COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 36, 298–320 (1986)

# Shape from Perspective: A Rule-Based Approach

PRASANNA G. MULGAONKAR

*SRI International, Menlo Park, California 94025*

LINDA G. SHAPIRO

*Machine Vision International, Ann Arbor, Michigan 48104*

AND

ROBERT M. HARALICK

*Machine Vision International, Ann Arbor, Michigan 48104*

Received January 22, 1986; revised June 22, 1986

A system which is capable of inferring some of the structure of a 3-dimensional scene from a single perspective line drawing is discussed. Closed form equations for the inverse of the perspective transformation are used as modular inference engines. These inference engines use hypothesized spatial relationships between world entities to compute the unknown quantities such as distances between points, camera position, and focal length. The result of applying such modular computations is shown to be order independent and stable. In the absence of high-level models of objects in the scene, a hypothesize-and-test approach can be used to interpret the organization of the structures in the scene. The search space of hypotheses can be improved by utilizing known semantics of spatial relationships. © 1986 Academic Press, Inc.

## I. INTRODUCTION

One of the central aspects of the human visual system is its ability to reason about the contents and the physical structure of complex scenes. The visual system is capable of understanding the physical structure of the world even from single 2-dimensional images which may be noisy and can sometimes make reasonable estimates of surfaces and configurations of objects to which it has never been previously exposed. There are several different sources of information which the visual system seems to exploit in order to reliably analyze what we see. In order to perform visual tasks using computers, it is necessary to understand all the available information sources and develop computational tools for efficiently utilizing them.

The process of computer vision has traditionally been divided into a hierarchy of processing steps. The lower steps usually termed **low-level vision**, deal with image level operations and are concerned with such tasks as segmentation into homogeneous regions and extraction of local features such as edges or highlights. The top levels of the hierarchy are concerned with the reasoning aspects of visual processing. Such **high-level vision** tasks include naming and categorizing the observed visual



0734-189X/86 $3.00

and regions extracted by the low-level feature extractors and group them into 3-dimensional structures without making any use of 3D models or a priori object level descriptions.

In subsequent sections, we first describe the nature of the interpretation problem and the perspective equations that we can use to interpret line drawings. Next, we present the incorporation of these equations into modular processes that can cooperatively determine consistency of scene interpretations. We show that even though these modular processes operate in a distributed fashion on a shared data structure, their operation is stable and terminates. We then develop the notion of a hypothesize-and-test paradigm for utilizing the perspective equations. Such a computational system can be used in scene understanding tasks in which detailed geometric models of scene objects are not available. Although such an approach would generally be undesirable due to the computational restrictions, we show that by understanding the search space of hypotheses analyzed by the process, we can control the search path and improve the performance of the reasoning. We finally present some examples of perspective line drawings processed by an experimental system written in interpreted PROLOG with a rule base of approximately 100 equations of perspective geometry.

## II. CONSTRAINTS OF PERSPECTIVE GEOMETRY

The human visual system uses a great deal of diverse knowledge sources in concert to analyze its visual input. Information such as stereo disparity, focal adaptation, and occlusion due to motion contribute important cues which provide distance information in absolute or relative terms. In addition we have the capacity to infer 3-dimensional structure from monocular grey-tone pictures. The process of forming such images from scene structures is one of central projection. Central projection is not an information preserving mapping. Points in the real world are 3-dimensional. Points in the perspective or central projection are 2-dimensional. In fact, lines viewed on their ends will appear as points in the perspective projection. Planes viewed on their sides will appear as lines in the perspective projection. Therefore it is not possible to compute a unique 3-dimensional interpretation for any given 2-dimensional image.

During the process of image interpretation, some additional knowledge has to be inserted into the computation in order to compensate for the lost information. Researchers in the past have used various techniques to supply this missing information. The pioneering work by Roberts [19] on model-based interpretation of images made up for the missing information by supplying exact 3-dimensional models of expected objects. Later researchers concentrated on various different ways of defining the models. Brooks [2] worked with a symbolic theorem prover to constrain the free parameters of the image formation process using known models and projective invariants to guide the search. Mulgaonkar [17] utilized spatial relationships between 3-dimensional primitives of rough object models to control the computation of the free parameters. Recent work on the back projection problem by Barnard [1] has examined some conditions under which meaningful inferences can be made about 3-dimensional structures without the use of an object model. However, the inferences that can be made by back projection of individual elements in an image are very few, precisely because of the multiplicity of interpretation that each primitive element may have.

stimuli, model based vision, and construction and manipulation of data bases of expected objects based on immediate experiences and contextual cues.

The large area in between these extremes is usually termed **mid-level vision**. This area has received the least amount of formal study in the computer vision literature. What is unclear is the exact nature of the processing involved in grouping the low-level entities into more meaningful super-groups which are amenable to high-level reasoning. The exact nature of the knowledge sources which can be used to bridge the gap between syntactic image elements and semantic world level entities is not clear.

Techniques for extracting meaningful 3-dimensional descriptions from single 2-dimensional images have traditionally been termed "shape-from-" methods. Such techniques use knowledge of the physical processes which govern image formation in order to understand the structures that are visible.

The earliest shape-from- approach in the computer vision literature was the line and junction labeling technique of Guzman [8], which was refined and extended in the polyhedral world by Huffman [13] and Clowes [5], and Waltz [20]. Kanade [14] extended those ideas to the world consisting of thin planar bodies (origami world). Chakravarty [3] introduced similar concepts for objects consisting of curved surfaces. One of the most recent extensions of junction labeling also incorporated the effects of perspective into the projection geometry (Lee, Haralick, and Zhang [15]). All these techniques use world information which most generally characterizes the domain under consideration. For example, the early works on junction labeling restricted the domain to polyhedral objects with up to trihedral vertices. The 3-dimensional structures computed by these labeling procedures consisted of relative orientations between surfaces and detection of concavities and convexities in the 3-dimensional world.

Shape-from-shading techniques pioneered by Horn [11, 12] and with recent contributions by Pentland [18], make minimal assumptions about the characteristics of the scene: surfaces are assumed lambertian, and some minimal smoothness criterion is enforced for computational tractability, and simple point and diffuse lighting models are used. The main source of knowledge utilized for computing the surface shapes consists of models of the image formation method.

Shape-from-texture techniques (Witkin [21], and Davis, Janos, and Dunn [6]) perform similar reasoning based on the transformations that spatial frequencies (texture elements) undergo as a result of image formation.

Shape-from-skewed symmetry techniques (Kanade [14]) are based on the assumption of symmetric 3-dimensional shapes and the equations governing their apparent skew induced by the projection process.

In this paper we present a technique for utilizing a strong knowledge source that provides very tight constraints on ways in which 3-dimensional entities can be arranged in the world in order to give rise to the observed image. The knowledge consists of the rules that govern the perspective projection process. We show that it is possible to use the equations of perspective geometry, organized as modular inference engines, to cooperatively process the image level primitives and interpret their configurations in terms of a plausible arrangement of lines, points, arcs, and planes in 3-space. This approach is philosophically an extension of the shape-from-skewed symmetry approach in that it considers the apparent projective distortions of arbitrary assemblages of image-level primitives. It would operate on points, lines,

We show in this paper, that we can utilize equations of perspective geometry in a cooperative sense to rule out a large number of the multiple interpretations and arrive at a plausible structure or structures that could give rise to the image. Haralick [10] has an excellent compilation of relevant equations for the perspective projection of points, lines, and planes. Mulgaonkar [16] contains additional equations dealing with the perspective projection of conic sections in a common framework.

The domain from which we draw the examples in this paper, consists of solid objects made up of planar and cylindrical faces. We are given an image consisting of edges and arcs corresponding to edges between surfaces in the scene. We do not assume any a priori knowledge about the objects other than the class of surfaces defined above and possible spatial relationships that can be used to define their arrangement in space. For example, we know that straight lines can be parallel to each other, they can lie in a plane, they can be perpendicular to planes, etc. We show how we can use this knowledge along with the mathematics which transforms the generic class of 3-dimensional primitives into their corresponding images to infer the 3-dimensional structure given a single 2-dimensional image. We assume a non-singular or general viewpoint which means that straight lines in the image are projections of straight line segments in the world and no lines in 3-space project onto points in the image.

As an example, consider the type of reasoning that would be involved in interpreting the line drawing shown in Fig. 1. In this example as in subsequent ones, we follow the convention that image primitives are labeled using lower case letters, and their corresponding 3-dimensional counterparts are labeled with the equivalent upper case letters. The image consists of nine visible straight line segments, which when taken four at a time, bound three surfaces. We do not know how these surfaces are arranged in space. However, we do know that any possible arrangement must be such that from some camera position, it produces the observed arrangement of lines and regions in the image. Not all possible arrangements of these lines would satisfy this condition. For example, if lines $A$ and $B$ (corresponding to line segments $a$ and $b$) were parallel and $C$ and $D$ (correspondng to line segments $c$ and $d$) were parallel, then the lengths of the line segments $C$ and $D$ must be equal. In addition, the plane $ABCD$ corresponding to the region $abcd$ would have to be horizontal because the intersection of the images of the parallel pairs lie on the horizontal line through the center of the image.

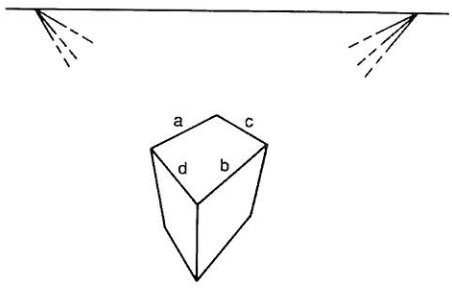


FIG. 1. A perspective view of a simple cube.

The key feature of the reasoning is that we use measured relationships between the lower case entities to determine the possible relationships between the corresponding upper case entities. We define an **interpretation** of an image to be a set of inferred relationships between the 3-dimensional entities which when transformed by the rules of perspective geometry, agree with the measurements made in the image. Of all possible interpretations, we look for those that are in some sense *maximal* or *best*. In this paper, we define an **optimal interpretation** to be that interpretation (or interpretations) that constrains the most number of 3-dimensional entities.

The search may be performed by hypothesizing possible relationships between groups of 3D entities and verifying their correspondence with measurements by applying the perspective equations. In reality, this problem is a lot more complicated. Spatial relationships between 3-dimensional entities have numerical valued attributes whose values are drawn from the infinite domain of real numbers. For example, parallel lines are attributed by the normal distance between them. Surfaces have the direction cosines of their normals as attributes. The values of these attributes control the appearance of the entities in the projection. Therefore, a hypothesis cannot be verified until all relevant attributes of the spatial relationships have numerical values. The domain of the attribute values is infinite and therefore we must compute them rather than search for them.

The equations of perspective geometry can be inverted and values for the attributes computed, based on the spatial relationships present in the hypothesis. For example, if a pair of lines in the image is hypothesized parallel in 3-space, then the direction cosines of the lines can be computed based on the measured location of their vanishing point in the image. Perspective geometry provides a large repertoire of such inferences and, therefore, a large number of attributes relate to more than one image-level measurement. Thus there are often multiple computation paths by which we can compute the values of most attributes. The definition of consistency in this framework then becomes

*A hypothesis consisting of proposed spatial relationships between 3-dimensional entities is* **inconsistent** *if the measurements from the image imply that some attribute of some relational tuple in the hypothesis, simultaneously must have more than one distinct value. A hypothesis that is not inconsistent is* **consistent**.

The vision system described in this paper uses this definition of consistency to find the largest consistent hypothesis relating the 3-dimensional entities corresponding to the entities visible in a given perspective image.

In the next section, we define the notion of an **inference engine** used for applying a single rule of perspective geometry to a hypothesis. We develop the concept of a string of inference steps consisting of sequential applications of inference engines to compute numerical values for attributes. We then prove the stability of such a distributed computation scheme for determining the consistency of a given hypothesis.

## III. INFERENCE ENGINES

**Inference engines** are modular computation units, which accept as input a specific set of attributed relational tuples made up of possible relationships between world entities, and a set of measurements taken from the image. Based on the measure-

ments and on the previously computed values for the attributes of the relational tuples, they compute values for other attributes of tuples in the input set.

The mode of operation of these inference engines is as follows: The initial input consists of a hypothesis whose validity is to be determined. All the attributes of all the relational tuples in the hypothesis are initially valueless. Inference engines are triggered based on their input requirements and compute values for some attributes. For example, if the hypothesis contains a relational tuple of the form: (*parallel lineA lineB*), the vanishing point inference engine would be triggered since all parallel lines have the same vanishing point, and would compute a value for the vanishing point attribute of **line*A*** and **line*B***. The processing involved in this case is to compute the intersection of the corresponding image lines **line*a*** and **line*b***. Subsequent inference engines whose computations use vanishing points may then be triggered. For example, one inference engine may compute the focal length of the camera based on vanishing points for 2 non-parallel, planar pairs of lines. This inference engine would look for the following relational tuples in the hypothesis

(*3D-line lineA *a *b *c *d*)

(*3D-line lineB *a *b *c *d*)

(*parallel lineA lineB *vpx *vpz*)

(*parallel lineC lineD *vpx *vpz*)

(*in-a-plane lineA lineC *normala *normalb *normalx *offsetd*)

(*not-parallel lineA lineC *angle*).

The terms with asterisks are place holders for the attributes of the relational tuples. For example, the attributes of the *3D-line* relational tuples define the equation of the line in the form

$$*ax + *by + *cz + *d = 0;$$

the attributes of the *parallel* relational tuples encode the vanishing point in screen coordinates for the pair of lines participating in the relation; and the attributes of a plane define the plane equation. The inference engine would examine the vanishing point attributes of the two **parallel** tuples to see if they had values assigned to them. To summarize, inference engines are independent computational modules which compute values for attributes of an input set of relational tuples, based on measurements from the image and possibly some previously computed attributes of some tuples in the input. A hypothesis is inconsistent if applications of these engines lead to distinctly different values for any attribute.

The questions that arise in this context deal with the stability of termination of the computations. Suppose a set of inference engine applications determine a hypothesis to be valid. Is it possible to apply a different set of engines or to change the order of application and arrive at the conclusion that the same hypothesis is inconsistent? Is it possible that changing the order of engine applications changes the final set of values for the attributes in a hypothesis and, if so, does the application process involve a search for the *correct* sequence of applications.

These questions arise in every **blackboard**-type distributed computation system where a group of modules can independently update and change information in a common-data store. Independent modules can be shown to be correct. However, since the inference engines interact with each other and use the results of each

other's calculations, the question of order dependence and uniqueness of result must be proved.

In predicate calculus, consistency of a set of predicates may be viewed as a conjunction of conditions that the set of predicates must jointly satisfy. Since conjunction is a commutative operation, the order in which the terms of the predicate are tested is irrelevant. Intuitively it may seem that similar results should hold for applications of inference engines because they, too, check consistency with respect to individual rules of perspective geometry. However, inference engines cannot be applied in any arbitrary order. An engine can *only* be applied if the information it requires to execute is already present. That is, if an inference engine requires that some attributes of some tuples in the input hypothesis set have previously computed values, it cannot operate until some other module computes the required values. Thus, there is a partial ordering which describes all the legal sequences in which the engines may be applied. Other complications result from the fact that inference engines compute values for attributes in addition to providing a *consistent/inconsistent* response. These values cause the state of the hypothesis to incrementally change. Therefore, some thought is required to show that intuitive result does indeed hold and that inference engine application is a stable process whose result does not depend on the particular sequence chosen.

By carefully formalizing the concept of an inference engine, we can prove a series of interesting theorems which show the desired properties of the inferencing process. The proofs of the theorems are presented in the Appendix, using the terminology defined in this section.

We start with some definitions of the terms involved. We say that an inference engine $E$ is **applicable** to a subset $K$ of the input hypothesis $H$ if the following conditions are met:

(1) $K$ satisfies the input requirements of $E$. That is, $K$ contains the relational tuples that $E$ uses as the basis of its computation and the required attributes of tuples in $K$ have previously been assigned values.

(2) No proper subset of $K$ satisfies the input requirements of $E$.

An **application** of $E$ to $K$ is denoted $E(K)$, where $E$ is applicable to $K$. An application may succeed or fail. An application fails if the value computed by $E$ for some attribute of $K$ is inconsistent with a previously computed value for the same attribute. If the application succeeds, the set $K$ is changed to a new set $K'$ which differs from the original in that at most one attribute of $K$ which was previously undefined, now has a numeric value associated with it.

An **inference step** consists of one application of $E$ to a subset $K$ of the hypothesis and the replacement of the subset $K$ by the new subset $K'$ in the set of input hypotheses.

A **sequence of inferences** on a hypothesis $H$ is a sequence $E_j(E_{j-1} \cdots (H) \cdots)$ of inference steps where each engine is applied to the output of the previous step. We stipulate that no inference engine may apply more than once to the same subset $K$ of $H$.

A sequence of inferences $E_j(E_{j-1} \cdots (H) \cdots)$ is called **terminating** if either $E_j(\cdot)$ is a failed inference step, or there does not exist any other inference engine $E_{j+1}$ which is applicable to the result of the sequence.

THEOREM 1. *Consider a sequence of applications of the inference engines to a hypothesis H. If at the ith step, engine $E_i$ is applicable, then $E_i$ will remain applicable at all inference steps $j > i$. That is, we can defer the application of an applicable inference engine.*

THEOREM 2. *If at step i, application of inference engine $E_i$ would fail, then $E_i$ would fail even if its application is deferred.*

THEOREM 3. *Changing the order of application of inference engines does not cause a successful sequence to terminate in failure.*

THEOREM 4. *Any sequence is either a terminating sequence or can be extended by further applications of inference engines to form a terminating sequence.*

THEOREM 5. *If any one sequence of applications of a set of inference engines to a hypothesis terminates in failure, then all possible sequences of applications terminate in failure.*

THEOREM 6. *If any one sequence of applications terminates successfully, then all possible sequences terminate successfully.*

THEOREM 7. *Ignoring permutations, there is at most one successful sequence of applications of a set of inference engines to a hypothesis.*

We use the concept of terminating sequences to define consistency of hypotheses as follows: A given hypothesis is inconsistent with respect to the knowledge encoded in a given set of inference engines and the measurements from a given image if it has an associated sequence of applications which terminate in failure. If on the other hand, the associated sequence of applications (which may be of zero length) ends successfully, then the hypothesis is consistent.

As examples, we list a few of the inference engines that are implemented in the experimental system. The system currently has 100 inference engines. Any vision system that goes all the way from low-level image operations to a final interpretation of the scene would require a much greater variety of knowledge sources. The system reported here has the specific task of verifying our hypothesis that perspective geometry provides a strong enough set of constraints to produce reasonable hypotheses about scene structure. In addition, these constraints can be implemented to use closed form inverse projection equations instead of a theorem prover search over the infinite real number set.

*Inference Engine 1.* Given the hypothesis that two lines are parallel and whose images are not parallel in the image, determine the vanishing point of the lines as the intersection of their images.

*Inference Engine 2.* Given the hypothesis that two lines are coplanar, and they do not have a common vanishing point, compute the vanishing trace of their common plane.

*Inference Engine 3.* Given a plane with a known vanishing trace, compute the vanishing points of all lines hypothesized as lying in that plane.

Note the Engines 1 and 3 both compute values for the same attribute—the vanishing point of a line. This is the basis of consistency checking. The method

relies on the fact that perspective geometry has a large number of such mutually constraining equations.

In summary, we have shown that the problem of checking the consistency of hypotheses against the constraints of perspective geometry can be solved efficiently. Thus, such a process can be usefully incorporated as one component of an image interpretation system, on an equal footing with other shape-from- processes. It could be used to verify the consistency of hypotheses generated using, say, shape-from-shading or shape-from-symmetry. In addition, by incorporating processes for hypotheses generation, it can be used to generate all consistent interpretations for the structures in the scene.

## IV. CHARACTERIZATION OF INFERENCE ENGINES

In the previous section, we have defined the notion of independent inference engines which can be used to determine the consistency of a hypothesis. We showed that the process of applying the engines is a sequential process, with no search involved. Because of this, the overall problem of finding an optimal interpretation becomes an *NP*-complete problem.

In essence, when the problem is framed in this way, it is reduced from a problem of theorem proving in predicate calculus to the problem of verifying the consistency of a propositional logic statement. In this section, we characterize the nature of this transformation to understand the exact difference.

### *A. Predicate Calculus with Restrictions*

We will use a small example to illustrate what happens when closed-form expressions are used to encode the interrelationships between numeric attributes of the tuples which constitute a hypothesis. Consider the following problem: Suppose we make the hypothesis that a point in 3-dimensional space corresponds to a particular point on the image with known coordinates $x_s$, $z_s$. Further, we know the $x$ coordinate of the 3-dimensional point, and the focal length $f$ of the imaging system. Using boldface symbols ($\mathbf{x}$) for known values, underlined symbols ($\underline{x}$) as place holders for the unknowns, and italic symbols ($x$) for function names, the problem of determining if the hypothesis is consistent could be phrased in theorem proving as the problem of determining the proof for a theorem,

$$\exists \underline{y} \exists \underline{z} \mathit{Proj}\left[\mathbf{x}, \underline{y}, \underline{z}, \mathbf{f}, \mathbf{x_s}, \mathbf{z_s}\right]$$

where the predicate *Proj*: $\mathscr{R}^6 \rightarrow \{\text{True}, \text{False}\}$ is true if the point defined by the 3-dimensional space coordinates given by the first three arguments corresponds to the screen coordinates given by the last two arguments, when the camera focal length is the fourth argument. The axioms that the theorem prover would use would consist of all axioms of arithmetic and the fact that the projection of a point $(x, y, z)$ is $(xf/y, zf/y)$, where $f$ is the lens focal length and $y$ is measured along the optic axis. We will assume that there is some suitable representation for these axioms which permits the predicate *Proj* to do what is necessary.

If the hypothesis is consistent, there would be some values for the unknown variables $y$ and $z$ which would satisfy the theorem. If no such values could be determined, the theorem would be false and the hypothesis would be inconsistent. Consider how a theorem prover would try to prove such a predicate calculus statement. It would essentially expand the semantic tree [4] and attempt to find a

node in the tree that would disprove the negation of the theorem. This is precisely where the problem arises. The semantic tree corresponding to the theorem above is infinite because of the domain from which the variables $\underline{y}$ and $\underline{z}$ are drawn is the infinite domain $\mathscr{R}$ of reals.

Consider what happens to the same theorem when it is recast into a framework in which the inference engines compute, in closed form, the inverse of the projection. The proof of the first-order logic theorem above becomes equivalent to determining the consistency of a predicate

$$Proj\left[\mathbf{x}, E_y[\mathbf{x}, \mathbf{f}, \mathbf{x}_s, \mathbf{z}_s], E_z[\mathbf{x}, \mathbf{f}, \mathbf{x}_s, \mathbf{z}_s], \mathbf{f}, \mathbf{x}_s, \mathbf{z}_s\right]$$

where the functions $E_y$ and $E_z$ are mappings from $\mathscr{R}^4 \rightarrow \mathscr{R}$ which determine the $y$ and $z$ coordinates of a point. The mathematics for computing the space coordinates $y$ and $z$ of a point given its projection $(x_s, z_s)$, the value for its $x$ coordinate, and the focal length $f$ can be found in Haralick [10].

Now there is no search involved. The search that the theorem prover had to do has been *compiled* into the functions $E_y$ and $E_z$. By proper construction of these functions, it is possible to assert that the predicate in the second form is true if and only if the theorem in the first form is satisfiable.

It is precisely these functions that are encoded in the action part of the inference engines. The theorem proving part is abstracted and precompiled into the predicates, reducing the complexity of the problem to a simpler domain. The search involved in systematically generating possible hypotheses then works in a domain equivalent to propositional logic, thus reducing the overall problem complexity.

### *B. Inference Engines as a Reasoning Path*

In addition to the characterization of inference engines described above, the inference engines describe a path that the reasoning process follows in determining the consistency or inconsistency of a given hypothesis. An inference engine $E$: $H \rightarrow H'$ is a function which maps a hypothesis into one which may have values for some attributes whose values were unknown prior to the application. $H$ is a subset of the space of all attributed tuples that can be constructed from the set of image primitives and relationships that we can reason about. Each inference engine has a specific input tuple set that it works on, and its result is the computation of a value for some attribute of the input set. We implicitly state that an inference engine works on a particular input set only once. That is, once the engine has "seen" a particular subset of the hypothesis, it will not apply to the same subset again. Once all applicable inference engines have worked on all their possible input sets in a given hypothesis, we define it to be a terminating sequence of applications. An inconsistent hypothesis is one for which a terminating application sequence assigns contradictory values for some attribute.

Let the hypothesis have $m$ attributes associated with the tuples. Therefore at the end of any terminating application sequence, at most $m$ attributes could have values associated with them. Let $n$ of these attributes ($n \leq m$) be the ones which are computed by the inference engines. Ignoring the actual values themselves, there are $2^n$ combinations in which values are assigned to attributes. Initially the hypothesis starts off with all attributes "unknown." Let the state of the hypothesis be defined

by the pattern of which attribute values are known and which are still unknown. These states can be considered to be nodes in a network, with the inference engines defining the arcs between the nodes. If a particular inference engine can cause a transition from one state to another, there is a directed arc between the two states. The process of inference engine applications results in sequential state transitions which start with the state corresponding to all attribute values unknown and end with the state in which $n$ or the $m$ attributes have values. Let the starting node be labeled $N_0$, and the final node be labeled $N_n$. Let each arc be labeled by the name of the inference engine which can cause transition between the nodes it connects. This network has several interesting properties:

- The digraph of the network has no cycles of length greater than one. This follows from the fact that edges in the graph are constrained by the nature of the inference engines which define the arc. If the hypothesis is in a state in which $k$ of the attributes have values, application of any engine will not reduce the number of attributes with values.
- The maximum length of any path from the start node to the final node is exactly $n$.
- The graph can have cycles of length one. Such arcs correspond to applications of an engine which recomputes the values for some attribute. In fact, if there is an arc corresponding to engine $E_i$ from node $N_i$ to node $N_j$, then there is a loop also labeled $E_i$ from node $N_j$ to itself.
- Conversely, if there is a loop from node $N_j$ to itself labeled $E_i$, then there must exist an arc from some other node $N_i \neq N_j$ to $N_j$ also labeled $E_i$. Since we are looking for paths from the start node, we can eliminate the loops from the network.
- In addition to the start node, there may be other nodes with an in-degree of zero. For example, it is not possible to compute the 3-dimensional coordinates of a point based on the image information alone. Thus, the node which corresponds to known values for coordinate attributes cannot have any arcs leading into it. However, if some coordinate information is supplied a priori, then it can be used to compute the locations of other points in the world. Thus, the corresponding node can have arcs leading out.
- The graph of the network formed by deleting all loops and removing nodes with in-degree zero, except the start node $N_0$, is a simple connected digraph.

This network characterizes the paths taken by the engine application process through the space of value assignments to the attributes. The point to note here is that the graph defines a very small and structured subspace of all possible paths which could be defined in an arbitrary network on $m$ nodes. This structure can be contrasted with what a theorem proving machine would have to contend with if the problem were phrased as a first-order predicate logic problem.

## V. SEARCHING FOR THE MAXIMAL CONSISTENT HYPOTHESIS

In the previous sections, we discussed a technique for encoding the knowledge of perspective geometry and applying that knowledge to determine the consistency of a hypothesis. We showed that the problem of determining consistency of a hypothesis is a linear complexity task which can be performed efficiently. Thus, the practicality

of using shape-from-perspective processes to infer scene structure is contingent on the efficiency of algorithms for generating hypotheses to be tested.

The efficiency of any processing depends on the amount of world knowledge that is brought to bear. Model-based vision systems which have geometric or structural models of the 3-dimensional world have an advantage over non-model-based systems because the models constrain the space of interpretations. Vision-based object identification systems can test the more likely hypotheses before the unlikely ones. For example, to identify objects in an office scene, the hypothesis generator could check for "telephone" before "refrigerator." However, in such cases, it is hard to separate the contribution of the hypothesis generation process from that of the hypothesis testing process towards the overall interpretive power of the system. For example, if some hypothesis was not produced in the list of "consistent interpretations" output by the system, it may not be clear if the knowledge embodied in the consistency checking was powerful enough to reject the hypothesis or if the hypothesis was never generated.

To study the interpretive power of perspective geometry, we restrict our attention in this paper to a hypothesis generation system that does not use object models. Another motivation for studying such a system is that there may be cases where object-level knowledge is simply not available. Such bottom-up hypothesis generation techniques could also be used to augment and extend partial top-down hypotheses which may be model driven. We show that although the search space of hypotheses is extremely large, it has a structure which can be utilized in order to quickly converge to the desired point.

The space over which the search for the *best* hypotheses is performed is the set of all possible subsets of relational tuples that can be constructed from the set of 3-dimensional entities. Recall that although the consistency of hypotheses is determined by numeric values assigned to the attributes of the tuples, the search is not performed over the infinite set of real numbers. Instead, by using the inverse perspective equations encoded as closed-form inference engines, we *compute* the numeric values based on the finite set of relational tuples. This reduces the overall complexity of the search to an *NP*-complete problem. However, a blind search through this exponential space is still prohibitively expensive.

The key structure of the space comes from two sources. The first deals with the nature of the consistency determination. If a hypothesis is consistent as defined by the terminating computations of a group of inference engines, all subsets of the hypothesis are also consistent. We cannot force inconsistency into a set of relational tuples by *removing* any elements from it. In addition, if a hypothesis is inconsistent, it cannot be made consistent by adding new tuples. As long as the inconsistent core remains, the new superset of tuples would still result in a failed terminating sequence of inferences.

This structure can be utilized during the search process by ensuring that the search algorithm never considers parts of the space that cannot contain the desired solution. Consider the search space organized as a binary tree. Suppose there are $n$ possible relational tuples. The tree would then have $n + 1$ levels labeled 0 through $n$. The $2^n$ leaf nodes correspond to the possible hypotheses. At each internal node at level $i$, the left subtree corresponds to the hypotheses in which tuple $i$ is present and the right subtree corresponds to the hypotheses without tuple $i$. Figure 2 shows the binary tree corresponding to a three tuple search space. It can be seen that the

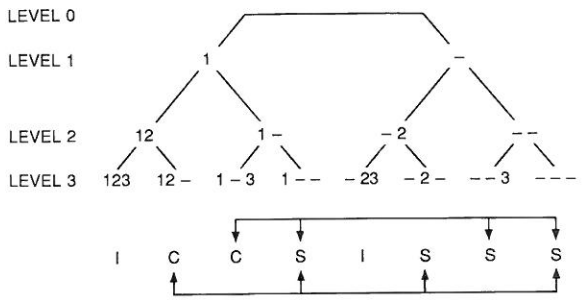


FIG. 2. Tree representation for a sample search space of 3 relational tuples with 123 and 23 being inconsistent ($I$), 12 and 13 being maximally consistent ($C$), and 1, 2, 3, and NUL being the subsets ($S$) of the maximally consistent subsets. Also note how the subsets of the maximally consistent hypotheses do not follow any regular pattern.

subsets of the consistent nodes and the supersets of the inconsistent nodes are not organized in any simply denotable or regular fashion. Thus it is not easy to automatically disregard all subsets once a consistent solution is found.

However, it can be proved that if the *leftmost* leaf node in any subtree rooted at an internal node is a subset of some previously generated solution, then *all* nodes in that subtree are subsets of the same solution. Therefore the entire subtree can be pruned and ignored by the search algorithm. Similarly if the *rightmost* leaf node is not a subset of any previous solution, then none of the possible solutions in the tree are subsets. This tree pruning can be augmented by forward checking [9] applied at internal nodes in the tree.

The second source of knowledge which structures the search space is the knowledge of the semantics of the spatial relations from which the hypotheses are drawn. For example, if one of the spatial relations is the **parallel** relation between pairs of lines, then it is meaningless to construct tuples of the form **(parallel PlaneA PlaneB)**[1], when **PlaneA** and **PlaneB** are planes. Second, the relations themselves may be symmetric, reflexive, or transitive. In such cases, hypotheses which do not satisfy these conditions are automatically **inconsistent**. For example, if a hypothesis consists of the following two tuples: {(parallel line$A$ line$B$) (parallel line$B$ line$C$)} but does not contain the tuple (parallel line$A$ line$C$) then it can be declared inconsistent by virtue of being incomplete. This form of transitivity can be extended to include interrelation relationships. For example, if three lines $A$, $B$, and $C$, lie in a common plane with line $B$ perpendicular to both lines $A$ and $C$, then lines $A$ and $C$ must be parallel. In fact, such interrelationships between the various spatial relationships can be used to automatically extend partial hypotheses of the form {(parallel line$A$ line$B$) (parallel line$B$ line$C$)} to include the implied relational tuple (parallel line$A$ line$C$).

Such semantic consistency rules can be used to efficiently reduce the search space by forcing logical completeness at all internal nodes of the tree. For example, if at internal node $i$ in the tree, we take the left branch, (include tuple $i$ in the partial hypothesis), we also include all the tuples implied by the selections at levels 0

[1]Note: For clarity, the place holders for the attributes have been omitted in this and subsequent examples.

through $i-1$ and the new tuple $i$. This allows us to make decisions about lower levels in the tree well before we reach them, reducing the number of levels that we actually have to search. For example, if we have already included a relationship (*parallel lineA lineB*) in the partial hypothesis and then add the tuple (*parallel lineB lineC*), we can automatically include the logically implied tuple (*parallel lineA lineC*).

## VI. EXPERIMENTAL RESULTS

The reasoning system described in the previous sections has been implemented using a rule base of approximately 100 inference engines. It can reason about 3-dimensional points, lines, circular arcs, and planes. It is supplied with a digitized line drawing of a perspective drawing of 3-dimensional solids taken from an unknown camera position with a camera of unknown focal length. The input can be augmented if necessary by supplying any of the unknowns such as position of some point in space or some of the camera parameters as a priori knowledge.

The final output of the system is a listing of the **best** hypothesis along with the computed values for the numeric attributes such as coordinates of points, direction cosines of lines, normals to planes, etc. Absolute coordinates of points and lengths of lines cannot be computed without some reference point whose location is known because perspective projection involves an unknown scale change. However, lengths and distances relative to the focal length can be computed.

The system was tested on a variety of digitized line drawings as reported by Mulgaonkar [16]. The drawings were taken from a book of perspective etchings by deVries [7] containing a large number of drawings of architectural, indoor, and

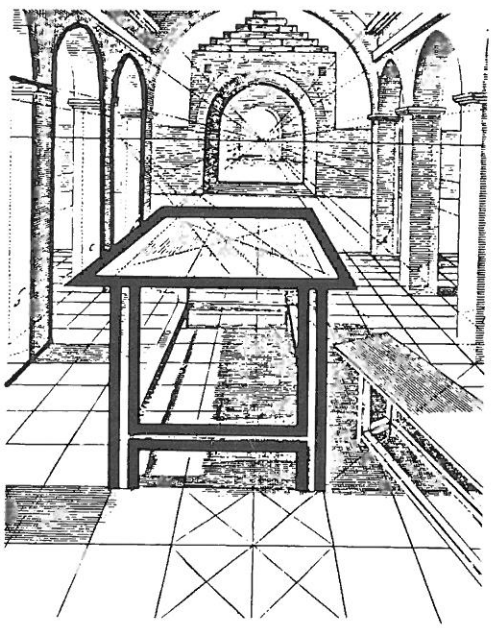

FIG. 3. Image of a table from deVries [7]. Lines digitized as input to the reasoning system are shown bold.

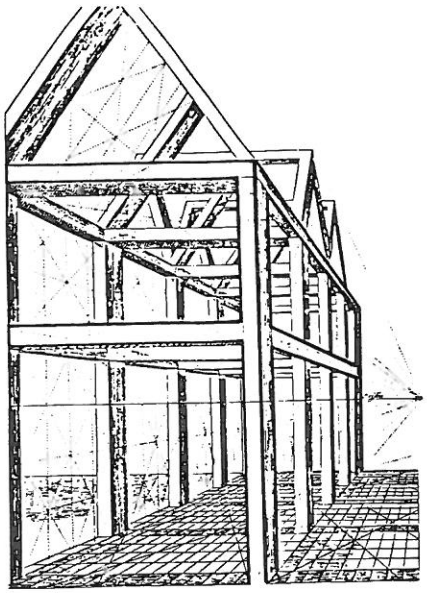

FIG. 4. Image of a building from DeVries [7].

outdoor scenes. The drawings consisted of lines and circular arcs shown in perspective. In addition, several projections were constructed of synthetic 3-dimensional objects by a graphics system. Figures 3 and 4 are two of the digitized figures from DeVries [7]. Figures 5 and 6 show the result of the processing shown as a perspective view of the structure hypothesized by the system taken from a different viewpoint. This was done by feeding the computed point coordinates and line directions to a graphics program and generating an image of the result.

We illustrate the reasoning path followed by the inference engine applications by listing the analysis of a simple image—a perspective projection of a rectangle. Figure 7a shows the geometry of the projection, and the projected image itself, is shown in Fig. 7b. *ABCD* is a rectangle in 3D, which projects onto the quadrilateral *abcd*. The focal length $f$ of the camera is not known.

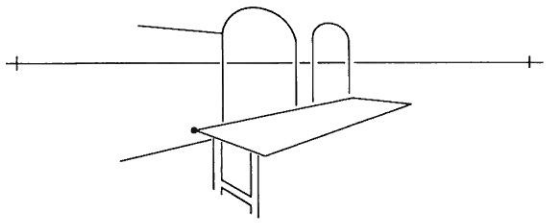

FIG. 5. Perspective image of the structure inferred by the reasoning program for the input shown in Fig. 3. Coordinates of the point marked were input as external knowledge for the absolute locational inferences.

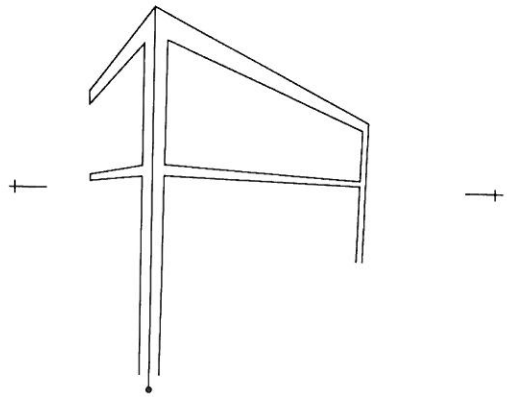

FIG. 6. Results of processing Fig. 4.

The input to the reasoning system consists of assertions specifying the end points of the lines $a$, $b$, $c$, and $d$, and the 2D connectivity relationships between the lines. A segment of the input is shown below. Annotations are enclosed in square brackets:

```
(assert

    ((2D-line a (1 2) (0.0 1.0 1.82)))
    [line between points 1 and 2 with screen equation y = 1.82]

    ((3D-line A (unknown unknown unknown unknown)))
    [corresponding 3-D line, with unknown 3-D equation]
            .
            .
            .
            .
    ((2D-point 1 (0.05 −1.82)))
    ((3D-point 1 (unknown unknown unknown)))
            .
            .
            .
            .
    ((line-intersection (a b) (−0.36 −1.82)))
            .
            .
            .
            .
```

The goal of the inference engines is to hypothesize plausible camera centered 3D relationships between the lines $A$, $B$, $C$, and $D$ in the world, which would account for the observed screen image *abcd*. For illustrative purposes, we predefine coplanarity of the lines $A$, $B$, $C$, and $D$ in space by appending the following

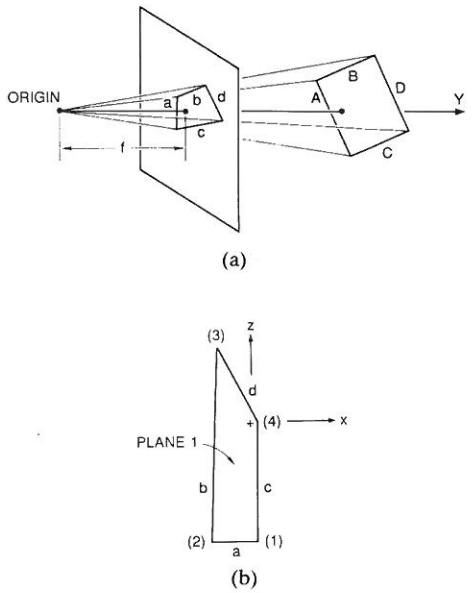


FIG. 7. Perspective projection of a rectangle.

assertions to the input shown above:

(assert
((in-a-plane *A B* (unknown unknown unknown unknown)))
[lines *A* and *B* are coplanar with unknown plane parameters]

((in-a-plane *B C* (unknown unknown unknown unknown)))
.
.
.
.

Thus, the choice of 3D relationships that may hold between the lines are parallel, perpendicular, and skewed. The unknown numeric attributes of the resulting relational tuples are the locations, and orientations of the 3D points, lines, and planes, and described in Section III.

An annotated trace of the search for the maximal consistent hypothesis for this example is shown below. The four lines can participate in 12 relational tuples. Consequently, the search space for the maximal hypothesis is $2^{12} = 4096$ possible sets of tuples. There are a total of 28 numeric attributes that qualify the 3D relations ($3 \times 4$ line orientation attributes, $3 \times 4$ point locations, 3 surface normal attributes, and the unknown focal length). In the trace, hypothesized tuples are shown in parentheses, annotations are in square brackets, and computation of attribute values indicated in angular brackets.

The search proceeds by hypothesizing potential relational tuples, which are then checked for consistency with existing tuples, and discarded if found inappropriate. Attribute values are calculated when enough tuples have been instantiated to permit

closed-form computation of the appropriate values:

| | |
|---|---|
| (parallel $A$ $B$) | [First hypothesis]<br>[Rejected because $a$ and $b$ touch<br>[ implying infinite lengths for $A$ and $B$] |
| (perpendicular $A$ $B$) | [Not inconsistent with anything] |
| (parallel $A$ $C$) | [Rejected, same reason as (parallel $A$ $B$)] |
| (perpendicular $A$ $C$) | [Not inconsistent] |
| (parallel $B$ $C$) | [Hypothesized at this level because<br>[ it is logically implied by $B$ and $C$ both<br>[ being perpendicular to $A$, coplanar with $A$<br>[ and not colinear] |
| ⟨compute vanishing point for lines $B$ $C$⟩ | [Since they are hypothesized parallel] |
| (parallel $A$ $D$) | [Back to the regular search sequence] |
| ⟨compute vanishing point for lines $A$ $D$⟩ | [Since they are hypothesized parallel] |
| (perpendicular $B$ $D$) | [Logical implication of (perpendicular $A$ $B$)<br>[ and parallel $A$ $D$] |
| (perpendicular $C$ $D$) | |
| ⟨compute focal length⟩ | [Given pair of conjugate vanishing points<br>[ from perpendicular lines $A$ and $B$] |
| ⟨compute direction cosines of the lines $A$, $B$, $C$, $D$⟩ | [Given focal length and vanishing points] |
| ⟨compute normal to the surface defined by $A$, $B$, $C$, $D$⟩ | [As the cross product of the direction<br>[ cosines of the lines] |
| (perpendicular $A$ $D$) | [Inconsistent with prior tuple<br>[ parallel $A$ $D$. These tuples never get<br>[ instantiated] |
| (perpendicular $B$ $C$) | [Inconsistent.] |
| (parallel $B$ $D$) | [Inconsistent.] |
| (parallel $C$ $D$) | [Inconsistent.] |

[Thus the search is now at a leaf node in the tree, corresponding
[ the maximal consistent hypothesis. The hypothesis at this stage
[ consists of the tuples:

(parallel $A$ $D$)
(parallel $B$ $C$)
(perpendicular $A$ $B$)

(perpendicular *A C*)
(perpendicular *B D*)
(perpendicular *C D*)

[ The computed attribute values are as follows:
(focal length 5.0)
(3D-Line *A* direction cosines 0.17 0.92 −0.33)
(3D-Line *B* direction cosines 0.00 0.34 0.94)
(Normal to the plane 0.17 0.93 −0.34)

[ The constant in the 3D line equation and the offset of the plane cannot be computed since there is no absolute distance information available. If a 3-D object is scaled up in size and moved further out from the camera, its projection stays the same. If absolute coordinate information is available for any one point in the world, all the remaining points and distances can be calculated. ]

The CPU time required to reach this solution was 67.7 s on a time shared VAX 11/780, using an extended PROLOG interpreter written in RATFOR. We are planning to re-implement the entire system in a procedural language, such as C, which should cut down the execution time by at least one order of magnitude.

## VII. CONCLUSIONS

In this paper we have developed and described a system which generated partial 3-dimensional interpretations from 2-dimensional primitives extracted from a single perspective view of an unknown scene. Three main points were addressed in this paper. They are:

- The use of closed form inverse perspective equations as knowledge sources for analyzing images was examined. Expression of perspective geometry knowledge in this form allows meaningful hypotheses to be made about scene structure without the usual extended theorem prover type search.

- The question of proving termination and stability of a distributed processing scheme where each independent module modifies a common global data base was addressed. Such computational schemes are commonly referred to as using a **blackboard** data sharing approach. When the order of application is not defined, we must show that the results do not depend on it. For our system we prove that the result is independent of the order of application and is unique.

- It was shown that studying the search space over which a reasoning system works is a fruitful approach to reducing the overall size of the space.

A reasoning system based on the ideas presented in this paper would form one part of a complete vision system. It could be called upon to verify partial structures constructed using other sources of knowledge such as shape-from-shading or analysis of texture gradients. If the input structures were consistent with the geometry of perspective projection, a partial tree search could be performed to extend the interpretation by adding new relational tuples connecting other image primitives to those in the input set. If an inconsistency were detected, the lower level processing routines such as segmentation or line extraction could be invoked in the indicated problem areas with different parameters in an attempt to resolve the inconsistency,

or the processing modules generating the partial interpretation could be re-invoked to compute other alternative solutions.

## APPENDIX

In this section, we present proofs for the theorems presented in Section IV. The definitions and notations required for this section have been presented in the previous sections, and will be assumed.

THEOREM 1. *Consider a sequence of applications of the inference engines to a hypothesis $H$, if at the ith step engine $E_i$ is applicable, then $E_i$ will remain applicable at all inference steps $j > i$. That is, we can defer the application of an applicable inference engine.*

*Proof.* If $E_i$ is applicable at inference step $i$, then by the definition of **applicable**, $\exists$ a subset $K_i \subseteq H$ which satisfies all of $E_i$'s input requirements. As defined earlier, this entails:

- $K_i$ contains the required 3-dimensional relational tuples, and
- the required attributes of these tuples have previously been assigned values.

No application of any inference engine to $H$ removes any tuples from $H$ (note that the hypothesis generation and modification is a process distinct from the consistency evaluation part), and no application of any inference engine replaces a previously computed value for any attribute. Thus, at any later stage, $K_i$ will still be a subset of $H$ which satisfies the input requirement of $E_i$.

THEOREM 2. *If at step i, application of an inference engine $E_i$ would fail, then $E_i$ would fail even if its application is deferred to some step $j > i$.*

*Proof.* By Theorem 1, application of $E_i$ can be postponed, and whenever $E_i$ is reapplied, the values of all relational attributes that had values at step $i$ remain the same at step $j$.

By definition of failure, at step $i$, $E_i$ computed an inconsistent value for some attribute which had already been assigned a value. The computation is based on values of other attributes. Since the engine itself has not changed between inference steps $i$ and $j$, it follows that at step $j$, $E_i$ will compute the same value and therefore, by definition of a failed inference, it will still fail.

THEOREM 3. *Changing the order of application of inference engines does not cause a change in applicability of other engines.*

*Proof.* Let $E_k(E_{k-1} \cdots E_1(H) \cdots))$ and $E_j(E_{j-1} \cdots E_a(H) \cdots))$ be two sequence of inferences. Let the second sequence be a successful sequence (i.e., the application of $E_j(\cdot)$ succeeds).

Let these sequences satisfy the following conditions:

- Every $E_i$, $1 \leq i \leq k-1$ in the first sequence is also in the second sequence, though not necessarily in the same order.
- $E_k$ is not in the second sequence.

We show that, under these conditions, $E_k(E_j(E_{j-1} \cdots E_a(H) \cdots))$ is a valid sequence: i.e., $E_k$ is applicable at the end of the second sequence.

This follows from the fact that the condition of applicability of $E_k$ demands that certain attributes of certain tuples have non-"unknown" values. The application of the engines in the set $\{E_1 \cdots E_{k-1}\}$ form a sufficient condition for the determination of these values. Thus at the end of the second sequence, $E_k$ will be applicable.

THEOREM 4. *Any sequence of applications is either a terminating sequence or can be extended to form a terminating sequence.*

*Proof.* Let the sequence under consideration be $E_i(E_{j-1} \cdots (H) \cdots)$. The proof follows from the definition of a terminating sequence. If the last application $E_i$ fails, then by definition, the sequence is terminated. If it succeeds, then two cases arise. Either there is no other inference engine $E_k$ which is applicable to the result of the given sequence, which can be used to extend the sequence, or there is some engine $E_{i+1}$ which is applicable. In the first case, the sequence is again a terminating sequence. In the second case, apply the new engine to get $E_{i+1}(\cdot)$ as a new sequence. The same arguments that applied to the first sequence now apply to this sequence. Since the number of pairs of inference engines and the subset of the hypotheses on which they can be applied is finite in number, this process must terminate.

THEOREM 5. *If any one sequence of applications of a set of inference engines to a hypothesis terminates in failure then all possible sequences of applications terminate in failure.*

*Proof.* Let $E_k(\cdots E_1(H)\cdots)$ be a failed terminating sequence of inferences on hypothesis $H$. By definition, that means that $E_k(\cdot)$ fails. Let $E_{n\sim}(\cdots E_{i\sim}(E_{i-1}(\cdots(E_1(H)\cdots))))$ be a second sequence of inference engines in which the applications $E_1$ through $E_{i-1}$ are identical to those of the first sequence, and $E_{i\sim}$ is not the same as $E_i$. We will prove that this sequence, if extended to termination, will also result in failure. Without loss of generality (by Theorem 4), let the second sequence be a terminating sequence:

- *Case* I. If $E_{n\sim}$ fails, the proof is complete.
- *Case* II. If $E_{n\sim}$ succeeds, the second sequence is a successful terminating sequence. We will show that this leads to an inconsistency. The contradiction is by induction on the applications $E_i$ through $E_{k-1}$ of the first sequence.

By Theorem 3, either $E_i$ is present in the subsequence $E_{n\sim}(\cdots E_{i\sim}(\cdot)\cdots)$ or it is applicable after $E_{n\sim}$. Since the second sequence was considered a terminating sequence, it must be the case that $E_i$ occurs somewhere between $E_{i\sim}$ and $E_{n\sim}$.

Since $E_{n\sim}(\cdot)$ is a successful terminating sequence, the value computed for the output attribute of $E_i$ will be the same as that computed in the first sequence. To see this, note that the input terms for $E_i$ are computed by the applications of $E_1$ through $E_{j-1}$ which remain unchanged in the second sequence.

Thus the sequence $E_1$ through $E_{n\sim}$ contains all the engines from $E_i$ through $E_i$. Consider the sequence $E_i(\cdots E_{j-1}(\cdots(H)\cdots))$, which is the subsequence of the successfully terminating sequence $E_{n\sim}(\cdot)$, and the sequence $E_i(E_{i-1}\cdots(H)\cdots)$, which is a subsequence of the unsuccessfully terminating sequence $E_k(\cdot)$. since $E_{i+1}$ is applicable at the end of $E_i(\cdots(H)\cdots)$ in the second subsequence, by Theorem 3, it must also be applicable after the end of the first subsequence.

This argument can be inductively extended to show that all engines $E_j$, $j = 1 \cdots k$, must be in the sequence $E_1$ through $E_{n\sim}$ and that all attributes computed as the input to $E_k$ must have the same values as they did in the first sequence.

In the first sequence, the application of $E_k$ failed. Therefore the sequence $E_1 \cdots E_{k-1}$ must also compute a value for some attribute that contradicts the value computed by $E_k$. Let $E_j$ for some $j$ between 1 and $k-1$ be the engine in the first sequence that computed the contradictory value. We have shown that it computes the same values in the second sequence as it does in the first. By the same argument, so does $E_k$. Since these two values were contradictory in the first sequence, they must also be contradictory in the second, leading to some failed inference. This contradicts the assumption that the second sequence was a successfully terminating sequence.

THEOREM 6. *If any one sequence of applications terminates successfully, then all possible sequences terminate successfully.*

*Proof.* Follows as a corollary to Theorem 5.

THEOREM 7. *Ignoring permutation, there is at most one successful sequence of applications of a set of inference engines to a hypothesis.*

*Proof.* An argument exactly parallel to the proof of Theorem 5 shows that if there are two successful terminating sequences of inferences for a given hypothesis $H$, which differ by some inference engine applications, they cannot both be terminating sequences.

## REFERENCES


1. S. T. Barnard, *Interpreting Perspective Images*, Technical Note 271, A.I. Center, SRI International, Menlo Park, Ca., 1982.
2. R. A. Brooks, Symbolic reasoning among three dimensional models and two dimensional images, *Artif. Intell.* **17** (1981) (special issue on computer vision).
3. I. Chakravarty, *A Generalized Line and Junction Labeling Scheme with Applications to Scene Analysis*, CRL-55, Rensselaer Polytechnic, December 1977.
4. C. Chang and R. C. Lee, *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.
5. M. B. Clowes, On seeing things, *Artif. Intell.* **2**, No. 1, 1971.
6. L. S. Davis, L. Janos, and S. M. Dunn, Efficient recovery of shape from texture, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-5**, No. 5, 1983, 485–492.
7. J. V. deVries, *Perspective*, Dover, New York, 1968.
8. A. Guzman, *Computer Recognition of Three-Dimensional Objects in a Visual Scene*, Tech. Rep. 59, AI Laboratory, MIT, Cambridge, Mass., 1968.
9. R. M. Haralick and G. Elliott, Increasing tree search efficiency for constraint satisfaction problems, in *6th. Int. Joint Conf. Artif. Intell.*, Tokyo, Japan, 1979.
10. R. M. Haralick, Using perspective transformations in scene analysis, *Comput. Graphics Image Process.* **13**, 1980, 191–221.
11. B. P. K. Horn, Shape from shading, *The Psychology of Computer Vision* (P. H. Winston, Ed.), McGraw–Hill, New York, 1975.
12. B. P. K. Horn, Understanding image intensities, *Artif. Intell.* **8**, 1977, 201–231.
13. D. A. Huffman, Realizable configurations of lines in pictures of polyhedra, in *Machine Intelligence*, Vol. 8, Edinburgh Univ. Press, Edinburgh, 1977.
14. T. Kanade, Recovery of three-dimensional shape of an object from a single view, *Artif. Intell.* **17** (1981).

15. S. J. Lee, R. M. Haralick, and M. C. Zhang, Understanding objects with curved surfaces from a single perspective view of boundaries, *Artif. Intell.* **26**, 1985, 145–169.
16. P. G. Mulgaonkar, *Analyzing Perspective Line Drawings Using Hypothesis Based Reasoning*, Ph.D. thesis, Virginia Polytechnic Institute and State University, 1984.
17. P. G. Mulgaonkar, L. G. Shapiro, and R. M. Haralick, Matching sticks plates and blobs objects using geometric and relational constraints, *Image Vision Comput.* **1**, 1984.
18. A. P. Pentland, Local shading analysis, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-6**, No. 2, 1984, 170–187.
19. L. G. Roberts, *Machine Perception of Three-Dimensional Solids*, Report 315, Lincoln Laboratory, MIT, Cambridge, Mass., 1963; *Optical and Electro-Optical Information Processing* (J. Tippett *et al.*, Eds.), pp. 159–197, MIT Press, Cambridge, Mass., 1965.
20. D. Waltz, Understanding line drawings of scenes with shadows, *Psychology of Computer Vision* (P. Winston, Ed.), McGraw–Hill, New York, 1975.
21. A. P. Witkin, Recovering surface shape and orientation from texture, *Artif. Intell.* **17**, 1981.