and is performed under the assumption that an informative pattern is a connected set of pixels. Examples are edge detection, thresholding and corner finding.
   Grouping. The grouping phase collects connected sets of pixels with the same attributes. The grouping operations perform a logical change of data structure from pixels to symbolics. Examples are region growing and edge linking.
   These two steps work closely and therefore are referred to as parts of a single step.
3. *Interpretation*. Interpretation gives each group of pixels a semantic meaning. It usually contains two substeps.
   Extracting. The extracting operation computes for each group of pixels a list of its properties. Examples of properties are area, centroid, and spatial moments. Which properties are interesting depends on the world model.
   Matching. The matching operation determines the interpretation of some related set of image events and associates these events with some given 2D shape. There is a wide variety of matching operations, such as template matching and computing the Euclidean distance of property vectors.

Any successful vision system can perceive only a limited number of data.[3] A computational-complexity based analysis of visual search problem showed the intractability of the bottom-up approach.[6] Therefore, we assume that all steps in the process are performed within the context of a world model which specifies what the vision process can see at each of the various positions of an image.

One approach to solving the vision problem is to divide it into several steps corresponding to the vision model, and then to select appropriate operators at each level and synthesize these operators. How operators in each step are selected and how these operators are coordinated need to be solved in this approach.

Generally speaking, the issue of operator selection is domain dependent. Most literature in computer vision talks about designing an operator for specific steps; few researchers deal with the issue of operator coordination. The second issue is the concern of our research. We want to present a general approach to coordinate the operators.

2.2 *From Vision to State-Space Search*

State-space search has been explicitly used in some specific aspects of computer vision. Heuristic search was applied to contour tracing by Martelli.[7] Other researchers[8,9] have applied search methods to image segmentation problems. An extension of a graph-matching algorithm using state-space search was developed by Shapiro and Haralick.[10] The interpretation tree introduced by Grimson et al.[11,12] is a typical model-based approach for three-dimensional object recognition. The ARGOS system developed by Rubin[13] was able to interpret certain images of downtown Pittsburgh. Our work is much different because we intend to handle the complete process of computer vision.[14,15]

A vision problem can be transformed into a state-space search problem by changing its canonical procedure into the form of a state-space search problem. Assume the problem has an input image I and a set of operators. Each operator belongs to one of the five categories mentioned above. A state can be regarded as a representation of the input image using some format. The start state represents the input image I in the format of a two-dimensional array. The state space can be derived from the start state and the set of operators.

Formally, we can define the state-space search problem $\langle s, T, K, G\rangle$ as follows.

Start state.

$$s = I \tag{1}$$

Operators. The set of search operators is the set of image operators:

$$T = t_1 \cup t_2 \cup t_3 \cup t_4 \cup t_5 \tag{2}$$

in which

$$t_1 \subseteq \{\text{preprocessing operators}\}\ , \tag{3}$$

$$t_2 \subseteq \{\text{labeling operators}\}\ , \tag{4}$$

$$t_3 \subseteq \{\text{grouping operators}\}\ , \tag{5}$$

$$t_4 \subseteq \{\text{extracting operators}\}\ , \quad \text{and} \tag{6}$$

$$t_5 \subseteq \{\text{matching operators}\}\ . \tag{7}$$

State space. All states except the start state are derived from their parents and the operators leading to them. An exception is that in some domains the set of goal states can be known beforehand; this point of view is used in our model for the reasons discussed in the previous subsection. Thus,

$$K = k_1 \cup k_2 \cup k_3 \cup k_4 \cup k_5 \cup k_6 \tag{8}$$

in which

$$k_1=\{s\} \ , \tag{9}$$

$$k_2=t_1(k_1)\subseteq\{\text{preprocessed images}\} \ , \tag{10}$$

$$k_3=t_2(k_2)\subseteq\{\text{labeled images}\} \ , \tag{11}$$

$$k_4=t_3(k_3)\subseteq\{\text{grouped images}\} \ , \tag{12}$$

$$k_5=t_4(k_4)\subseteq\{\text{property vectors}\} \ , \quad \text{and} \tag{13}$$

$$k_6=t_5(k_5)\subseteq\{\text{interpretations}\} \tag{14}$$

where $t_i(k_i)$ is the range of operators t_i having k_i as its domain.

Goal states.

$$G=k_6 \tag{15}$$

This transformation makes one simplification of the vision process. It assumes that an operator is a function from a representation of an entire image to another representation of the entire image.

Some variation to this simple model may occur. First, some operators do not transform an image to its next stage. An example is mathematical morphological operators that transform a labeled image to another labeled image. It is not difficult to modify the formalization to accommodate loops. Suppose that state k_i contains a self-loop. Then k_i can be defined as

$$k_i=\bigcup_{j=0}^{\infty} k_i^j \tag{16}$$

in which

$$k_i^0=t_{i-1}(k_{i-1}) \ , \quad \text{and} \tag{17}$$

$$k_i^j=t_i(k_i^{j-1}) \quad \text{for } j>0 \ . \tag{18}$$

The first set k_i^0 represents the set of states derived from the parent state of k_i directly; all the other sets are obtained by applying t_i to the set of states that were derived previously. A self-loop generates a potentially infinite number of states during the search.

This variation can lead to several variations. First, each k_i^j can be a different group of states. In this case, states are classified based not only on their type of representation but also on how they are derived. Second, the number of times in which t_i is applied can be limited.

A very common situation is that an operator itself uses a search process.[7,10] We can refine the granularity of the state space on a higher level to reflect this fact. For example, Shapiro's pattern match is an operator in t_5. Suppose an operator t conducts a search process for a subproblem which is expressed as $\langle K', T', s', G'\rangle$. It is easy to accommodate this subproblem to the entire framework by letting a new state space be the union of the two existing state spaces with new operators being $T\cup T'-t$. The start state and goal states remain the same. Note that it is essential that the start state s' and the goal states g' have a conceptually equivalent entity with states in K; for example, if each state in K contains the information of I, then so do s' and G'.

The formalizations only specify the transformation of a vision problem from one model to another. Its feasibility needs more elaboration. The following make vision problems significant compared to other search problems:

Heterogeneous states. Clearly, the state-space K is not homogeneous, at least at the level of the representation of states. On one extreme, the start state s is a two-dimensional array of pixels. On the other extreme, the output is in the form of symbolic descriptions (depending on the application and implementation). The intermediate states have forms between these two extremes. This characteristic violates one of the reasons of the success of previous studies, a uniform representation of states.

Heterogeneous operators. The set T includes various image operators. Different operators need different inputs and have various degrees of computational complexity. The characteristic also violates the homogeneity.

Causal orders between operators and states. Although operators and states are heterogeneous, there exist certain relations between operators and operators, operators and states, and states and states. Operators must be applied in certain orders and have certain sets of states as their inputs and outputs.

An immediate observation is that the state-space search model is not strong enough to completely specify vision problems. On the other hand, we can utilize the characteristics of vision problems to deal with some of the concerns of state-space search problems such as the combinatorial explosion. The modified model is described in the next subsection.

2.3 *Algorithm Graph*

The state-space search model can be modified with a new component to specify the characteristics of vision algorithms. A requirement of this new component is that it must reflect the causal relation and heterogeneity of vision algorithms. Because it is convenient to specify these characteristics using a graph-like structure, the term *algorithm graph* is used to denote such a structure. A node in an algorithm graph corresponds to a set of states; an arc corresponds to a subset of the operators.

Formally, let $A=\langle V, E\rangle$ be an algorithm graph with vertices $V=\cup_i v_{k_i}$ and edges $E=\cup_i e_{T_i}$. Each vertex v_i denotes a set of states. A directed edge $e_{T_j}=(v_{k_i}, v_{k_j})$ from node v_{k_i} to node v_{k_j} is a set of operators T_j that have k_i as their domains and k_j as their ranges.

Suppose k_i is the set of states represented by v_{k_i}; then the following properties hold:

$$K=\bigcup_i k_i \tag{19}$$

$$k_i\cap k_j=\emptyset, \quad \text{provided } i\neq j \ . \tag{20}$$

Note that the operators in the set are not necessarily classified into disjoint subsets by the algorithm graph. It is possible for an operator to be used for different states.

Now a vision problem V_S in terms of the state-space search model can be completely characterized by a vector

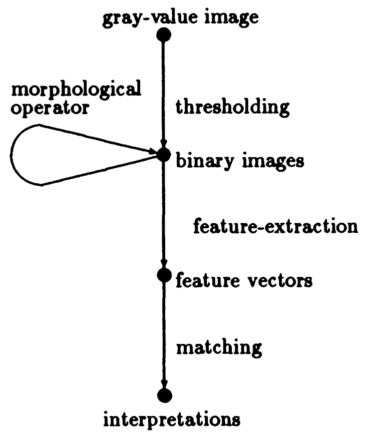


Fig. 1 Example of an algorithm graph.

with five components: start state s, operators T, state space K, goal states G, and an algorithm graph A. Or

$$V_S = \langle s, T, K, G, A \rangle \ . \tag{21}$$

In this formalization, the information contained in a state is more than the image data. The corresponding node in the algorithm graph is another piece of information contained in a state.

An algorithm graph may be regarded as a problem-solving process for a given vision problem. It specifies the order that the operators must follow. A very general example of an algorithm graph is shown in Fig. 1. This graph represents a sequential problem-solving process that captures the essential steps of image understanding. Besides specification, an algorithm graph serves another important purpose: It represents a move generator for the given vision problem. Because any state has associated information about the corresponding nodes in the algorithm graph, we can apply the class of operators corresponding to the arcs emanating from the node. It is necessary to associate with each state a status to indicate which node in the algorithm graph to which the state corresponds. This association is required because sometimes we cannot tell the status of a state from the representation of the image only. For example, the same binary image might have a different status and involve the use of different operators to generate its successors. This situation occurs when a specific algorithm graph is used rather than a general one.

Conceptually, any vision problem has a most general algorithm graph that is implied by the causal relations between operators. Since each operator has its specific range and domain, the most general algorithm graph can be obtained by gathering all operators with the same domain and range together. Besides the two properties of all algorithm graphs, the following property also holds for the most general graphs:

$$e_{T_i} \cap e_{T_j} = \emptyset, \quad \text{provided } i \neq j \ . \tag{22}$$

The property states that operators are classified into disjoint subsets corresponding to vertices in a most general algorithm graph.

The most general algorithm graph gives no information except the data types accepted and generated by operators. A more specific algorithm graph can reveal more structure of the problem. A user can specify an algorithm graph to be used by the system, or alternatively, the system can try to solve a problem through the most general algorithm graph. The selection of individual operators at each state is controlled by the search strategy as well as the cost function defined in the previous subsection. In specifying the algorithm graph, the user can use heuristics or domain knowledge to select the operators. For example, planning[16] and Bolles' operator-suggestion mechanism[17] are good methods that can be used in the stage of algorithm specification.

3 Information Distortion as Cost Function

The design of cost functions plays a key role in modeling vision problems as state-space search problems. Cost functions are defined based on the information distortions. Details are discussed in this section.

3.1 *Why Information Distortion?*

Computer vision is a process of information retrieval and transformation. The output of an image recognition process should present what the input image has. The only difference between the input and output is that they use different representations of the information. Unfortunately, during the course of information transformation, some distortions will occur. Consider the preprocessing phase. Noises are removed under the situation that no information about the image has been retrieved. The result is that interesting patterns are modified, although the intention is to remove noises. Another example is the matching phase. We usually need to guess which model most resembles the data because an exact match is unlikely.

Since we want the output to have the exact information that the input has, the best way to judge the appropriateness of different outputs is to compare their information difference from the input. The information difference between an output and the input comes from two sources. The first category comes from the changes in image content made by operators on the input image. The second category is the difference between models and data. Both categories of information difference are referred to as information distortion and are treated uniformly in the design of cost function.

Cost functions are ideally defined in terms of the information distortion introduced by an operator or a sequence of operators. However, there are two major difficulties in this approach:

1. *Irrelevance of information.* Because computer vision itself is a task of information processing, operations at earlier stages of computer vision are usually done without the knowledge of what the final interpretation will be. It is therefore impossible to design cost functions that are applicable at these stages and that are based on the correct interpretation, an interpretation that cannot be known until the last step of the vision process has been completed. This difficulty is referred to as the "potential" irrelevance of information because one attempts to find useful information at the steps where that information is not immediately available.

2. *Heterogeneity of information*. The second difficulty comes from the diversity of information sources in computer vision. On one end we have images in the form of two-dimensional arrays. On the other end, we have the representations in the form of symbolic structures. Furthermore, an inappropriate granularity of operators may result in states that represent different objects and it is difficult to define the difference between a pixel and an object. It is even meaningless to compute the difference of two numbers if they represent different ranges of quantities, such as the difference of a gray-value pixel and a binary pixel. It is unlikely that a unique cost function can exist in such a diverse domain. Because it is obvious that we need more than one type of distortion function, the difficulty of heterogeneity is reduced to how different measurements of information distortion are unified.

3.2 *Information Distortion at the Iconic Level*

One method for solving the difficulty of irrelevance of information is to delay the computation of the information distortion occurring at the preprocessing and labeling steps to later steps.

3.2.1 *Delayed computation*

Vision itself is a process of information retrieval. We can obtain the information contained in an image only after we process the image. Obviously, if the computation of information is delayed to the last step of machine vision, we can get exact information in the input image. Doing this results in an unrealistic situation: We need to execute several steps to decide the next step. A compromise between this extreme and computing the pure difference pixel by pixel is to compute the information distortion at the grouping step. An appropriate selection of the compromise should allow the decision to be made early enough and should provide an accurate measurement of the information distortion.

The image is segmented into several groups after the grouping step. At this point the attributes of pixels are decided and we can have better measurements of the distortions resulting from the sequence of image operations. For simplicity we assume that each group is a region. Other types of groups, such as edges, can be analyzed in the same way. These groups are used as the information units for computing the information distortion. The information units of the input are defined with respect to the information units of the output.

Formally, let $I(r, c)$ be the very first input image, and let $\Re = \langle R_1 \cdots R_k \rangle$ be a segmentation of the output image of a sequence of operations that have $I(r, c)$ as the first input. Define a segmentation $\Re_I$ on $I(r, c)$ to be the same as $\Re$. That is, after a segmented image is obtained after the grouping step, the input image is segmented using the identical segmentation. The information distortion resulting from preprocessing, labeling, and grouping is computed based on the information units formed after the grouping step. Since the computation can be done only after the image is segmented, the method is referred to as "delayed computation." The information distortion can be defined as

$$\sum_i P(R_i)\delta_{R_i} \ , \tag{23}$$

in which δ_{R_i} denotes a measurement of difference of two corresponding regions, $R_i \in \Re_I$ and $R_i \in \Re$. A single subscript R_i is used because the two regions in both images have the same set of pixels. $P(R_i)$ represents the weight of region R_i in the image.

Before defining δ_R, let's first define an ideal image $\Lambda(r, c)$ as an image in which each pixel has an ideal value. The ideal value of a pixel is defined as the value that can yield the exact attribute used in the grouping phase. How ideal values are computed is discussed in the next subsection.

Several measurements of the function δ_R can be defined from $\Lambda(r, c)$. Two of them are the total difference T_R of a group and the average difference A_R of the group. The total difference of a group R is defined as the summation of the square of the differences between $\Lambda(r, c)$ and pixels belonging to same group in $I(r, c)$. The average difference is the total difference normalized by the size of the group. Formally, they are defined as

$$T_R = \sum_{(r,\ c) \in R} [I(r, c) - \Lambda(r, c)]^2 = \sum_{(r,\ c) \in R} [d(r, c)]^2 \ , \text{ and} \tag{24}$$

$$A_R = \frac{T_R}{|R|} \ , \tag{25}$$

in which $|R|$ is the size of R, or the number of pixels in the group $|R|$.

Intuitively, the definition is based on the assumption that the groups used after the grouping step have their values presented on the very first input image. However, these values are idealized so they can be processed by the later steps. The machine "hallucinates" some ideal groups that are not presented by the very first input image. For example, if the foreground pixels should have their gray values higher than 200, then all values higher than 200 are ideal. Some preprocessing procedures might increase some pixels to the ideal value, or the grouping phase might merge some pixels with gray values lower than 200 to a larger foreground region. No matter what the values are after grouping, the machine "sees" an image $I(r, c)$, which is the input and "hallucinates" an image $\Lambda(r, c)$, which is an ideal image.

An example is shown in Fig. 2. In this figure, (a) is the input image and (b) is the image after processing. A segmentation of the image is shown in (c) with ideal values in two regions. In this example, we assume that all pixels in region R_1 have the ideal value 7, pixels at region R_2 have the ideal value 6, and pixels at region R_3 have the ideal value 0. The differences between pixels of $I(r, c)$ and $\Lambda(r, c)$ are shown in (d). The total differences and average differences of all groups are

$$T_{R1} = 10(1) + 4 + 4 + 4 = 22 \tag{26}$$

$$A_{R1} = \frac{22}{27} \tag{27}$$

$$T_{R2} = 0 \tag{28}$$

$$A_{R2} = 0 \tag{29}$$

$$T_{R3} = 1 + 9 + 49 = 59 \tag{30}$$

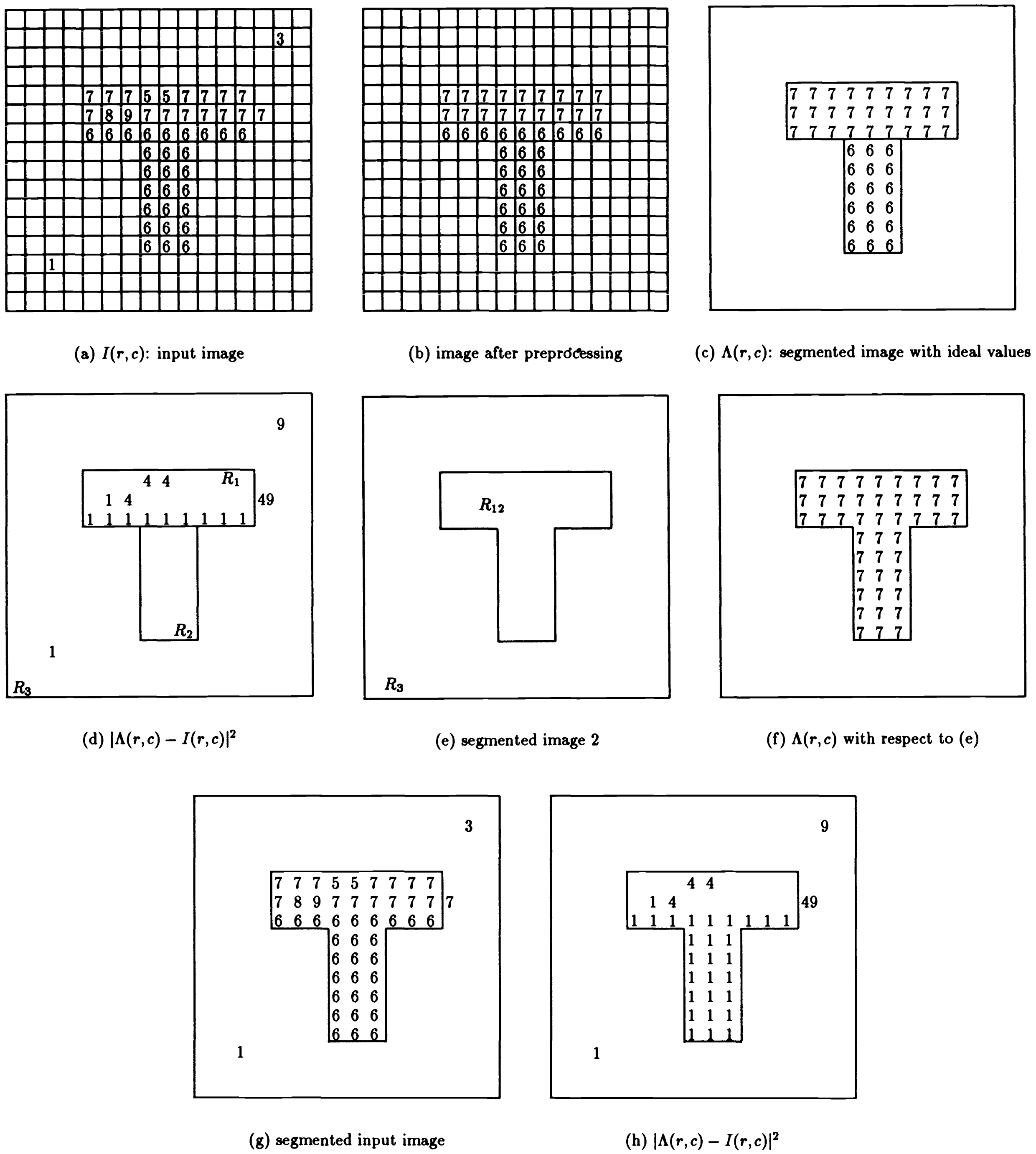

(a) $I(r,c)$: input image
(b) image after preprocessing
(c) $\Lambda(r,c)$: segmented image with ideal values
(d) $|\Lambda(r,c) - I(r,c)|^2$
R_1
R_2
R_3
(e) segmented image 2
R_{12}
R_3
(f) $\Lambda(r,c)$ with respect to (e)
(g) segmented input image
(h) $|\Lambda(r,c) - I(r,c)|^2$

Fig. 2 Example of information distortion at the iconic level; a blank denotes 0.

$$A_{R3} = \frac{59}{211} \ . \tag{31}$$

The average difference is a better measurement of information distortion than the total difference because it has been normalized by the size of the group. The total difference T_R does not provide any useful measurement because it is actually the pure difference pixel by pixel. However, we show later that its probability distribution is a useful measurement.

Notice that the groups used in defining the information distortion are the groups input to the extracting step. Whenever there is a change of groups, $\Lambda(r, c)$ as well as the distortion must be recomputed. For example, in the same figure, (e) is another possible segmentation, for which the ideal values of pixels are shown in (f). Because regions may be merged or split several times in the grouping phase, the ideal values may vary. If (e) is used, then R_{12} is the group (region) used for the extracting step and T_{R12} or A_{R12} should be used. A new $\Re_I$ is shown in (g), and (h) is the difference between $I(r, c)$ and the new $\Lambda(r, c)$. Based on these values, we have

$$T_{R12} = 4 + 4 + 4 + 28(1) = 40 \tag{32}$$

$$A_{R12} = \frac{40}{45} \ . \tag{33}$$

T_{R3} and A_{R3} remain the same.

Although regions are used to derive the information distortion throughout this section, we want to point out that this approach also applies to other types of groups such as edges. As long as we can decide the ideal value of a pixel, we can calculate the difference between the data and its ideal model. This approach is practical because most vision processes have only one operator for each of the steps of preprocessing, labeling, and grouping. The delay should not cause many unnecessary searches.

3.2.2 *Deciding ideal values*

An assumption used in the previous subsection is that the ideal values of pixels are known. This subsection is devoted to deriving ideal values. The idea behind the use of ideal values is to dynamically decide the models of groups. In Martelli's edge detection,[7] low curvature and high contrast are the models of edges. The models of groups are hard to define statically because they vary with the standard to evaluate the groups as well as the grouping methods.

There are two cases in the decision of ideal values, *direct mapping* and *spectrum analysis,* which needs more elaboration.

Direct mapping. In some situations, the attribute of a pixel is defined in terms of a numerical function. The ideal value of a pixel can be derived exactly in these cases. An example is the facet model of image,[18] which declares that the ideal value of a pixel is a linear function of its row and column:

$$\Lambda(r, c) = l \times r + m \times c + n \tag{34}$$

in which l, m, and n are constants. One special case of the facet model is that in which all pixels in a region have an identical value:

$$\Lambda(r, c) = k \tag{35}$$

in which k is a constant.

Spectrum analysis. Direct mapping is not very common in computer vision. In most situations the attributes of pixels are defined symbolically. We need to define a mapping from these symbolic terms to their corresponding numerical values. For example, suppose thresholding is used in labeling pixels, and the image is segmented by grouping foreground pixels and background pixels. The attributes used in the grouping phase can be interpreted as "dark" and "bright." We need to decide the corresponding gray values of the two attributes in order to compute the distortion. In general, the mapping is one-to-many: An attribute corresponds to a range of numerical values. Pixels that fall in the range and are labeled with the corresponding attribute are considered ideal and contribute no distortion.

The approach is to analyze the range of the mapping, or the spectrum of corresponding numerical values. In the previous example, the histogram of the gray values can be analyzed and used to decide which range contributes to dark pixels and which range contributes to bright pixels. This decision process is called *quantification* of attributes.[19] It can be considered an inverse transformation of information processing, as opposed to the vision process. A conceptually similar process is that of computing the projection of three-dimensional objects.

The steps for deciding the ideal values of a set of two attributes are as follows:

1. Construct the spectrum of numerical values from the data.
2. Detect the valleys of the spectrum. A valley is a flat range between two local maxima.
3. Use the lower (higher) bound of the valley as the ideal value of the symbolic terms assigned to the range of numerical values left (right) of the valley.

An example is shown in Fig. 3. The figure is a hypothesized spectrum of gray values used for quantifying two attributes, dark and bright. The spectrum has two obvious peaks and a valley. The ideal value of the bright pixels that are left of the valley is 45, while the ideal value of the dark pixels is 50. If a pixel is grouped in a foreground object and its gray value is larger than 45 then it will contribute positively to the amount of distortion measured.

Windowing is necessary in any complex scene to construct an appropriate spectrum, because different areas in a complex scene usually have the same attributes mapping to different ranges of numerical values. A similar approach is to use a mask to decide thresholding values for different parts of an image.[20]

3.3 *Information Distortion at the Symbolic Level*

Methods used to measure the information distortion occurring at the symbolic level are highly domain dependent. Distortions occur when a model is used to interpret the data and there are differences between the model and the data. These differences contribute to the information distortion. Although the idea is straightforward, the measurement is not because we need to use numerical values to represent the differences between two symbolically described objects. Moreover, the differences appear at all levels of the representation of objects, such as component, structural, and relational differences.

The measurement of differences certainly depends on the representation of objects. The simplest representation of an object is a vector of features, and therefore the information distortion can be defined as a weighted-Euclidean distance between two vectors. A system that uses such measurement is the ARGOS system.[13]

Feature vectors are not enough for computing the differences between two objects with complex shapes. Objects with complex shapes are usually described by primitive components and their relationships. To compute the difference, both the feature vectors of components and the relationships between components are considered. A similar example is the structural description of objects and the inexact match between them.[10] The degree of inexactness is

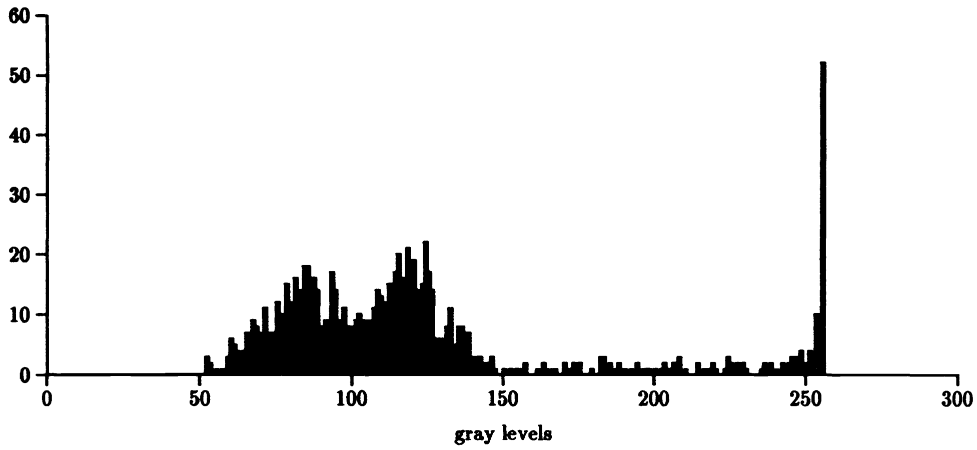


Fig. 3 A hypothesized spectrum of gray values used for attributes *bright* and *dark*.

computed by summing the differences between two objects. The differences consist of the following:

1. The summation of differences between primitive components.
2. The errors of missing components. When two objects are compared, it is possible that a primitive component of one object does not have a corresponding component on the other object.
3. Relational errors, which have only two possibilities: match or no match. Two objects match and have no relational error if the errors of corresponding components and missing parts are smaller than some thresholds.

A cost function can be designed by modifying the relational errors to have a value 0 if the summation of component error and missing component error is smaller than a threshold, and to have an arbitrary large value otherwise.

The measurement of information distortion in interpreting images with more than one object should include two more quantities:

1. *Adjacency constraint.* The interpretation of one region as being an object may affect the interpretation of its adjacent regions. For example, when a region is interpreted as a river then it is unlikely that its adjacent regions will be sky in the domain of outdoor scenes. A high cost must be added to the region interpreted as sky when its adjacent region has been interpreted as river.
2. *Location constraint.* When the world model is restricted, certain portions of the image cannot be interpreted as some objects. An example of impossible interpretation that violates the location knowledge of the world model is a region on the bottom of an image being interpreted as sky. In this case a high cost must be added to the cost function.

Nearly all vision systems use a model-based approach in the high-level vision part. To compute the information distortion at this level is to compute the difference between the data and its interpretation, which is a model known to the system. The computation depends completely on the representation methods of the models.

3.4 *Unification of Cost Functions*

Different categories of information distortions need to be unified to have a simple control mechanism in the state-space search model.

3.4.1 *The approach*

One approach for unification is to find each of the probability distributions. If the distributions can be found, each kind of information distortion can be normalized to a number between 0 and 1. Let X denote the random variable of information distortion introduced by image operation T. Suppose that the information distortion is x. Because the distribution of X is known, the probability that $X>x$ can be found. This value can be interpreted as the "likelihood of similarity" of the output O_T of T with respect to its input I_T. Therefore, we can define the *confidence* level of O_T with respect to I_T as $P(X>x)$, where $P(y)$ is the probability of the event y.

A benefit of using probability-like confidence levels is that traditional reasoning methods such as Bayesian decision theory[21] have developed to a rather mature level in providing the probability of correct interpretation at the pattern recognition level. Two types of probabilities can be easily combined. Instead of using the weighted sum $\Sigma_i w_i d_i$ of information distortion as the cost of paths, we can define the confidence level of the path as $\Pi_i C_i$, in which d_i and C_i are information distortion and confidence level of arc i, and w_i is a weighting factor. The computation of the probability of correct interpretation is easier than that of the probability of errors or weighted sum of errors. Various confidence values are derived in the following.

3.4.2 *Confidence values at the iconic level*

Let us assume that each $d(r, c)$ [see Eq. (24)] has a normal distribution with mean 0 and variance σ^2, denoted as $n(0, \sigma^2)$. Based on this assumption, the distribution of T_R and A_R can be derived. Because $d(r, c)$ has $n(0, \sigma^2)$, $[d(r, c)]^2$ has a gamma distribution with parameters 1/2 and $1/2\sigma^2$ (Ref. 22), denoted as $\Gamma(1/2, 1/2\sigma^2)$. Then, T_R is the summation of $|R|$ independent random variables with $\Gamma(1/2, 1/2\sigma^2)$ distribution; therefore, T_R has a gamma distribution with parameters $|R|/2$ and $1/2\sigma^2$, denoted as $\Gamma(|R|/2, 1/2\sigma^2)$.

The probability distribution of A_R can be derived too. Since

$$A_R = \frac{\sum_{(r,\ c)\in R} d(r,\ c)^2}{|R|} = \sum_{(r,\ c)\in R} \frac{d(r,\ c)^2}{|R|} \tag{36}$$

and $d(r,\ c)/(|R|)^{1/2}$ has a normal distribution with mean 0 and variance $\sigma^2/|R|$, the random variable $d(r,\ c)^2/|R|$ has a gamma distribution with parameters 1/2 and $|R|/2\sigma^2$, denoted as $\Gamma(1/2,\ |R|/2\sigma^2)$. Finally, $\Sigma d(r,\ c)^2/|R|$ has a gamma distribution with parameters $|R|/2$ and $|R|/2\sigma^2$, denoted as $\Gamma(|R|/2,\ |R|/2\sigma^2)$, which is the distribution function of the average difference A_R.

Now we can give the probability that R is ideal, or equivalently, the confidence level for the image operations that transform R to its ideal. Once the value of T_R or A_R has been computed, let this value be x. The probability that $T_R \leq x$ can be found; the confidence level of the hypothesis that T_R is 0, or R is perfect, will be $1-P(T_R \leq x)$, or

$$\text{confidence of } R = \begin{cases} 1-P(T_R \leq x) \\ 1-F(x) \end{cases} \tag{37}$$

in which $F(x)$ is the cumulative distribution function of x.

Because the gamma function is actually a family of functions and their values cannot be computed by evaluating a formula, a further simplification is required. The value T_R/σ^2, which is the total difference normalized by the factor σ^2, can be used if we assume that the variance σ is fixed in a certain domain, that is, T_R/σ^2 is a measurement of distortion. Its distribution is simpler than T_R and A_R. The random variable $d(r,\ c)/\sigma$ has a standard normal distribution with mean 0 and variance 1, or $n(0,\ 1)$. The square of $d(r,\ c)/\sigma$ has a gamma distribution with parameters 1/2 and 1/2, or $\Gamma(1/2,\ 1/2)$. The summation of the normalized pixel distortion over the entire group having $|R|$ pixels has a gamma distribution with parameters $|R|/2$ and 1/2; this special case of the gamma function has a chi-square distribution with degree of freedom $|R|$, or

$$\Gamma\left(\frac{|R|}{2},\ \frac{1}{2}\right) = \chi^2(|R|)\ . \tag{38}$$

Figure 4 gives T/σ^2 versus group size under various confidence levels. The region (0 region) above the curve denoting the confidence level 0 is the region in which all points have the confidence level 0. All points in this region are small groups having large distortion. Similarly, the region (1 region) below the curve denoting the confidence level 1 is the region in which all points have the confidence level 1. All points in this region are large groups having small distortions.

3.4.3 *Confidence values at the symbolic level*

Various methods can be used to compute the confidence levels for transformed images at the symbolic level. Examples of these methods include the use of Dempster-Shafer methods[23] and Bayesian decision rules.[21] Both methods compute the degree of confidence in a prediction or guess under some given pieces of evidence. In computer vision, items of evidence are generally features of an image, and the guess is to which model the image belongs.

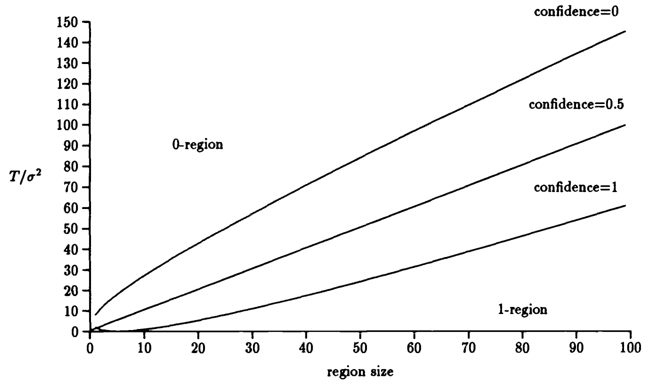


Fig. 4 An illustration of normalized distortion versus region size.

An important assumption in applying these methods is that the distortion resulting from gathering evidence from data can be ignored. This assumption might not be true for two reasons:

1. Operators for feature extraction are not accurate, especially when they discretize the continuous data.
2. The transformation from image data to features is a many-to-one mapping; information is lost during the transformation.

These distortions can be ignored under the condition that the number of training instances used to construct the models is large enough to cover all possible outcomes. Instead of using a single model for a known object, the Bayesian approach constructs the *a priori* probabilities by considering as many instances as possible. All possible outcomes of features should be taken into account. We assume that the distortions resulting from evidence collection or feature extraction can be ignored. That means the confidence level of the extracting step is 1.

Confidence values at the scene level depend on the purpose of the vision process. In purposive vision,[16] the goal is to locate an object in an image, and therefore the confidence level would be the same as for a single pattern's recognition as we have presented above. If the goal is to label the entire image, the confidence level will be the product of confidence levels of each region's labeling, multiplied by the numbers representing the modifications of other knowledge, such as that of location and adjacency.

It is sometimes difficult to maintain a pure probability form because the location knowledge and adjacency knowledge are hard to represent as probabilities. In the ARGOS system, for example, the term ''likelihood'' is used instead, because the values used to represent the adjacency knowledge are assigned without justifications.

4 Building a Vision System

A vision system, VISTAS (for VISion as STAte-space Search), is constructed based on the state-space search model.

4.1 *The Philosophy of Design*

It has been pointed out that complex AI systems are often built on intensive knowledge or general methods.[24] The former approach is useful in solving a particular problem but cannot be used as a general problem-solving methodology. Because our goal is to develop a general approach to solving vision problems, the latter method is adopted to design our prototype system.

The success of systems based on general methods must satisfy two requirements:

1. The system must be able to provide problem-solving techniques when it lacks domain knowledge.
2. The system must be easily combined with domain knowledge to have more powerful problem-solving techniques when the domain knowledge is available.

An example of systems based on this principle is SOAR.[25]

The most fundamental problem-solving method in the state-space search model is the generic search procedure. Based on the generic search procedure, some more specific methods such as depth-first search strategy and breadth-first strategy can be constructed and put into the system because these methods need little domain knowledge. Cost functions should be included in the basic general methods, because they are defined in terms of the type of images. Once the cost functions are built in, the best-first search strategy can be put into the system.

The basic strategy of designing a move generator is to have all operators typed, or categorized into classes corresponding to the steps of the vision process. This categorization allows the system to use the most general algorithm graph to expand a search tree. At each step, the control will generate all possible successors of a node by applying operators if the type of the node is consistent with the input type of the operators.

Domain knowledge is consulted through specific algorithm graphs defined by users. When an algorithm graph is specified, it restricts the availability of operators as well as the order in which the operators are applied. The second method of consulting domain knowledge is to parameterize operators. An example is the thresholding operator. When domain knowledge is available, the thresholding values are selected by consulting the heuristic routines based on domain knowledge; otherwise all possible values are used.

4.2 *System Organization*

The VISTAS system is based on the philosophy discussed in the previous subsection. The system is implemented in Kyoto Common Lisp (KCL) under UNIX. It can solve search-based vision problems when little knowledge about the domain is known. The method it uses is to apply all possible operators and parameters to generate a complete search tree under the control of the cost functions.

Figure 5 shows the organization of VISTAS. The heart of the system is the control module which is essentially a generic search routine with cost functions. Various search techniques can be set by users.

The database module has two parts. One is the long-term memory, which stores the models of the world recognized by the system. In the context of character recognition, models consist of feature vectors of characters and their *a priori* probabilities. The second part of the database is the short-term memory, which is the working space when the system is running. Usually, the short-term memory stores a search tree and related information such as an open list and a closed list.

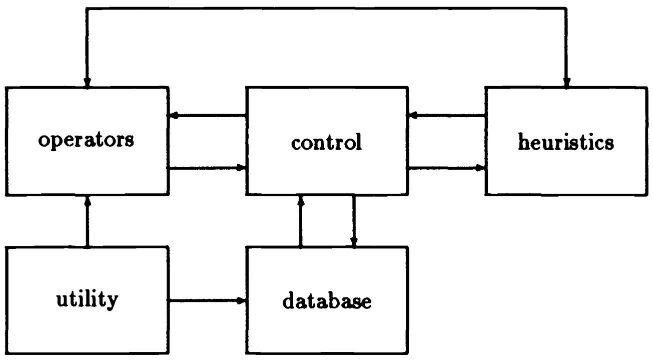


Fig. 5 System organization of VISTAS.

The operators module is a collection of image operators. There are two types of operators. One category of operators can directly be applied as search operators; i.e., their domain and range are consistent with the states used in search. The second category of operators does not correspond with the states either at their range or domain or both. An example of the latter is the histogram computation routine. To distinguish these two categories, the former are called operators and the latter are called functions.

The heuristics module is a collection of methods for operator selection, such as those in Ref. 26. The outputs of these methods are plausible operators or parameters of operators for use in state-space search. The existence of domain-specific knowledge does not affect the functionalities of other components. This knowledge is used to increase the power of the problem-solving techniques. An example of heuristics is the selection of thresholding values based on the histogram analysis.

The utility module is a collection of service routines, including a Lisp interface to WFF file format,[27] some low-level image-processing routines for generating images, and a database interface for computing *a priori* probabilities of models. The system is able to accept an algorithm graph as its generic search tree. The space of search is generated by expanding the generic search tree.

4.3 *Experiments on Character Recognition*

To test the feasibility of VISTAS, we have performed experiments on simple character recognition. Although the domain is restricted, it did explore some characteristics of the state-space search model as well as the VISTAS system.

4.3.1 *Informal description*

The input for the character recognition problem is a gray-value image consisting of a 32^2 array of 8-bit pixels. Each image contains one character, which may occur at one of various sizes. Characters are in the foreground with darker pixels. Figure 6 shows some examples of input images. The output is the classification of the input image.

Characters are selected from the first few pages of the novel *Moby Dick,* published by The New American Library of World Literature in 1961, thirteenth printing. Image data are generated and stored in WFF format,[27] which is written in C. A KCL to C interface has been implemented for calling the WFF package from VISTAS.

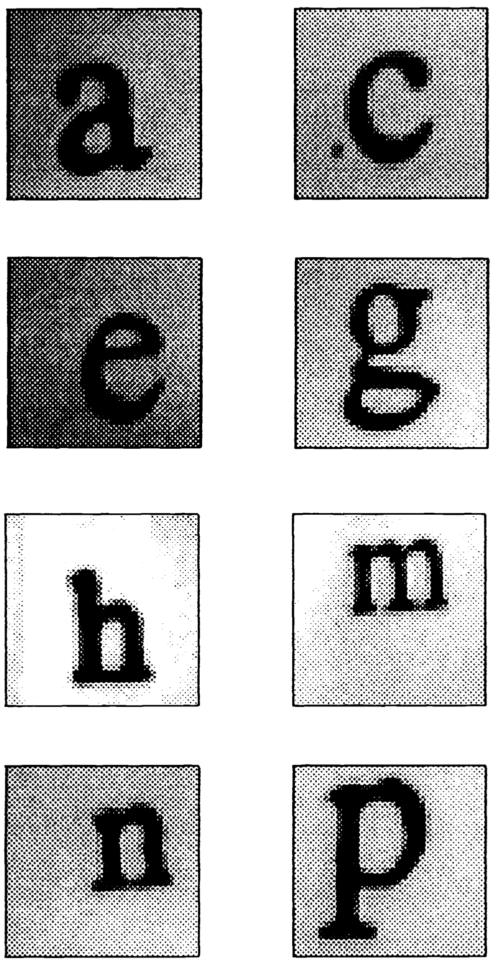


Fig. 6 Examples of input data used in the experiment of character recognition.

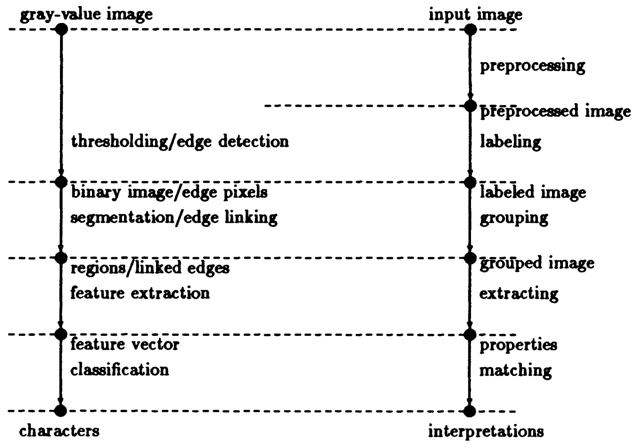


Fig. 7 Character recognition versus computer vision.

Table 1 Experimental results of character recognition (a) without validation and (b) with validation in classification.

	best-first	depth-first	breadth-first
average open nodes at first level	4.2	3.3	6.6
average total open nodes	11.2	7.9	16.9
recognition rate	43/48	41/48	41/48

(a)

	best-first	depth-first	breadth-first
average open nodes at first level	4.1	3.2	5.9
average total open nodes	19.1	15.2	28.2
recognition rate	43/48	40/48	40/48

(b)

4.3.2 *Problem decomposition*

The character recognition problem can roughly be decomposed into several steps, as shown in Fig. 7. The left side is the character recognition problem and the right side is the process of general vision problems, which was described in Sec. 2.

To simplify the experiment, the preprocessing phase is ignored because it is relatively problem independent. Noise removal operations can be done at a higher level when more information about the image has been retrieved. Cost functions are evaluated after a sequence of image operations but not after a single operator has been applied. These situations allow us to perform the experiment without including the preprocessing step.

Because the classification is based on the geometry of the character, either edge detection or region growing can serve as the operator for segmentation. Therefore, the labeling phase can be done by labeling pixels as either edge and nonedge or dark and bright.

The extracting phase is effected by feature-extraction routines. The matching phase is served by matching the feature vector of data to prestored feature vectors. In VISTAS the extracting phase and the matching phase are combined for computational efficiency.

From the comparison of the character recognition problem and the process of computer vision, we can see that the former follows most steps of the latter, which is what we want to explore. What the character recognition problem does not present is mostly in the matching step. However, since the cost function of this step is model-based, a simple matching step does not affect our major concern of the state-space search model and the design of cost functions.

4.3.3 *Experimental result*

We have performed three search strategies on the top level and two strategies on the second level, so their combinations have six different search strategies. Details about these strategies can be found in Ref. 15. Table 1(a) shows the results without validation in classification; Table 1(b) shows the results with validation in classification.

The recognition rate shown in Table 1(b) needs some explanation. In all three strategies, three test data are classified as belonging to no class. Therefore, the probability of correct classification should be higher than the values shown in the table. Some characters are missing because the feature space is sparse or the training data are incomplete. We suggest that in this case strategies with validation should be used. After a set of enough training data is gathered, then strategies without validation can be used to save computational cost.

4.4 *Extensibility*

The VISTAS system is not a complete vision system because currently only operators for simple object and character

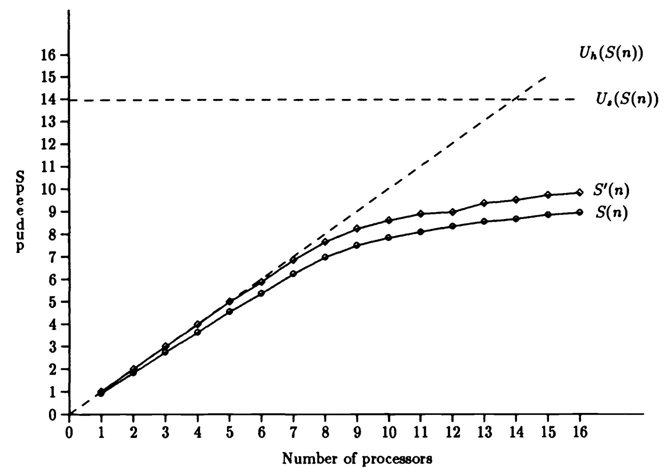


Fig. 8 Performance for parallelizing the search paths.

recognition are implemented. Also, the long-term memory of the database contains models for characters only. It is our goal that the programs for handling character recognition problems need to be changed only minimally for other applications. The achievement of this goal relies on the generality of the state-space search model as well as the design of the VISTAS system.

It has been emphasized in Sec. 2 that one reason for using the state-space search model is its generality. The model allows different problems to share the same coordination scheme and possibly many common operators. The coordination scheme is the search routine encoded in the *control* module of VISTAS; therefore, the most robust component in VISTAS is its control module.

The possibility of future extension has been taken into account in the design of all components of VISTAS. The organization does not need any change for further extension; the extension affects only the content of each component. For example, adding models to the database or adding intelligent methods to the heuristics can broaden the scope of the VISTAS system, while the functionality of each component remains the same.

5 Parallel Execution of Multiple Paths

One benefit of structuring the vision algorithm as a state-space search is that a multiplicity of paths toward goal nodes in the state space can be explored concurrently. There are essentially three ways to introduce parallelism into a sequential search problem[28]:

1. The loop can be performed by several processors concurrently; that is, they can expand more than one node at a time. This is the standard way for parallelizing a search algorithm.
2. The successors of a node can be generated by several processors concurrently.
3. Parallel searching and sorting methods can be applied to manipulate the open list.

The third method does not gain much speedup because in the domain of vision the time spent in expanding a node is much longer than that in manipulating the open list. The former is actually an image operation. The time spent in the latter can be ignored compared to any image operation.

Based on these observations, we developed a new method, the V^* algorithm, which, unlike earlier parallel search algorithms, generates the successors of a state in parallel. In machine vision, this part of the search process is very expensive, and thus V^* permits substantial speedup. An experimental evaluation of V^* has been done based on a simulation of a character recognition algorithm. The parallel programs were developed on the Sequent Balance 21000 system, under the PRESTO parallel programming environment.[29] Balance 21000 uses a UNIX-like operating system and has up to 30 physical processors. However, PRESTO limits the number of processors to 16.

There are two metrics showing the performance of the algorithm V^*, as shown in Fig. 8. One shows the "real" speedup using n processors, $S(n)$, which is computed by the following formula:

$$S(n) = \frac{\text{running time of sequential program}}{\text{running time of parallel program using } n \text{ processors}} \tag{39}$$

$$S'(n) = \frac{\text{running time of parallel program using one processor}}{\text{running time of parallel program using } n \text{ processors}} . \tag{40}$$

This measure is inappropriate because it does not tell the speedup gained by parallelizing the algorithm. However, we can learn how the parallel algorithm is scalable using the number of processors and how synchronization affects the performance by comparing two figures.

Theoretically, the speedup cannot increase without limitation. It is restricted by both the available hardware and the software structure.[30] The former is the number of processors used in parallel. Let us define $U_h[S(n)]$ to be the hardware upper bound; then we have

$$U_h[S(n)] = n . \tag{41}$$

The latter is the inherently sequential part of the program. In the context of a best-first search problem, it is the execution time from the start state to the goal state. The time spent in the solution path is inherently sequential and any search instance must execute the operators along the solution path, no matter how many processors the system has. Let $U_s[S(n)]$ be the software upper bound; we have

$$U_s[S(n)] = \frac{\text{running time of sequential search program}}{\text{running time on the solution path}} , \tag{42}$$

which is a constant, independent of n.

The upper bound of the speedup, $U[S(n)]$, of the parallel algorithm is the minimum of the hardware upper bound and the software upper bound. The details of the V^* algorithms and its experimental results can be found in Ref. 31.

6 Conclusions

The development of image processing at the individual operator level is rather mature in the sense that many optimal or near-optimal operators for specific types of images have been developed. However, the lack of knowledge about other levels of processing while designing these operators makes them far from robust. The development of an algorithm for the specific application of computer vision is more difficult.

This paper presented a promising approach to synthesize vision process based on the state-space search model. The model is able to find the best solution path from the available image operators. The appropriateness of a solution path is judged in terms of information distortion, which is a measure of difference between the input image and the interpretation made by the system. Robustness can be achieved when all possible sequences of image operators have been accommodated.

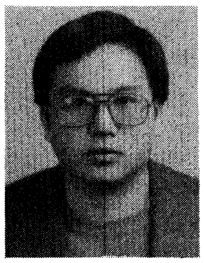
One limitation of the approach presented in this paper is that the cost of a path can be evaluated only after certain steps of operators have been executed. This is an inherently hard problem in the research of computer vision, but not a specific limitation of our model. Exact cost cannot be found at the earlier stages of computer vision unless we can really model the distributions of images at these levels.

Acknowledgment

This work is supported by National Science Council, Republic of China, under Grant NSC 80-0408-E009-30. Thanks to Dr. Tanimoto for his initial idea and his advice while the author was at the University of Washington.

References

1. D. Marr, *Vision,* Freeman, San Francisco (1982).
2. J. S. Weszka, "A survey of threshold selection techniques," *Computer Graphics and Image Processing,* **7,** 259–265 (1978).
3. D. H. Ballard and C. M. Brown, *Computer Vision,* Prentice-Hall, Englewood Cliffs, NJ (1982).
4. R. M. Haralick and L. G. Shapiro, "Segmentation and its place in machine vision," in *Proc. 6th Pfefferkorn Conference at Nicagara Falls,* pp. 39–54 (1987).
5. A. Rosenfeld, "Computer vision: basic principles," *Proc. IEEE* **76**(8), 863–868 (1988).
6. J. K. Tsotsos, "The complexity of perceptual search tasks," Technical Report RBCV-TR-89-28, Department of Computer Science, University of Toronto (1989).
7. A. Martelli, "An application of heuristic search methods to edge and contour detection," *Commun. ACM* **19**(2), 73–83 (1976).
8. C. Brice and C. Fennema, "Scene analysis using regions," *Artif. Intell.* **5**(4), 205–226 (1970).
9. J. A. Feldman and Y. Yakimovsky, "Decision theory and artificial intelligence," *Artif. Intell.* **5**(4), 349–371 (1974).
10. L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching," *IEEE Trans. Pattern Anal. Machine Intell.* **3**(5), 504–519 (1981).
11. P. C. Gaston and T. Lozano-Pérez, "Tactile recognition and localization using object models: the case of polyhedra on a plane," *IEEE Trans. Pattern Anal. Machine Intell.* **6**(3), 257–265 (1984).
12. W. E. L. Grimson and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *Int. J. Robotics Res.* **3**(3), 3–35 (1984).
13. S. Rubin, "The ARGOS image understanding system," Thesis, Department of Computer Science, Carnegie-Mellon University (1978).
14. S. L. Tanimoto, "Machine vision as state-space search," in *Machine Vision: Architectures, Algorithms, and Systems,* H. Freeman, Ed., Academic, New York (1988).
15. S. Y. Hwang, "Synthesis of vision algorithms based on state-space search," thesis, Department of Computer Science, University of Washington (1989).
16. T. D. Garvey, "Perceptual strategies for purposive vision," Technical Note 117, SRI International (1976).
17. R. C. Bolles, "Verification vision for programmable assembly," in *Proc. 5th International Joint Conference of Artificial Intelligence,* pp. 569–575 (1977).
18. R. M. Haralick, "Edge and region analysis for digital image data," *Computer Graphics and Image Processing* **12,** 60–73 (1980).
19. O. Hason and A. Meisels, "Quantifying the operations of image segmentation and understanding," Technical Report FC-TR-018 MCS-310, Department of Mathematics and Computer Science, Ben Gurion University (1988).
20. S. L. Horowitz, "The split and merge algorithm for region analysis," in *Structured Computer Vision,* S. Tanimoto and A. Klinger, Eds., Academic, New York (1980).
21. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis,* Wiley, New York (1973).
22. I. W. Burr, *Applied Statistical Methods,* Academic, New York (1974).
23. Z. Li, "Comparisons of reasoning mechanisms for computer vision," in *Proc. AAAI 87 Workshop on Uncertainty In Artificial Intelligence,* pp. 287–294 (1987).
24. J. Laird and A. Newell, "A universal weak method," Technical Report CMU-CS-83-141, Department of Computer Science, Carnegie-Mellon University (1983).
25. J. Laird, P. S. Rosenbloom, and A. Newell, "SOAR: an architecture for general problem solving," *Artif. Intell.* **33**(8), 1–64 (1987).
26. S. Y. Hwang, "Two heuristics for arranging feature-extraction operators in recursive Bayesian decision rule," *Pattern Recognition* **23**(12), 1389–1392 (1990).
27. K. R. Sloan, "The Washington image file format and dumb frame buffer," GRAIL Memo 87-1, Department of Computer Science, University of Washington (1987).
28. K. B. Irani and Y. Shih, "Parallel *A** and *AO** algorithms: an optimality criterion and performance evaluation," in *Proc. International Conference of Parallel Processing,* pp. 274–277 (1986).
29. B. N. Bershad, E. D. Lazowska, and H. M. Levy, "PRESTO: a system for objected-oriented parallel programming," Technical report TR 87-09-01, Department of Computer Science, University of Washington (1987).
30. D. L. Eager, J. Zahorjan, and E. D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. Comput.* **38**(3), 408–423 (1989).
31. S. Y. Hwang and S. L. Tanimoto, "Parallel coordination of image operators on shared-memory architecture," in *Proc. 10th International Conference on Pattern Recognition,* pp. 343–349 (1990).



Shu-Yuen Hwang is currently an associate professor in the Department of Computer Science and Information Engineering at National Chiao-Tung University. He received BS and MS degrees from National Taiwan University in 1981 and 1983, both in electrical engineering and MS and PhD degrees in computer science from the University of Washington in 1987 and 1989, respectively. His research interests include computer vision, artificial intelligence, and system simulation. Dr. Hwang is a member of AAAI, ACM, IEEE, and SPIE.