Fast Visual Recognition of Large Object Sets

by

Michael Joseph Villalba

B.S., Rensselaer Polytechnic Institute (1981)
S.M., Massachusetts Institute of Technology (1984)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1990

Signature of Author .... [illegible]
Department of Aeronautics and Astronautics
February 1, 1990

Certified by .... [illegible]
Professor Tomaso Poggio
Uncas and Helen Whitaker Professor
Department of Brain and Cognitive Sciences
Thesis Supervisor

Certified by ....
Professor Wallace Vander Velde
Department of Aeronautics and Astronautics
Doctoral Committee Chairman

Certified by .... [illegible]
Professor Rodney Brooks
Department of Electrical Engineering and Computer Science

Accepted by..... ....
Professor Harold Y. Wachman
Chairman, Department Graduate Committee

Fast Visual Recognition of Large Object Sets

by

Michael Joseph Villalba

Submitted to the Department of Aeronautics and Astronautics
on February 1, 1990, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

An approach to three-dimensional object recognition tailored to large object sets is presented. It varies from others in that representations encoding qualitative instead of precise information are employed, resulting in finite-view descriptions of 3D objects. With no dimensional mismatch between image and model reference frames, recognition occurs quickly on the basis of information immediately extractable from images, and thus becomes feasible for large object sets.

Qualitative representations are justified because coarse information often distinguishes objects. Yet unique identification can not be guaranteed since qualitative characterization does not capture fine distinctions among objects. So qualitative recognition can be viewed as a example of *indexing*, where simple information is used to quickly and cheaply eliminate dissimilar objects from consideration. With a small number of similar objects remaining, application of more detailed and costly procedures yielding unique identification becomes feasible.

Indexing is demonstrated on line drawings of a set of sixteen library objects. Representations are obtained by uniformly sampling the viewing sphere. The simplicity of resulting descriptions allows parallel matching over all known objects on fine-grain parallel hardware. Effective indexing is demonstrated for isolated scene objects, typically resulting in little ambiguity among library objects. When multiple objects are present, accurate object segmentation is needed to avoid increased ambiguity due to segmentation error.

Thesis Supervisor: Tomaso Poggio
Title: Uncas and Helen Whitaker Professor
Department of Brain and Cognitive Sciences

Contents

Acknowledgments

Tomaso Poggio gave me the opportunity to perform vision research at the M.I.T. Artificial Intelligence Laboratory, and was a source of consistently useful insight and advice. This work is substantially enriched by Tommy's supervision.

My gratitude goes as well to Wallace Vander Velde of the Department of Aeronautics and Astronautics for his guidance in navigating the Department's doctoral program. I am honored that Professor Vander Velde was able to serve as my doctoral committee chairman.

I am also indebted to Rodney Brooks for his thoughtful participation as a member of my doctoral committee.

Patrick Sobalvarro provided many useful comments regarding thesis content, and I feel the presentation is much improved as a result.

Finally, I would like to thank my family, especially my parents, for their love and support during the many good and bad times which define a doctoral program.

This report describes research done within the Center for Biological Information Processing, in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research is sponsored by a grant from the Office of Naval Research (ONR), Cognitive and Neural Sciences Division; by the Artificial Intelligence Center of Hughes Aircraft Corporation; by the Alfred P. Sloan Foundation; by the National Science Foundation; by the Artificial Intelligence Center of Hughes Aircraft Corporation (S1-801534-2); by E.I. DuPont DeNemours and Company; and by the NATO Scientific Affairs Division (0403/87). Support for the A. I. Laboratory's artificial intelligence research is provided by the Advanced Research Projects Agency of the Department of Defense under Army contract DACA76–85–C–0010, and in part by ONR contract N00014–85–K–0124. Tomaso Poggio is supported by the Uncas and Helen Whitaker Chair at the Massachusetts Institute of Technology, Whitaker College.

Chapter 1

Introduction

Visual recognition takes place by comparing information extracted from imagery with stored representations identifying known objects. Assuming unconstrained viewpoint, the success of recognition requires that representations encode object appearance from all possible perspectives. One approach is to store precise 3D geometric information in an object-centered reference frame, perhaps via computer-aided design, resulting in a conceptually compact representation implicitly encoding appearance from all viewpoints. Serious problems result, however, the most significant being the difficulty and computational expense of converting object-centered model information and viewer-centered image data to a common domain for comparison.

Here qualitative object representations, meaning those encoding coarse instead of precise information, are motivated. The primary advantage is that qualitative object appearance can be captured by a finite number of views, hinting that fast recognition, without the need for conversion between object- and viewer-centered frames, is possible. Furthermore, reduction of 3D objects into viewer-centered descriptions greatly simplifies learning, which then becomes the relatively simple task of compiling qualitatively different object views.

But qualitative representation is not guaranteed to result in unique identification because encoded information may fail to capture subtle object differences. Thus a two-stage paradigm is proposed, where qualitative recognition is used as an *indexing* step designed to retrieve a small number of similar objects from a large set of known possibilities. Unique identification could then be achieved by more detailed examination of these objects using a procedure too expensive to be applied directly to large object sets.

This chapter serves primarily as an introduction to the qualitative approach, and includes brief surveys of previous recognition systems and related psychophysics. The chapter concludes with an overview of those following.

1.1 Motivation

One of the most remarkable characteristics of the human visual system is its ability to quickly recognize large numbers of 3D objects, perhaps on the order of tens of thousands [5]. In contrast, current work in computer vision typically considers only one object [65], or perhaps a few very simple ones [37]. Perhaps the view is widespread that algorithms designed for small object sets will naturally extend to solutions for the many-object case.

But there seems to be little progress in this direction. The goal of this work is to address the issue more directly by considering from the beginning the requirements of large object sets in order to obtain recognition strategies more appropriate than brute-force application of techniques designed for small sets.

The most important requirement regards computation. If recognition proceeds by considering known objects serially, then for the sake of speed each can not require much processing. More plausibly, recognition could occur in parallel over the set of known objects. In theory this would allow more extensive computation for each object. However, currently available parallel processors are relatively simple and limited, restricting allowable computation. For example, the Connection Machine model CM-2 parallel supercomputer uses processors having one-bit arithmetic units and only 64K bits of memory each. Thus for parallel as well as serial implementations, recognition using available hardware cannot require large amounts of computation (per object).

Current 3D recognition algorithms are limited to serial implementations precisely because their computational requirements are beyond the capabilities of available parallel processors. Furthermore, typically recognition of at most a few objects is attempted because consideration of each is lengthy. Several of these techniques will be reviewed in detail in the next section, but for now it suffices to note that many are founded on the use of object representations encoding precise 3D information in object-centered reference frames, meaning those rigidly attached to objects [47].

Such representations present a computational bottleneck for recognition because image information is embedded in 2D viewer-centered (retinocentric) frames, meaning those defining spatial relations on the viewer's retina [47]. Since recognition occurs by comparing imagery with stored object representations, information stored in 2D and 3D reference frames must be converted to a common domain for comparison. As will be discussed, this typically requires large amounts of computation, perhaps much more than is required for comparison once conversion is complete.

It would seem then that object-centered representations fundamentally hinder the performance of recognition systems. One possible remedy suggested by Grimson and Lozano-Pérez [29, 30] and Grimson [27] is to use simple geometric constraints to efficiently prune scene interpretations, thereby restricting the number of 3D geometric models to be considered. An alternative is to represent objects in terms of retinocentric information, meaning collections of view descriptions. With this approach, there is no need for conversion of image and model data since both are defined in viewer-centered frames. Of course this is not a new idea: the industrial-part recognition literature yields many such techniques [13]. However, they typically apply to heavily constrained environments [58] involving controlled illumination and viewpoint, resulting in representations such as the iconic model [59, 2], which essentially consists of images obtained at allowable viewpoints. More general applications involving arbitrary viewpoint are not addressed primarily due to concern that many views would be required to capture object appearance from all possible perspectives [47, 58, 13].

But that may be exactly what is desirable, at least for parallel implementations. Instead of relying on a single 3D object-centered representation to implicitly encode all possible views, a collection of retinocentric descriptions could be used to implicitly encode 3D structure. Recognition then reduces to comparing stored view descriptions with imagery. Since view descriptions are independent, each could be assigned to a distinct processor, which would then have a lightened computational burden because reference frame transformations would no longer be needed. Thus, instead of assigning an object-centered representation to each processor, which is probably not computationally feasible for any but the most trivial objects, representing objects retinocentrically allows the computational burden to be distributed among multiple processors, permitting recognition to occur in parallel in a manner potentially within the computational capabilities of current parallel hardware.

Explicitly encoding viewer-centered instead of object-centered information also helps address another important recognition issue, namely object learning. Since representations consist of collections of view descriptions, learning simply becomes a matter of sampling object appearance from different viewpoints.

Thus viewer-centered representations are attractive because they ease the computational burden of recognition, and because they simplify object learning. But little has been said to this point concerning the nature of view descriptions. Assuming an arbitrary viewpoint, all perspectives of each object must be captured by a finite number of view descriptions. This criterion rules out descriptions encoding information which varies continuously with viewpoint. In general, precise geometric information falls in this category and thus the iconic model as well as most other published viewer-centered representations [13] are not appropriate. Instead, object features can be described *qualitatively*, meaning here that all possible appearances are partitioned into a finite number of distinct descriptions. A finite number of observed features, each described qualitatively, results in a finite number of possible view descriptions.

Furthermore, feature descriptions must be stable with respect to viewpoint. Otherwise, blind sampling of viewpoints is not likely to result in satisfactory object representations, making learning impractical.

To make the notion of qualitative feature characterization concrete, consider the machined part pictured in Figure 1-1. Such line drawings can be segmented into collections of digitized curves, which become reasonable choices for the primitive

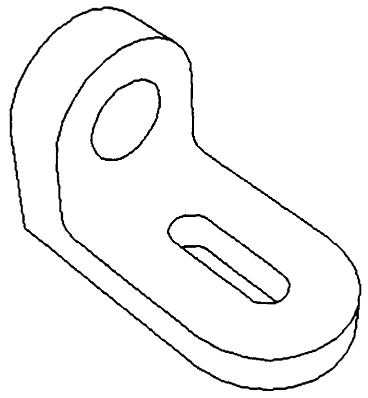


Figure 1-1: A digitized line drawing.

features of view descriptions. The problem is to characterize these curves such that the number of possible descriptions is finite. Strictly speaking the curves themselves are qualitative entities since there are a finite number of distinct digital curves possible in any image array of finite size. But describing curves by listing constituent pixels is not useful due to viewpoint instability.

A much more reasonable and perceptually significant approach is to code each curve according to number of concavities. Here "concavity" refers to a curve interval of apparently monotonic orientation variation, without regard for the direction of variation (there is no notion of "inside" or "outside"), and possibly including corners. Several curves with equal numbers of concavities and therefore identical qualitative descriptions are pictured in Figure 1-2. We can reasonably expect such a description to be quite insensitive with respect to viewpoint, as is suggested by comparing corresponding curves in Figure 1-3. And significantly, we shall see in the next chapter that number of concavities is a quantity easily and robustly extracted from digitized curves, which is not the case with many other curvature evaluation methods.

Qualitative descriptions of a quite different nature can be obtained simply by quantizing continuous variables. In the context of line drawing analysis, such variables might include curve length and angular separation of curves at intersections. Here qualitative characterization provides an additional benefit, namely noise suppression

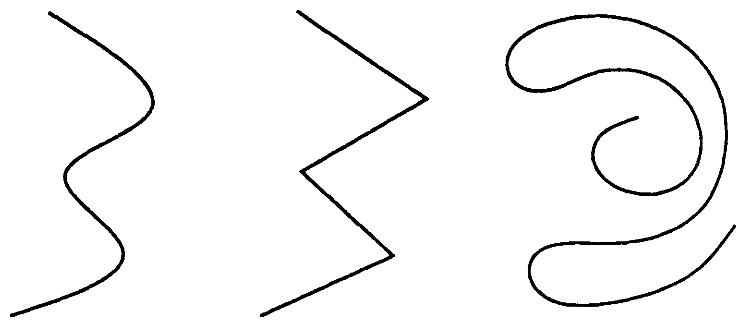


Figure 1-2: Digital curves with three concavities each.

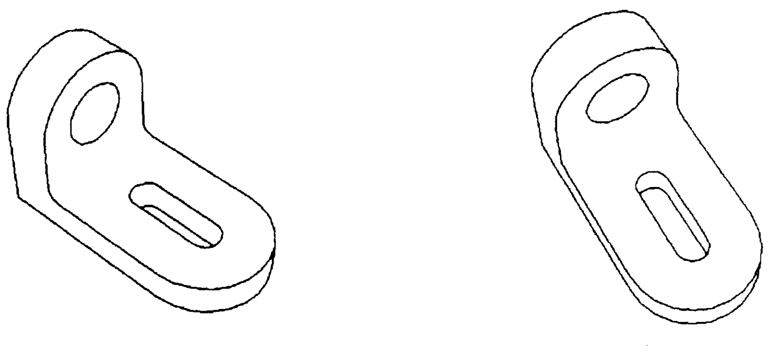


Figure 1-3: Two views having qualitatively identical curves.

due to quantization.

To recapitulate, qualitative descriptions yield a finite number of feature classifications, allowing 3D objects to be represented by finite numbers of views. Resulting object representations encode 3D information implicitly in terms of view descriptions, making learning and fast recognition of large sets of 3D objects feasible.

1.2 Recognition Paradigm

Qualitative representation is justified because objects of interest often differ radically. But ambiguity among similar objects can be expected since coarse characterization may fail to capture subtle object differences. Thus qualitative representation is not guaranteed to result in unique identification. Instead qualitative recognition must be followed by a second recognition stage employing more precise object information. Thus the recognition paradigm proposed here involves an *indexing* stage employing qualitative information to retrieve a small number of grossly similar objects from the known set, followed by a verification stage employing detailed information to yield unique identification[1].

Two stages are proposed for efficiency. Since it involves manipulation of simple information, indexing occurs quickly, leaving more costly verification to a significantly reduced set of possibilities. But the burden of verification is in turn reduced because detailed information is required to distinguish only similar objects of the known set, and thus can be rather specific, reducing the amount of data manipulation required. One possibility is to structure verification as a tiny expert system which applies detailed information based on the class of objects returned by indexing [52]. Detailed information is then applied only if relevant, thereby improving verification efficiency.

Verification will not be considered in detail in the remainder of this work. Instead emphasis is placed on achieving effective qualitative indexing. Thus when the term "recognition" is used, it often refers to the first stage of the paradigm.

[1]The reader will note the similarity between this scheme and the approach proposed by Ullman [67], where "universal routines" applied in the absence of a priori information are used to obtain initial classification, followed by selective application of specialized routines.

1.3 Previous Work

Recognition algorithms can be distinguished based on the extent of scene modelling required. Recent systems using strong scene assumptions are the context-dependent recognizers of Silberberg [60], and of Harwood, Prasannappa, and Davis [32]. Others are discussed in the survey by Binford [6]. Since context often serves to define a small number of observable objects, such techniques circumvent the problems associated with large object sets, and are not of interest here. Instead, the more general case where context is unknown will be addressed.

Without scene modelling, rich object representations capable of distinguishing objects in unknown positions and orientations become necessary. Representations can be divided into two classes depending on whether object information is stored in viewer-centered or object-centered reference frames [13]. Viewer-centered representations consist of collections of view descriptions, and have the advantage that comparison of image and model information does not require spatial transformations. But the recognition algorithms receiving the most attention recently use object-centered models.

Object-centered representations are favored for several reasons [47, 13, 3, 6]. Foremost is the implicit encoding of all object views, enabling recognition from arbitrary perspective. Additionally, in industrial settings computerized design procedures can result in prior availability of such representations. The main disadvantage is the reference frame mismatch between model and image data, forcing spatial transformations during recognition.

Roberts pioneered the use of object-centered representations with the publication of his efforts in polyhedron recognition [56]. The input to his system consists of line drawings of one or more objects, possibly extracted from imagery using edge detection techniques. Since extraction of object-centered information from imagery in general and line drawings specifically is non-trivial, matching is performed by transforming object-centered model information to the image domain. This requires filtering a finite number of plausible transformations from the infinity of possibilities.

Hypothesized transformations are obtained by postulating correspondences between line drawing and model points. The only points considered are endpoints of lines forming model or drawing polygons, excluding those belonging to image T-junctions. The technique employed requires at least four pairs of corresponding points to allow estimation of object position, orientation, and scale (in three axes) in a camera-centered reference frame. Model information can then be projected into the image domain using the assumed perspective imaging model.

All possible image/model point pairings can be used, but to further reduce the number of model transformations to be considered, only topologically equivalent points are paired. For example, one point-matching technique requires polygons to be classified based on number of sides. Points are paired if their surrounding polygons match. Lists of pairs passing this test are constructed, each resulting in a hypothesized transformation of the model under consideration.

Having obtained a finite set of plausible transformations for each model, object matching proceeds in the image domain. First, a measure of the fit of model to drawing points obtained by the least-squares transformation estimation procedure is obtained, and all models whose error measures are too large are discarded. Second, all model points are projected into the drawing, resulting in the elimination of all models having points falling outside the boundary of the image object. All models passing these two tests are considered valid.

Thus we see that Roberts' recognition algorithm consists of two parts. First, a finite number of object views are guessed by postulating correspondences between image and object features. Second, matching occurs in the image domain by comparing hypothesized views to the image in question. Roberts proposed this paradigm in 1965, and it has continued to receive considerable attention [12, 35, 33, 25, 37, 38].

Consider for example Huttenlocher's recent thesis [37], which also considers polyhedra, as well as a limited class of curved-surface objects. In contrast to the perspective imaging model used by Roberts, Huttenlocher uses an approximate model, namely "weak perspective" projection, consisting of orthographic imaging plus a scale factor to approximate object size variation with distance under perspective projection. The work is founded on the claim that curvature zeroes of 3D space curves are preserved under orthographic projection [2], allowing pairs of image and model edge segments bounded by curvature zeroes to be placed in postulated correspondence (here an image edge corresponds to a grey-scale intensity discontinuity, and a model edge to a surface orientation discontinuity). One or two such pairs are used to hypothesize object orientation and position in a camera-centered reference frame. For each hypothesized transformation, matching takes place by projecting model edges into the image domain. Transformations for which a "certain percentage" of model edges match image edges are accepted.

Goad [25] considers a variation of the Roberts paradigm where viewpoint hypothesis and object matching are performed iteratively instead of in discrete stages. Each step of the iteration begins by predicting the image position of an object edge (not previously considered) using the current viewpoint hypothesis. If an image edge is nearby, a match is assumed, and the difference between predicted and actual position is used to refine the viewpoint estimate. The process is repeated until enough edges are matched to infer object identification reliably.

A popular approach using object-centered models distinct from Roberts' paradigm employs the generalized Hough transform [1, 2]. As in Roberts' paradigm, model to image transformations are obtained by postulating matches between image and model features. Typically all possible pairings are checked, resulting in a large set of potential transformations. Since each usually involves six degrees of freedom (three each for orientation and position in the camera reference frame), transformations can be represented in a six-dimensional parameter space. Since similarity of incorrect transformations is unlikely, clusters in parameter space ideally signal correct transformations. Thus, transformations are verified not by matching in the image domain, as in Roberts' paradigm, but instead by cluster detection in parameter space [16, 65, 19, 9].

A recent example of the generalized Hough transform approach is the work of Connolly et al. [16], which employs polyhedral object models and piecewise linear

[2]Although Huttenlocher provides intuitive justification for this claim [37, pages 92–93], no rigorous proof is given. Certainly its validity does not extend to all viewpoints, as it is easy to imagine situations where a 3D curvature zero disappears when viewed along its tangent.

approximations of image edges. The matching features are "vertex-pairs", each of which consists of two vertices (edge intersections) plus two edges from one of the vertices. The two vertices do not have to be connected by an edge. Assuming weak perspective imaging, a single pair of corresponding image and model vertex-pairs results in a six degree of freedom transformation of the object model into the camera reference frame. Transformations are calculated for every pairing of image vertex-pairs to a subset of the model vertex-pairs. The most plausible is determined by finding clusters in the resulting six dimensional parameter space.

Roberts' paradigm and the generalized Hough transform technique together account for most recent work in arbitrary-view 3D object recognition. Both raise critical issues regarding the generality, and suitability for large object sets, of algorithms using object-centered geometric representations. Consider for example the the necessary step of hypothesizing object viewpoints. As we have seen, this typically requires pairing image features with 3D features fixed on objects, such as those associated with surface orientation discontinuities. However, image features do not necessarily correspond to fixed 3D features. Examples include T-junctions generated by self-occlusion, and occluding boundaries generated by smooth surfaces. These cannot be used to hypothesize object viewpoints. Thus smooth objects cannot be identified, even in the presence of characteristic image features[3].

Other problems appear during the viewpoint verification stage. Consider Huttenlocher's algorithm, for example, which verifies hypothesized viewpoints by projecting model edges into the image domain. For general objects, this requires hidden line elimination and limb[4] detection. Such calculations are lengthy: rendering of individual objects in Figure 1-4 typically takes several minutes on a Symbolics model 3650 Lisp Machine. Thus few hypothesized views of general objects can be considered in practice, especially when considering large object sets. However, even the simple

[3]Roberts' addressed only polyhedron recognition. This point is mentioned only to indicate one of the difficulties in extending his approach to more general objects.

[4]Limbs are collections of points whose lines of sight are surface tangents. They usually project into image edges, making their detection mandatory in Huttenlocher's verification scheme. Since limb location is a function of viewpoint, precomputation is not possible, as it is with surface orientation discontinuities.

polyhedra considered by Huttenlocher require on the order of a thousand hypothesized views each when in typical scenes [37, page 123], suggesting that recognition of more complicated objects such as those in Figure 1-4 is simply not feasible. Of course it might be argued that each view could be considered in parallel, but this would require thousands of powerful processors for each known object, making consideration of large object sets unlikely.

To avoid the rendering bottleneck, Huttenlocher uses a representation very similar to the characteristic view approach of Chakravarty and Freeman [11]. A characteristic view is representative of a set of object views defined by isomorphism with respect to line structure. Since object-centered positions of corresponding 3D edges are included with characteristic views, all views in isomorphic sets can be obtained from corresponding characteristic views without hidden line removal or limb detection. But this assumes that there is no self-occlusion and that all image edges are projections of object edges. When these assumptions are violated the characteristic view approach is not applicable. For example, limbs are not surface-fixed so their projections can not be predicted via the characteristic view approach. When present, limbs must be detected at all viewpoints[5]. And self-occlusion causes point visibility to vary, requiring hidden line removal. Thus the characteristic view approach allows efficient and accurate rendering only of objects without significant limbs or self-occlusion, eliminating a huge class of objects from consideration (all objects in Figure 1-4 for example). Fast viewpoint verification is obtained, but at the cost of severely restricting allowable objects.

The generalized Hough transform approach performs viewpoint verification by clustering hypothesized model-to-image transformations. However, clustering appears to be quite costly. For example, the algorithm of Connolly et al. implemented on the Connection Machine by Thompson and Mundy [65] requires roughly several minutes for each model vertex-pair considered. Thus a small set of model features must be chosen in advance, taking into consideration visibility and robustness of resulting

[5]Limb position could also be estimated, but this requires 3D surface curvature to be stored with characteristic views [68].

viewpoint estimates [50]. Even if a small set is used, the ability to handle large object sets is questionable, as parallel recognition of all known objects is not currently possible due to the large number of processors required by each object[6].

In addition to problems with generality and efficiency, recognition algorithms using object-centered representations suffer from difficulties associated with model generation. Specifically, the above systems are not able to "learn from experience" [3]. That is, when presented with an unknown object, they are not able to extract enough information to recognize the object if again seen from a similar viewpoint. Instead they rely on models provided by humans[7]. The problem appears fundamental, apparently due to the difficulty in forming 3D object-centered models from viewer-centered data.

Algorithms using viewer-centered representations do not suffer from these problems, at least not to the same extent. Because each object is represented by a set of view descriptions, there is no need for a separate viewpoint hypothesis step, eliminating associated object restrictions. Furthermore, recognition is not slowed by the extraction of views from object-centered models during matching. Instead, views are compiled during learning, which becomes trivial since representations encode appearance explicitly. This is particularly useful when objects have surface markings or other distinguishing features not related to geometry, since these are usually difficult to include in 3D models.

Of course, the use of viewer-centered object representations is not a new idea. They are in fact employed by most industrial vision systems [13]. However, their application appears limited to those situations involving a small number of fixed viewpoints. The problem seems to be that typically such representations encode information which varies continuously with viewpoint, resulting in the need to store an infinite number of views to capture the appearance of each object from all possible perspectives. An example is the representation used by Schwartz and Sharir [57],

[6]Detailed analyses of additional problems with the generalized Hough transform are presented by Grimson and Huttenlocher [28].

[7]Although acceptable in controlled environments where all possible objects are known in advance, manual generation of a large number of objects may prove tedious, making the use of human generated models impractical (except where they already exist, perhaps as a result of the industrial design process).

which uses boundary curves to represent object views.

The obvious solution is to store a finite number of such views, using an error tolerance to allow each stored view to account for the viewpoints in some non-zero volume, hopefully resulting in a covering of all allowable viewpoints. An alternative is to encode coarse qualitative instead of precise quantitative information, resulting in a finite number of possible descriptions for each view feature. In each case the result is essentially the same, namely that a finite number of stored views encodes object appearance from all perspectives. The difference is in the relative ease and robustness expected of information extraction in the qualitative case.

Weinshall [72] recently considered this issue in detail by studying the extraction of depth information from pairs of corresponding points in stereo imagery. She concluded that extraction of precise depth is much harder and less reliable than extraction of qualitative depth, here meaning an ordering of points with respect to depth. Such results are expected since qualitative characterization involves loss of information, suggesting simpler acquisition and/or reduced sensitivity to noise.

Qualitative object representations are actually found quite frequently in the literature. Often cited is the "visual potential" of Koenderink and van Doorn [43], which is founded on a topological view classification known as "aspect." Because aspect is a qualitative description stable with respect to viewpoint almost everywhere, the space surrounding an object can be divided into mutually exclusive open sets, each corresponding to one of a finite number of aspects. All aspects are incorporated into the visual potential, which is a connected graph whose nodes represent aspects, and whose edges connect spatially adjacent aspects. The visual potential thus represents appearance from arbitrary viewpoints. In particular, any observer trajectory corresponds to a path through the visual potential.

Applications using this style of representation include Ikeuchi and Kanade [39], and Burns and Kitchen [8]. Ikeuchi and Kanade use aspects to automatically generate recognition strategies, but their primary interest is not recognition speed. Instead they are concerned with compensating for sensor effects, aspects being a convenient representation for doing so.

The work of Burns and Kitchen is actually quite similar in scope to the present work. Primarily both are concerned with large object sets, and conclude that object-centered models are incompatible with the need for speed. Instead, viewer-centered representations are employed to avoid costly transformations between image and object-centered reference frames during recognition. But the similarity ends there. Burns and Kitchen's most significant diversion from this work is their collective storage of all view descriptions of all objects in a single hierarchical graph. The motivation for doing so is economy of storage and matching: identical features found in several views are stored once, and during recognition matched once. However, the natural parallelism implied by an independent data structure for each stored view is destroyed, posing recognition as a complicated search of the collective graph, conducted by serially seeking matches to image features. Robustness thus becomes an issue, as missing image features could block the graph search. Furthermore, the addition of new objects to the known set increases the size of the graph, making recognition of any one object necessarily more expensive. Thus the loss of parallelism results both in potentially catastrophic sensitivity to noise, and in reduced speed for larger object sets.

Qualitative representations can also be found in the sequential pattern recognition literature [22]. Pattern recognition in the context of the present work can loosely be defined as identifying objects by measuring and classifying image features. Classically the feature set is considered a vector in partitioned n-space, where each subset corresponds to a known object. Recognition proceeds by determining feature vector membership. In single-stage classifiers, partition subsets are checked directly for feature vector presence, a potentially difficult task since large object sets could result in complex partitions. A different approach is used by sequential pattern recognition, which considers individual features separately. A commonly used object representation in this case is the "decision tree" [71, 63, 4], which collectively and qualitatively represents all known objects by partitioning the domain of each feature into a small number of subsets. Recognition proceeds by sequentially locating the partition in which each feature lies, eliminating from future consideration known objects whose corresponding features lie elsewhere. The procedure ends when a single known object remains.

The motivation for using decision trees is not that their qualitative nature allows finite-view representations of 3D objects. I am not familiar with any applications considering arbitrary-view recognition. Instead, the simple feature partitions together approximate the complex n-space partition, allowing classification via simple, local decisions. Thus recognition is potentially fast, but for reasons having nothing to do with facilitation of finite-view representations.

Decision trees can be thought of as collections of filters, each of which uses simple, coarse information to shrink the set of possible objects. Such indexing behavior can be found elsewhere, for example in the algorithms of Kalvin et al. [41] and Jacobs [40]. Kalvin et al. address the recognition of 2D objects, representing objects by their boundary curves. Recognition is performed by matching observed boundaries to those of known objects, using the algorithm of Schwartz and Sharir [57]. Because of the impracticality of matching all boundaries (especially when from a large library), Kalvin et al. propose indexing via "footprints," which are mappings of boundaries into rotationally and translationally invariant five-dimensional curves, to reduce the number of boundaries considered. Footprint space is quantized, the resulting 5D hypercubes storing the identities of objects with intersecting footprints. Only those library boundaries found in hypercubes traversed by the observed boundary footprint are considered for matching. Again we see that objects are represented qualitatively, namely by lists of hypercubes in footprint space, to facilitate indexing.

Jacobs also considers 2D object recognition, similarly proposing indexing followed by boundary matching. Exact boundaries are not considered, and are instead replaced by polygonal approximations. Indexing employs segment pairs from observed boundary polygons. Since five parameters determine the relationship between two segments, each pair can be represented by a point in five-dimensional space. This space is quantized, each resulting hypercube storing the library objects whose segment pairs it contains. Indexing proceeds by selecting a segment pair from the observed boundary, noting which library objects have segment pairs residing in the same hypercube. Only those objects are considered for boundary matching.

To summarize this section, we have seen that most algorithms for arbitrary-view recognition of 3D objects employ object-centered models, resulting in slow algorithms not appropriate for objects without significant surface-fixed features. The essential problem seems to be the dimensional mismatch between 3D object-centered representations and 2D viewer-centered data. A fairly obvious solution is to represent objects in terms of views, eliminating the mismatch problem and allowing object recognition based solely on appearance. This obviates the need for detectable, surface-fixed 3D features such as surface-orientation discontinuities, and facilitates learning of new objects. Viewer-centered descriptions have actually been used for some time, but because precise quantitative information is often encoded, complete representation of object appearance requires an infinity of stored views. Recently this issue has been addressed by encoding coarse, qualitative instead of precise, quantitative information. Coincidentally, qualitative information has frequently been associated with those algorithms seeking speed, not via 2D representation of 3D objects, but by indexing, meaning the use of coarse information and simple decisions to achieve recognition through "process of elimination." Although this may seem counterintuitive, meaningful initial categorization is potentially realizable after only a small number of indexing steps [54].

1.4 Psychophysics

In addressing all but the most trivial recognition problems, prudence dictates consideration of human sensory performance. It is not that we require artificial recognition systems to mimic mental processes, but instead because perceptual limitations may provide algorithmic hints not available elsewhere.

Above I discuss at length certain motivations, primarily computational, for using coarse, qualitative instead of precise, quantitative representations. At least as compelling is the imprecise nature of absolute quantitative judgments (comparisons against metric standards in memory) made by humans [48, 5]. In his classic survey, Miller [48] observes that for a wide range of sensory variables precision is limited to between four and ten distinguishable alternatives. Reliable judgment of line segment length, for example, appears limited to about eight categories.

In contrast, the precision of comparative judgments (those where two stimuli are observed simultaneously) appears several orders of magnitude greater [5]. For example, extremely small differences in length can be detected between two parallel line segments simultaneously visible. It can be concluded then that the coarseness of absolute quantitative judgments is not a result of poor sensory resolution, but is instead due to memory limitations restricting the precision of stored quantitative representations. Thus it is doubtful that memorization of object-centered models encoding precise quantitative information is possible because the brain does not appear to allocate enough memory[8].

Note, however, that coarse categorization of quantitative information is fully consistent with the notion of qualitative representation. But viewer-centered representations, although enabled by qualitative information, are not suggested directly. Koenderink and van Doorn provide justification by reinterpreting the results of mental rotation experiments [43, page 216], which ironically are among the primary psychophysical justifications for object-centered models [37]. In these experiments, an observer is asked to determine whether two objects, each in a different image, are identical. When the objects are identical, the observer's response time is reported to be proportional to the extent of rotation required to align one view with the other, suggesting that matching is performed via mental rotation of a 3D model. Based on their observation that response time depends on object complexity, loosely meaning number of aspects, Koenderink and van Doorn dispute this interpretation, suggesting that the relevant variable may not be angle of rotation but instead the number of aspects that must be visited in bringing one view into correspondence with the other.

In suggesting a viewer-centered interpretation, Koenderink and van Doorn provide us with a plausible alternative to the common interpretation that views these

[8]Even if memory limitations were not suggested, the use of precise quantitative information in recognition is implausible because absolute judgments typically require more time than recognition itself [5].

experiments as convincing evidence of 3D object-centered mental representations[9]. In their words, "It is not necessary to suppose that the internal model of a cube is a little cube in the head." Nor are we compelled to do so, as the well-known limits on human absolute judgment clearly suggest that precise, quantitative sensory information is not memorized.

1.5 Overview

The broad goal of this work is to present a recognition strategy for large object sets. Specifically, the role of qualitative object representation is considered. Qualitative information is suggested because it allows viewer-centered representation of 3D objects. The result is an indexing algorithm which retrieves a small number of similar possibilities from the known object set. The algorithm is presented as follows.

Chapter 2 considers the extraction of qualitative information from line drawings like that in Figure 1-1. Primarily curvature information is discussed due to its perceptual significance and invariance with respect to scaling and rotation in the image plane. Natural images are not considered because significant low-level vision problems exist, such as reliable extraction of edges and their junctions [53]. Consideration of line drawings, which quite possibly serve as intermediate representations in human vision [69, 46], allows discussion of recognition strategies without entanglement in low-level issues.

Because qualitative representations discard information, there is no guarantee that their use allows objects to be distinguished. Such is not the case with precise object-centered descriptions, which essentially define represented objects. This issue is addressed in Chapter 3, where recognition is viewed as a communication problem using ideas from information theory [51, 23]. Specifically, qualitative descriptions are viewed as codes required to distinguish only the set of known objects, not all possible objects. Code selection then depends on the number of known objects as

[9]It is interesting to note recent results [20] which suggest that recognition delays thought to indicate mental rotation may decrease with object familiarity, eventually leading to parallel consideration of stored views as proposed here.

well as the amount of redundancy necessary to provide tolerance to noise, occlusion, and other deformities. Thus the critical issue regarding qualitative representation is *representational capacity*, meaning the number of objects distinguishable in the presence of noise and occlusion. Assuming capacity is large compared to the size of the known object set, good discrimination can reasonably be expected from indexing. In Chapter 3 an upper bound on capacity as a function of a representation's constituent features is derived.

In Chapter 4 recognition of the "large" object set pictured in Figure 1-4 is considered. These objects are taken from a mechanical drawing textbook [24, page 150] where they serve as part of an exercise in sketching. Although sixteen objects is not very many in human terms, it is significantly more than usually considered in the computer vision literature, where sets consisting of single objects [65], or at most a few simple ones[37] are the rule. For example, the 3D objects considered by Huttenlocher consist of a cube, a five-sided wedge, and the union of a rectangular block and a wedge. In contrast, the objects in Figure 1-4 are complicated in several ways: they have curved surfaces, holes, and significant limbs (depending on viewpoint). There is, however, nothing special about these objects; that is to say, there is nothing obvious that allows them to be easily distinguished[10]. Thus they serve as a worthy test for qualitative recognition: if similar objects can be distinguished, then success on much larger sets involving substantially different objects can be anticipated.

In particular, identification as well as orientation estimation are considered, requiring a representational capacity large enough not only to distinguish objects, but also to distinguish the views collectively representing each object. Based on the result derived in Chapter 3, it is shown that simple features, i.e. isolated curves classified by concavity count, do not yield the required capacity. Instead, compound features consisting of multiple curves encoding view structure must be used, resulting in a new interpretation for perceptual grouping.

The resulting algorithm is so simple as to allow assignment of each stored view

[10]They were chosen in part because they are representable by the Symbolics S-Geometry geometric modelling package [64] used here to generate arbitrary object views.

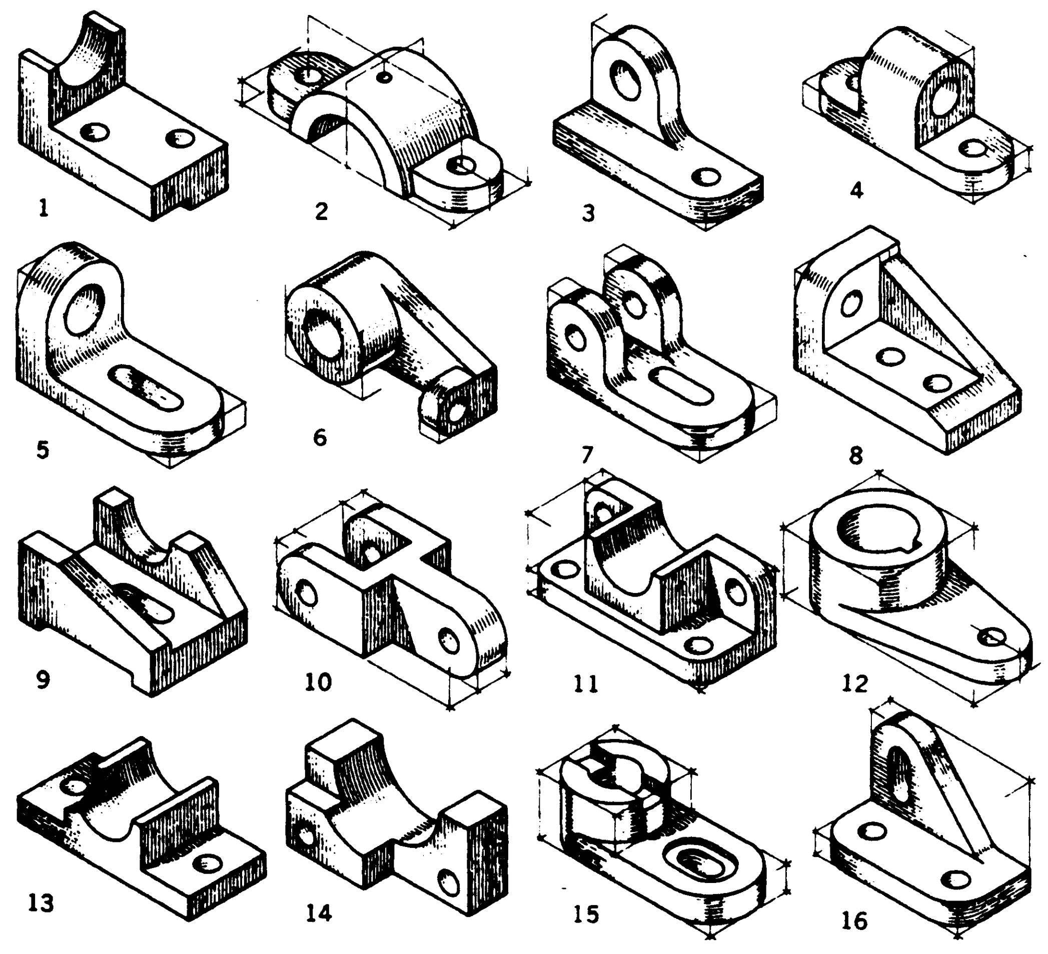


Figure 1-4: Object library considered here.

to a Connection Machine processor, resulting in parallel (constant time) recognition of known objects. Recognition results are presented for several one- and two-object scenes. Not surprisingly, the use of coarse information does not usually result in unique identification. For isolated objects, the large set of stored views of known objects is typically reduced to one much smaller and containing the correct object as well as similar views of others. This is acceptable, since indexing is fast and effective enough to allow application of more costly procedures yielding unique identification to the remaining views. When multiple objects are present, accurate object segmentation is required to avoid excessive ambiguity due to segmentation error since increased distortion lowers representational capacity.

The work is concluded in Chapter 5, where the central result is noted to be the upper bound on representational capacity derived in Chapter 3. Since this result does not restrict acceptable features to those associated with line drawings, the conclusions presented regarding the application of qualitative representation to 3D recognition are potentially quite general. Possible extensions include representations encoding qualitative surface shape, color, or texture, in addition to edge shape.

Chapter 2

Extraction of Qualitative Information from Digitized Line Drawings

In the previous chapter, qualitative representation for recognition of 3D objects was suggested. Here extraction of qualitative information from digitized (spatially quantized) line drawings is considered, with primary effort directed toward curvature characterization.

Encoding curvature in view descriptions is attractive due to its invariance with respect to scale and rotation in the image plane. But spatial quantization results in tangent orientation ambiguity and subsequent loss of curvature information. Coarse curvature characterization recoverable from the orientation information surviving digitization is required. Concavity count of digitized curves is one possibility, since it can often be extracted in spite of orientation ambiguity.

Following a discussion of previous line drawing research, the orientation problem is discussed in detail, leading to an algorithm for extraction of concavity count. The chapter concludes with comments regarding application to line drawings.

2.1 Previous Work: Line Drawing Analysis

Line drawings have enjoyed significant attention in the computer vision literature since the publication of Roberts' paper [56] in 1965. This work, discussed in more detail in the previous chapter, considered recognition of line drawings possibly extracted from imagery. Because geometric models were matched to the drawings, location and orientation of identified objects were estimated, resulting in reconstruction of 3D scenes.

Other attempts at 3D interpretation of line drawings followed, including the work of Guzman [31], Clowes [14], Huffman [36], and Waltz [70]. Guzman attempted to segment line drawings into separate objects without using object models, instead relying on heuristics to group line drawing regions. Because of dependence on heuristics instead of mathematical rigor, Guzman's algorithm fails on relatively simple drawings [58]. In an attempt to introduce rigor, Clowes and Huffman, in independent research, considered constraints on the vertices of trihedral polyhedra resulting in a well-known theory for labeling the concavity [1] of polyhedral edges in line drawings. Waltz [70] extended this work to a larger class of drawings, including those with lines due to shadows.

Heuristic attempts at interpretation of scenes containing curved objects followed [10, 44]. But in rigorously generalizing edge labeling to objects with C^3 surfaces, Malik [46] perhaps extends these ideas to their limit.

Simply stated, the principle behind line drawing labeling techniques is propagation of constraints itemizing possible configurations of convex and concave edges at junctions. Constraint propagation among vertices requires uniform convexity (or concavity) along each edge. However, it is possible for curved objects to exhibit edges whose convexity changes, for example the corner joints of many picture frames. Thus it appears that edge labeling algorithms, although of theoretical interest, are quite limited in their potential for practical 3D interpretation.

Although it has been suggested that 3D interpretation must precede line drawing

[1] "Concavity" here refers to the angle between polyhedral faces forming edges.

recognition [62], the view taken here is that prior interpretation is not required. Instead 2D descriptions are relied upon, avoiding the rather strict surface limitations imposed by constraint propagation, thus allowing consideration of a much larger class of objects. But 2D descriptions must be such that finite-view representations of 3D objects result, accomplished here by encoding qualitative information such as curve concavity count. In order to extract concavity count, evaluation of tangent orientation must first be addressed.

2.2 Orientation Bounds of Digitized Curves

2.2.1 Digitization Model

Extraction of tangent orientation along plane curves is required for evaluation of curvature, which defines shape, an obviously important visual attribute. Unfortunately the predominance of imaging devices (both biological and man-made) with spatially discrete sensors results in digitization of almost all curves of interest. Orientation estimation, or for that matter extraction of any curve information, then requires modelling of the digitization process.

The traditional model is that curves are digitized via uniform spatial sampling, allowing coordinates of activated grid elements (pixels) to be interpreted as if located on underlying continuous curves. Various curve fitting techniques for characterizing digitized curves such as spline methods [58] are based on this assumption. This model is of course is not accurate, as uniform sampling of a region containing a zero-area curve will most likely result in nothing. Compensation is achieved via techniques not requiring interpolants to intersect pixel coordinates, such as B-spline methods [2, 58] and methods involving least-squares fitting of parameterized curve models [58]. When direct calculation of tangent orientation or curvature is desired, numeric differentiation [18] is also used.

An alternative model implies that a grid element belongs to a digitized curve only if intersected by the original continuous curve, in agreement with standard graphics

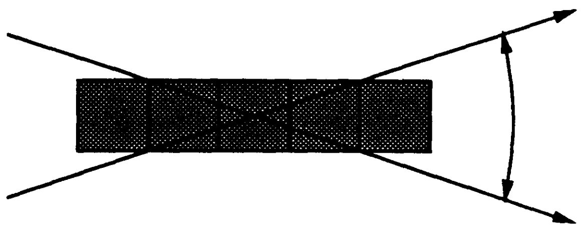


Figure 2-1: A five pixel configuration. The two rays illustrate the orientation range of lines intersecting all pixels.

algorithms for digitizing known curves [21]. Not surprisingly, its implications are quite different from those of the sampling model. Consider for a moment the problem of extracting tangent orientation along a continuous curve after digitization. Since orientation at a given point on the continuous curve is obtained by performing a limiting operation on linear approximations within point neighborhoods, we might analogously consider pixel neighborhoods along the corresponding digital curve. In the case of a square grid, the natural neighborhood shape is $n \times n$ square, with n odd to center the neighborhood on pixels of interest. Now within a given neighborhood there will be a configuration of pixels intersected by the original continuous curve. If n is much smaller than the radius of curvature of this curve, linear approximation is valid in the neighborhood. The resulting line defines orientation: the problem is to deduce it from the activated neighborhood pixels.

But consider the pixel configuration in Figure 2-1, which could correspond to the pixels from some digitized curve within a 5×5 neighborhood. The two rays indicate the orientation range of lines intersecting (and therefore activating) the pixels. There is no way to distinguish among these orientations. All are equally valid in the sense that each corresponds to a line accounting for the activated pixels. Thus in discussing orientation of digital curves, we must speak of intervals instead of single values. This uncertainty is a direct result of loss of information due to digitization, a phenomenon not directly accounted for by models interpreting pixel coordinates as curve samples.

Thus we have another hint at the plausibility of qualitative view descriptions. Digitization loses orientation information, the resulting uncertainty manifested in bounds on orientation instead of precise values. Curvature in the standard sense can not be determined unless injection of additional information, for example parametric curve models, takes place. Instead curvature must be characterized coarsely as discussed later. But first a procedure for calculating the orientation bounds of arbitrary pixel configurations is derived.

2.2.2 Orientation Range Calculation

To evaluate the orientation range for a given pixel configuration, all straight lines traversing the configuration must be determined. Thus the problem is to determine a set of necessary and sufficient conditions for straight line intersection of each pixel that is convenient for calculation of orientation bounds. The approach taken here is to define the necessary pixel order for straight line traversal of an arbitrary configuration, and the necessary and sufficient conditions for straight line intersection of consecutive pixels, resulting in a set of necessary and sufficient conditions for traversal of the entire configuration.

For the discussion which follows, pixels are defined as the *interiors* of the closed square regions covering the xy plane in Figure 2-2, each identified by the coordinates of its lower left hand corner. Furthermore, configurations of interest are assumed to consist of finite numbers of 8-connected pixels. Since pixels are disjoint, the order in which the pixels of a linear configuration (i.e. a configuration traversed by at least one straight line) are visited by a given traversing line is unique modulo traversal direction. Thus, a linear configuration can be represented as a sequence $S = \{p_1, \dots, p_N\}$, where $p_i = (x_i, y_i)$ is the ith pixel defined for a given traversing line, and $p_i = p_j$ iff $i = j$ due to pixel convexity. Next the ordering of pixels in S will be determined for all traversing lines.

In order to establish the traversal order through a linear configuration, first note that every subset of the configuration must also be linearly traversable, including those defined by the 3×3 neighborhoods centered on each pixel. Because the interior of the 3×3 square enclosing each neighborhood is convex, the pixels within such neighborhoods form contiguous subsequences of S. And since configurations of interest are assumed 8-connected, the pixels in S occurring immediately before and

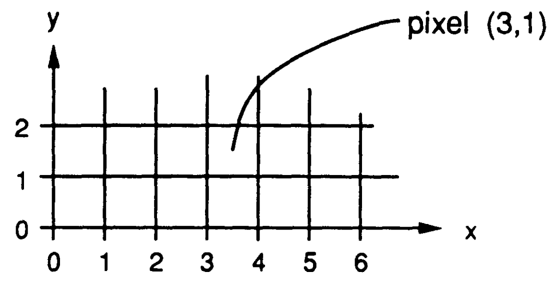


Figure 2-2: Pixels are defined here as the interiors of the regions delineated by a square grid, each identified by the coordinates of its lower left-hand corner.

after the center pixel of such a neighborhood must lie in the neighborhood. Thus, the traversal order through each neighborhood defines the order for the entire configuration. Clearly, the problem of ordering arbitrary linear configurations reduces to that for all linear 3×3 neighborhoods.

At this point it is unknown whether the pixel configuration inside a given 3×3 neighborhood is linear. But there is a rather strong constraint that can immediately be used to substantially prune the set of 256 possible neighborhoods (with center pixel activated). Consider the diagonal pixel pairs shown in Figure 2-3. Any straight-line intersecting both pixels in Figure 2-3(a) must have positive slope. Similarly, the pair in Figure 2-3(b) requires negative slope. Thus at most one of the pairs in Figure 2-3 can belong to a given linear configuration. Every 3×3 neighborhood containing at most one of these pairs is tabulated in Figure 2-4. Although it will later be shown that the diagonal pair constraint is not only necessary, but also sufficient for 3×3 neighborhood linearity, all that is assumed now is that the linear 3×3 neighborhoods are a subset of those in Figure 2-4.

To determine the pixel order implied by linear traversal of 3×3 neighborhoods, observe from Figure 2-4 that such neighborhoods have at most two pixel clusters in their outer layers, containing at most two pixels each. If it can be shown that the center pixel of those neighborhoods containing two outer layer clusters must lie between the two clusters in S, then ordering 3×3 neighborhoods can be reduced to ordering the pixels in outer-layer clusters. Although intuitively obvious, a rigorous

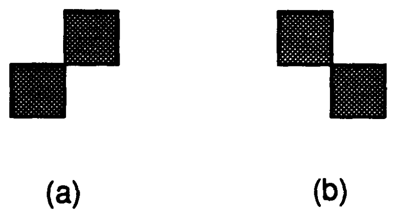


Figure 2-3: Inconsistent pixel pairs: (a) implies positive slope, whereas (b) implies negative slope.

proof follows.

First consider the case where a neighborhood has an outer-layer cluster consisting of a single pixel, as in Figure 2-5. Figure 2-5(a) and its three 90 degree rotations represents those cases where the outer-layer pixel (marked "a") has the same x or y value as the center pixel, and Figure 2-5(b) and its rotations represents those where the outer-layer pixel (marked "b") occupies a corner of the neighborhood. In both figures, the numbered pixel locations represent the possible locations of pixels in the other outer-layer cluster, as determined from Figure 2-4.

Consider Figure 2-5(a) first, assuming without loss of generality that the center pixel c follows the outer pixel a in S. If pixel *1* is activated, it cannot occur before a because then all pixels following a would have x-values less than that of c. Assuming it follows a, pixel *1* cannot precede c, because all pixels following it then would have y-values larger than that of c. Thus pixel *1*, if present, must follow c. Using similar arguments, the same conclusion can be drawn for pixels *2*, *4*, and *5*. If pixel *3* is present then it cannot precede a because then all pixels following a would have x-values less than that of c. Assuming it follows a, it cannot precede c, because all pixels following it must then have x-values greater than that of c. Thus all numbered pixels in Figure 2-5(a) must follow c, so in this case the center pixel must lie in S between the two outer-layer pixel clusters. The same conclusion can be similarly drawn for the 90 degree rotations of Figure 2-5(a).

Now consider Figure 2-5(b), also assuming that the center pixel c follows the outer pixel b in S. Pixels *6* and *7* must follow c for the same reasons as given for pixel *1*

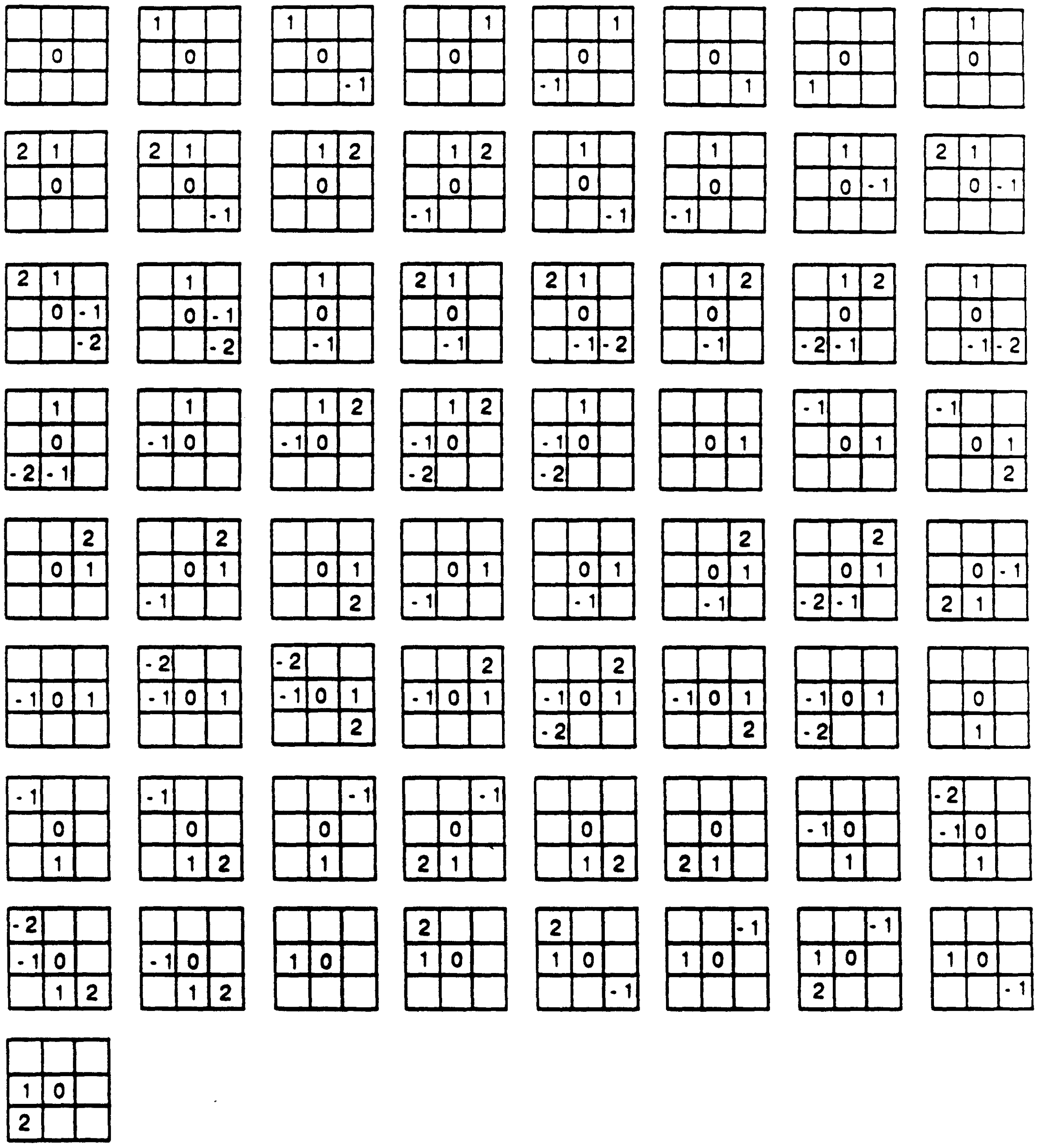


Figure 2-4: 3 × 3 pixel neighborhoods satisfying the diagonal pair constraint. The numbers occupying pixel locations indicate the ordering implied by linear traversal.

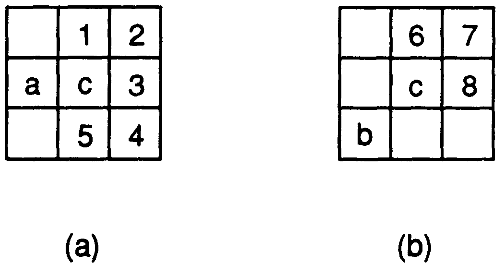


Figure 2-5: 3 × 3 neighborhoods containing two outer layer clusters, one of which contains a single pixel denoted by *a* or *b*. The numbered locations in each case indicate the possible locations of pixels belonging to the other cluster.

above, replacing pixel *a* with *b* in the argument. And, using the argument given in discussing pixel *3*, we can conclude the same for pixel *8*. Thus, here also the center pixel must reside between the outer-layer clusters in S, the same being true for the 90 degree rotations of Figure 2-5(b).

To complete the proof, consider the case where at least one of the outer-layer clusters has two pixels, as represented by Figure 2-6, its reflection about the center pixel x-value, and their six 90 degree rotations. The numbered pixel locations again represent the possible locations of pixels in the other outer-layer cluster as determined from Figure 2-4. From the analysis given for Figure 2-5(a), we can immediately conclude that the center pixel *c* lies between pixel *a* and the other cluster. The same can be concluded for pixel *b* from the analysis given for Figure 2-5(b). Clearly *a* and *b* must lie on the same side of *c* in S, completing the proof that the center pixel of a linear 3 × 3 neighborhood must reside in S between the outer-layer pixel clusters.

Thus the order in S of the pixels in a 3×3 neighborhood is determined by the order of the pixels in the outer layer clusters. Consider the ordering of two-pixel clusters as in Figure 2-6 since the single pixel case is trivial. Assuming that the center pixel follows pixels *a* and *b* in S, we can conclude immediately that *a* cannot precede *b*, because then all pixels following *b* would have y-values less than that of *c*. Similar reasoning holds not only for Figure 2-6, but for its reflection about the center pixel x-value, and their six 90 degree rotations. Thus for two-pixel clusters, the pixel with

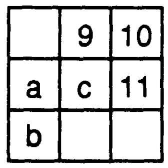


Figure 2-6: A 3×3 neighborhood containing two outer layer clusters, one of which contains two pixels, here denoted by a and b. The numbered locations indicate the possible locations of pixels belonging to the other cluster.

the same x- or y-value as the center pixel must lie between the corner pixel and center pixel in S.

We now have two rules for ordering the pixels in 3×3 linear neighborhoods:

1. The center pixel is ordered between the outer-layer pixel clusters.
2. In two-pixel outer-layer clusters the pixel with the same x- or y-value as the center pixel is ordered between the corner pixel and center pixel.

Using these rules we can derive the pixel order required by linear traversal of any 3×3 neighborhood. The results are presented in Figure 2-4, where center pixels are given index 0 to enable direction reversal by simply negating indices.

Having established the required pixel order for linear traversal of any linear 3×3 neighborhood, ordering of an arbitrary linear configuration can be performed by tiling the neighborhoods in Figure 2-4. But a much simpler approach is to use the two rules derived above directly. Consider for example the pixel configuration in Figure 2-7(a). We begin by finding the two end-pixels, which are the only pixels whose 3×3 neighborhoods contain only one outer layer cluster. These pixels are labelled in Figure 2-7(a) assuming that the traversal direction is from left to right. Next, the pixels inside the 3×3 neighborhood centered on the first pixel are labelled using rule 2. We consider next the 3×3 neighborhood of pixel 3 in Figure 2-7(c), where pixel 4 is trivially labelled using rule 1. Next the neighborhood of pixel 4 is considered,

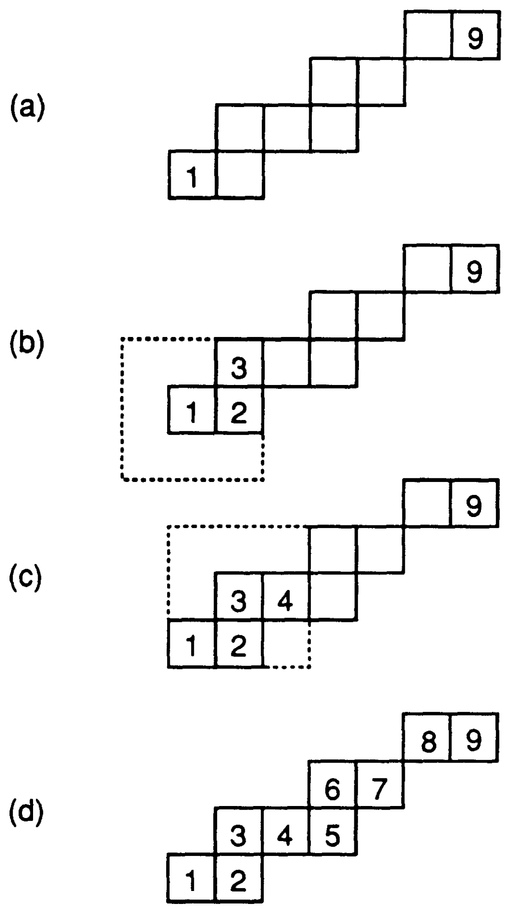


Figure 2-7: The procedure for determining the pixel order implied by linear traversal. (a) First label the end-pixels to establish a traversal direction. (b) Then label pixels 2 and 3 by applying rule 2 to the 3×3 neighborhood of pixel 1. (c) Next label pixel 4 by considering the neighborhood of pixel 3. (d) Repeat until pixels are exhausted.

yielding pixels 5 and 6, and so on, until all of the pixels are labelled. The final result is shown in Figure 2-7(d).

Because arbitrary linear pixel configurations can now be ordered, we can proceed with the orientation question. As mentioned above, the goal is to determine a set of necessary and sufficient conditions for straight-line intersection of each pixel in a given configuration which is convenient for orientation calculation. One approach is to form a collection of pixel pairs containing all pixels from the configuration of interest, for each pair writing a set of necessary and sufficient conditions for straight-line intersection. Simultaneous consideration of these conditions results in the set of traversing lines. Since a linear configuration can be represented as a pixel se-

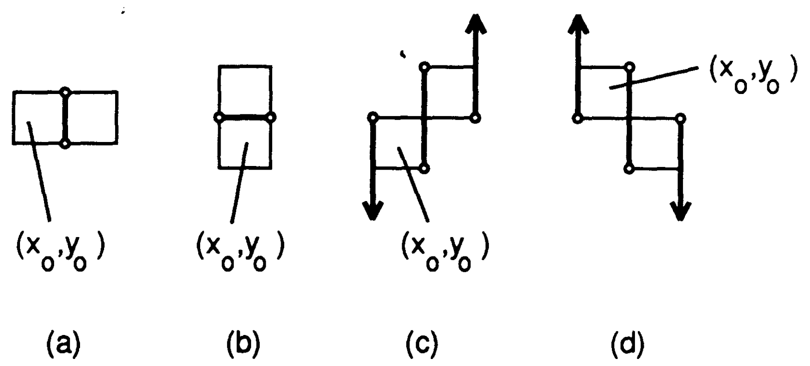


Figure 2-8: Possible pairs of 8-connected pixels. The darkened segments represent necessary and sufficient conditions for linear traversal.

quence $S = \{p_1, \ldots, p_N\}$, where $p_i = p_j$ iff $i = j$, collections with the minimum number of pairs can be simply constructed as follows. If N is even, then the pair collection is $\{(p_1, p_2), (p_3, p_4), \ldots, (p_{N-1}, p_N)\}$. If N is odd, then the collection is $\{(p_1, p_2), (p_3, p_4), \ldots, (p_{N-2}, p_{N-1}), (p_{N-1}, p_N)\}$. These minimal collections have an additional advantage, namely that the pixel pairs are 8-connected, reducing the number of cases that need be considered to the four in Figure 2-8.

To determine the necessary and sufficient conditions for linear traversal of these pairs, assume for the moment that $x_1 \neq x_N$. Because the x-intervals in p_1 and p_N then are disjoint, no vertical line can traverse the configuration, allowing us to represent traversing lines in the standard form $y = mx + b$, where m is the slope and b is the y-intercept. The strategy is to obtain bounds on m from the pixel pairs, leading to bounds on orientation.

Consider first the case where paired pixels have identical y-values, as in Figure 2-8(a). Clearly, any line intersecting pixel (x_o, y_o) and ray $x = x_o + 1, y \geq y_o + 1$ has positive slope, and thus is characterized by $y > y_o + 1$ for $x > x_o + 1$, eliminating the possibility of intersection with pixel $(x_o + 1, y_o)$. Similarly, any line intersecting pixel (x_o, y_o) and ray $x = x_o + 1, y \leq y_o$ is characterized by $y < y_o$ for $x > x_o + 1$, also ruling out intersection with pixel $(x_o + 1, y_o)$. Therefore, any line traversing both pixels must intersect the indicated open segment of $x = x_o + 1$. Since any non-vertical line intersecting this segment must intersect both pixels, the following is a necessary and sufficient condition for traversal of the pair in Figure 2-8(a) by $y = mx + b$:

$$y_o < (x_o + 1)m + b < y_o + 1 \tag{2.1}$$

The above suggests that if possible the conditions for linear traversal of the other three pixel pairs should also be expressed in terms of linear inequalities in m and b, allowing the bounds on slope m for any linear configuration to be obtained via solution of two linear programs, one minimizing cost m, the other minimizing cost $-m$. With this in mind, consider the pixel pair in Figure 2-8(b). Using arguments similar to those for the preceding pair, we can conclude that it is necessary and sufficient for any traversing line ($m \neq 0$) to intersect the indicated open line segment, or equivalently,

$$x_o < [(y_o + 1) - b]/m < x_o + 1 \tag{2.2}$$

In order to convert to a form linear in m and b, the sign of m must be determined. This can be done easily because if the configuration from which the pair was drawn is linear, $y_1 \neq y_N$, so the necessary sign of m is the same as the sign of $(y_N - y_1)/(x_N - x_1)$ since we have assumed $x_1 \neq x_N$. Thus, if $(y_N - y_1)/(x_N - x_1) > 0$,

$$x_o m + b < y_o + 1 < (x_o + 1)m + b \tag{2.3}$$

and if $(y_N - y_1)/(x_N - x_1) < 0$,

$$(x_o + 1)m + b < y_o + 1 < x_o m + b \tag{2.4}$$

To prove sufficiency, note that inequality 2.3 implies both $m > 0$ and inequality 2.2. Similarly, inequality 2.4 implies both $m < 0$ and inequality 2.2.

Proceeding to Figure 2-8(c), note immediately that any line traversing this pair must have positive slope. Since any line intersecting either

1. ray $x = x_o, y \geq y_o + 1$ and pixel (x_o, y_o), or

2. ray $x = x_o + 1, y \geq y_o + 2$ and pixel $(x_o + 1, y_o + 1)$, or

3. ray $x = x_o + 1, y \leq y_o$ and pixel (x_o, y_o), or

4. ray $x = x_o + 2, y \leq y_o + 1$ and pixel $(x_o + 1, y_o + 1)$

has negative slope, we have the following necessary conditions for traversal by line $y = mx + b$:

$$\begin{gathered} x_o m + b < y_o + 1 \\ y_o < (x_o + 1)m + b < y_o + 2 \\ y_o + 1 < (x_o + 2)m + b \end{gathered} \tag{2.5}$$

To prove sufficiency, consider three cases:

1. $(x_o + 1)m + b = y_o + 1$

2. $y_o < (x_o + 1)m + b < y_o + 1$

3. $y_o + 1 < (x_o + 1)m + b < y_o + 2$

Since the first and third expressions in 2.5 imply that $m > 0$, we see immediately that both pixels are traversed if $(x_o + 1)m + b = y_o + 1$. If $y_o < (x_o + 1)m + b < y_o + 1$, $y = mx + b$ obviously intersects pixel (x_o, y_o). Letting x^* denote the value of x where $y = mx + b$ intersects $y = y_o + 1$, and combining the third inequality in 2.5 with $(x_o + 1)m + b < y_o + 1$, results in $(x_o + 1)m + b < x^*m + b < (x_o + 2)m + b$, implying that $x_o + 1 < x^* < x_o + 2$. Since $m \neq 0$, we can conclude that pixel $(x_o + 1, y_o + 1)$ is traversed as well. Finally consider the case where $y_o + 1 < (x_o + 1)m + b < y_o + 2$. Immediately we see that $y = mx + b$ intersects pixel $(x_o + 1, y_o + 1)$. Defining x^* in the same fashion, and combining the first inequality in 2.5 with $y_o + 1 < (x_o + 1)m + b$, results in $x_o m + b < x^*m + b < (x_o + 1)m + b$, implying that $x_o < x^* < x_o + 1$. Since this implies intersection of pixel (x_o, y_o), we conclude that 2.5 defines a set of sufficient as well as necessary conditions for traversal of the two pixels in Figure 2-8(c).

The necessary and sufficient conditions for the pixel pair in Figure 2-8(d) are obtained in a similar fashion. They are:

$$\begin{gathered} y_o < x_o m + b \\ y_o - 1 < (x_o + 1)m + b < y_o + 1 \\ (x_o + 2)m + b < y_o \end{gathered} \tag{2.6}$$

Having established the conditions for linear traversal of the pixel pairs in Figure 2-8 in terms of linear inequalities in m and b, the slope bounds of an arbitrary linear pixel configuration can be obtained by solving two linear programs [61, 66], one each to minimize and maximize m. Consider for example the configuration in Figure 2-7, which has already been ordered. Decomposing the pixel sequence as discussed previously results in the pair collection $\{(p_1, p_2), (p_3, p_4), (p_5, p_6), (p_7, p_8), (p_8, p_9)\}$. The necessary and sufficient conditions for linear traversal of the configuration are obtained by collecting the conditions for individual pairs given in inequalities 2.1, 2.3, 2.4, 2.5, and 2.6. Assuming without loss of generality that pixel 1 has coordinates $(0, 0)$, we have:

$$\begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 2 & 1 \\ -2 & -1 \\ -3 & -1 \\ 4 & 1 \\ -4 & -1 \\ 5 & 1 \\ -5 & -1 \\ 6 & 1 \\ -6 & -1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} > \begin{bmatrix} 0 \\ -1 \\ 1 \\ -2 \\ -2 \\ 2 \\ -3 \\ 2 \\ -4 \\ 3 \\ -4 \end{bmatrix}$$

Combining this matrix inequality with cost functions m and $-m$ results in two linear programs, each of which can be solved by standard techniques such as the simplex method [61, 66] to respectively minimize and maximize m. The result is $2/5 < m < 3/4$.

Conversion of slope bounds to orientation bounds is straightforward, except if $x_1 > x_N$ and $y_1 = y_N$. Letting θ denote orientation angle, if $x_1 < x_N$ or $y_1 \neq y_N$ we have:

$$Tan^{-1}(x_N - x_1, m_{inf}(x_N - x_1)) < \theta < Tan^{-1}(x_N - x_1, m_{sup}(x_N - x_1)) \\ (x_1 < x_N \text{ or } y_1 \neq y_N) \tag{2.7}$$

where m_{inf} and m_{sup} denote respectively the infimum and supremum of possible slopes, and $Tan^{-1}(x, y)$ is the inverse tangent of y/x accounting for the sign of x and defined such that $-\pi < Tan^{-1}(x, y) \leq \pi$. The above expression maintains continuity of θ with respect to m because if $(x_N - x_1, m(x_N - x_1))$ sits on the branch cut of Tan^{-1} then $x_1 > x_N$ and $m = 0$, requiring $y_1 = y_N$. If $x_1 < x_N$ or $y_1 \neq y_N$, $(x_N - x_1, m(x_N - x_1))$ cannot cross the branch cut as m varies over the interval (m_{inf}, m_{sup}), thus there is no problem with continuity of the inverse tangent function. If on the contrary $x_1 > x_N$ and $y_1 = y_N$, the branch cut is crossed since $0 \in (m_{inf}, m_{sup})$ when $y_1 = y_N$. This is a result of the requirement that all pixels have identical y-values in a linear configuration where $y_1 = y_N$, which can be verified by assuming the contrary, i.e. that there is a pixel $p_m (1 < m < N)$ such that $y_m \neq y_N$. If $y_m < y_N$, then any line intersecting p_1 and p_m implies $y_n \leq y_m < y_N$ for $n > m$, prohibiting intersection with p_N. Similarly, $y_m > y_N$ implies that $y_n \geq y_m > y_N$ for $n > m$. Thus, if a given configuration is linear, and $y_1 = y_N$, then all pixels share the same y-value and are traversable by horizontal straight lines ($m = 0$). Therefore, if $x_1 > x_N$ and $y_1 = y_N$, $(x_N - x_1, m(x_N - x_1))$ crosses the branch cut of Tan^{-1}, requiring corrective addition of 2π to the expression defining the supremum of θ to correct for the discontinuity in Tan^{-1}:

$$Tan^{-1}(x_N - x_1, m_{inf}(x_N - x_1)) < \theta < Tan^{-1}(x_N - x_1, m_{sup}(x_N - x_1)) + 2\pi \\ (x_1 > x_N \text{ and } y_1 = y_N) \tag{2.8}$$

Concluding the example based on the pixel configuration in Figure 2-7, recall that the traversal direction there was arbitrarily chosen so that $x_1 < x_N$. Applying 2.7, we get $.3805 < \theta < .6435$, where θ is measured in radians.

So far, we have not considered the orientation bounds of linear configurations having $x_1 = x_N$. Since the pixels in such a configuration must have identical x-values, the bounds can be easily calculated by rotating the bounds $(\theta'_{inf}, \theta'_{sup})$ of a horizontal configuration having the same number of pixels, ordered such that x-value increases with the pixel index. Then if $y_1 < y_N$,

$$\theta'_{inf} + \pi/2 < \theta < \theta'_{sup} + \pi/2 \tag{2.9}$$

and if $y_1 > y_N$,

$$\theta'_{inf} - \pi/2 < \theta < \theta'_{sup} - \pi/2 \tag{2.10}$$

Before presenting an application, let us pause to summarize the procedure for calculating orientation bounds discussed above. Given an arbitrary 8-connected pixel cluster, the first step is to determine whether each of the 3×3 neighborhoods centered on configuration pixels belongs to the set presented in Figure 2-4. This can be determined simply by checking against the presence of both diagonal pairs in Figure 2-3. If not all neighborhoods satisfy the diagonal pair constraint, then the configuration can't possibly be linear. If all neighborhoods satisfy the constraint, the next step is to order the configuration pixels based on the neighborhood pixel ordering implied by linear traversal. If there are N pixels in the configuration, and $x_1 = x_N$, then the configuration is linear iff all pixels have identical x-values. In this case, the orientation bounds can be obtained using inequalities 2.9 or 2.10. If $x_1 \neq x_N$, then the necessary and sufficient conditions for traversal of pairs of consecutive pixels, as defined in inequalities 2.1, 2.3, 2.4, 2.5, and 2.6, are collected. Together these form a set of necessary and sufficient conditions for traversal of the entire configuration. Since the conditions are in the form of linear inequalities in slope and y-intercept, the extrema of the slope can be obtained by solving two linear programs via an efficient algorithm such as the simplex method[61, 66]. If there is no line which satisfies the conditions, the conclusion is of course that the configuration is not linear.

And one final note before proceeding. The 3×3 pixel neighborhoods in Figure 2-4 represent all neighborhoods not including both of the diagonal pairs in Figure 2-3. Thus, if the configuration within a given 3×3 neighborhood is linear, it must belong to Figure 2-4. What is not immediately obvious is whether each neighborhood in Figure 2-4 is in fact linear. Using the above approach for determining orientation ranges, this can be verified. Therefore, we can conclude that a 3×3 neighborhood is linear if and only if both of the pixel pairs in Figure 2-3 are not present.

2.3 Curvature Description of Digitized Curves

The calculation of curvature along a continuous curve requires evaluation of tangent orientation at each point. This is done by considering the orientation of linear curve approximations in neighborhoods of points of interest, taking the limit as the neighborhood size approaches zero. For digitized curves we are confined to neighborhoods of pixels, the natural neighborhood shape for a square grid as in Figure 2-2 being square with odd side length to center the neighborhood on pixels of interest. But in this case, instead of a single orientation associated with each neighborhood, there is an interval corresponding to the lines traversing the pixels inside. Since this interval grows as neighborhood size is decreased, a different kind of limiting scheme is required.

A quite reasonable approach under the circumstances is to start with an $M \times M$ neighborhood and reduce its size until the pixel configuration inside is linear, using the resulting orientation interval in evaluating curvature. An advantage of this approach is that the orientation bounds of linear neighborhoods size M and below can be computed in advance (using the procedure just discussed) and retrieved when needed[2], resulting in fast calculation of curvature descriptions including number of concavities

[2]Walters [69] proposes representing line drawings in ρ-space, which consists of three dimensions, two corresponding to the image plane, the other indicating tangent orientation. Because orientation intervals can be expressed in ρ-space, we might consider the stated orientation evaluation technique as means of generating this representation. But orientation information would then be available only at pixels with linear neighborhoods, defeating one of the important features of the ρ-space representation, namely the ability to represent multiple orientations found at nonlinear locations such as line intersections.

and bounds on total curvature[3].

We shall consider digitized curves derived from continuous curves having radii of curvature much larger than the pixel size. This implies that the continuous curves are approximately linear within 3×3 neighborhoods, yielding digitized curves containing in all likelihood a great majority of pixels with linear 3×3 neighborhoods. I shall refer to such pixels as *3-linear.* Since the ordering of pixels within linear 3×3 neighborhoods has been determined, a curve exclusively containing 3-linear pixels can be ordered in exactly the manner discussed for linear pixel configurations above. Curvature can then be assessed by calculating the orientation bounds at each pixel in accordance with this ordering. Thus we consider first those curves containing only 3-linear pixels, followed later by discussion of the more general case where nonlinear 3×3 neighborhoods are present.

The first step then toward describing curvature along a given 3-linear curve is to order the pixels. The next step is to calculate the orientation intervals along the curve in accordance with the established pixel ordering. We begin by finding the starting and ending orientation ranges. To obtain the starting range, we could consider neighborhoods of the first pixel. But these would not contain very many pixels, so to determine a more precise range we shall use the following approach. Recalling that the maximum neighborhood size to be considered is M, we find the first pixel in the curve whose $M \times M$ neighborhood does not contain the first pixel, and subtract one from its index. If the configuration within the resulting neighborhood is linear, the orientation range obtained with respect to the predetermined pixel ordering is the starting range. If the neighborhood is not linear, the neighborhood size is reduced by 2 until a linear neighborhood is found. The procedure is the same for the ending orientation range, except the index of the last neighborhood not containing the last pixel is incremented. In the discussion which follows, s and e denote respectively the indices of the pixels corresponding to the starting and ending orientations, and we assume that $s < e$ since $s \geq e$ implies the curve is linear.

[3]The total curvature of a given curve is the the total change in orientation noted as the curve is traced. For example, the total curvature of a circle is 2π.

The orientation ranges for the pixels between s and e are calculated as follows. Starting with neighborhood size M, the neighborhood at a given pixel is tested to determine if it is linear. If so, the orientation range obtained with respect to the ordering of the curve is associated with that pixel. If the neighborhood is not linear, the neighborhood size is reduced by two until linearity is achieved. Since all the pixels are assumed to have linear 3×3 neighborhoods, all pixels between s and e will have non-empty orientation ranges, resulting in a sequence of orientation intervals.

When considering curvature along continuous curves, the branch of the orientation function occasionally has to be changed to ensure continuity. Here we find ourselves in a similar situation: if the orientation bounds along the curve are to meaningfully describe the relative orientation between different pixels of the curve, compensation has to be made for the branch cut of the orientation function. This can be done by minimizing the distance between orientation intervals at consecutive pixels via translation by multiples of 2π. To be precise, let $(\theta(i)_{inf}, \theta(i)_{sup})$ and $(\theta(i+1)_{inf}, \theta(i+1)_{sup})$ denote respectively orientation ranges at pixels i and $i+1$. If the distance between the two intervals is zero, then no adjustment is necessary. If $\theta(i+1)_{sup} < \theta(i)_{inf}$, then $(\theta(i+1)_{inf}, \theta(i+1)_{sup})$ must be translated by 2π until the distance is minimized. Similarly, if $\theta(i+1)_{inf} > \theta(i)_{sup}$, then $(\theta(i+1)_{inf}, \theta(i+1)_{sup})$ must be translated by -2π until the distance between the two intervals is minimized. The orientation ranges must be treated in this way starting with $i = s$, and progressing along the orientation interval sequence until the ending interval is corrected.

Having compensated for the branch cut in the orientation function, we can immediately obtain bounds on the total curvature C of the digitized curve in question. Since the starting and ending orientation ranges are respectively $(\theta(s)_{inf}, \theta(s)_{sup})$ and $(\theta(e)_{inf}, \theta(e)_{sup})$, we simply have

$$\theta(e)_{inf} - \theta(s)_{sup} < C < \theta(e)_{sup} - \theta(s)_{inf} \tag{2.11}$$

To determine an estimate for number of concavities, note that concavity is signalled by disjoint orientation intervals (see Figure 2-9). The number of concavities

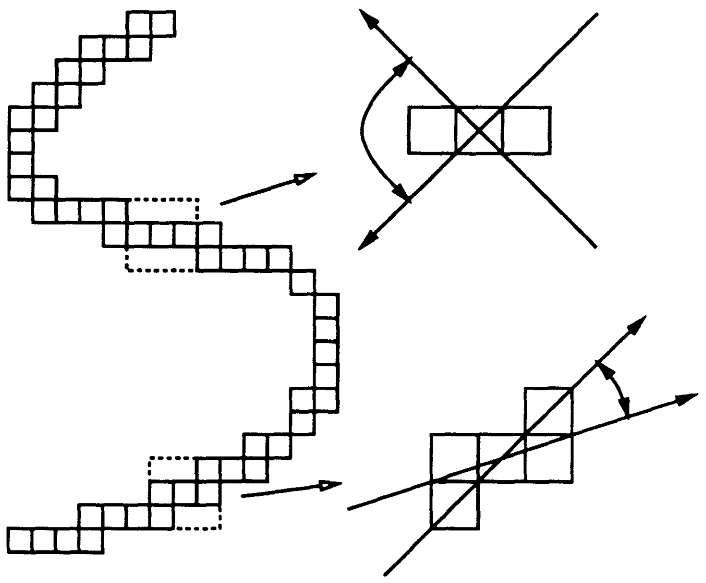


Figure 2-9: Disjoint orientation intervals signal concavity (here M=3, elsewhere M=9).

can thus be estimated by moving along the curve, determining if the orientation range of the current pixel is consistent with those noted previously. If not, concavity is detected. In more detail, let $R(i)$ denote the orientation range to be compared to the orientation interval at pixel $i + 1$, obtained by combining the orientation constraints at the previous pixels. Starting with $i = s$, let $R(i) = (\theta(s)_{inf}, \theta(s)_{sup})$. If $R(i) \cap (\theta(i+1)_{inf}, \theta(i+1)_{sup})$ is nonempty, set $R(i+1) = R(i) \cap (\theta(1+1)_{inf}, \theta(i+1)_{sup})$, and proceed to the next pixel.

If however the intersection is empty, then a concavity is detected. In this event, we record the detection in sequence D, which is initially empty. If $(\theta(i+1)_{inf}, \theta(i+1)_{sup}) > R(i)$, then a symbol representing positive curvature, say "+", is appended to D, but only if the previous symbol is not "+". If the previous symbol is "+", then there is no point to adding another, since the two symbols correspond to the same detected concavity. If $(\theta(1+1)_{inf}, \theta(i+1)_{sup}) < R(i)$, then a symbol representing negative curvature, say "–", is similarly appended to D. To continue, we reset $R(i+1) = (\theta(i+1)_{inf}, \theta(i+1)_{sup})$, and proceed to the next pixel.

After considering all orientation ranges in this way, from pixels s to e, the length of D represents the number of intervals along the curve where curvature of a given sign was found, and can be interpreted as the number of concavities detected[4]. We can coarsely describe the shape of a 3-linear curve with this *concavity count*, signed by the first symbol in D to indicate the curvature sign of the first concavity detected.

Concavity count is similar in flavor to the codon representation[34, 55], which is another qualitative shape description designed to provide initial indexing for recognition. Codons are intervals of curves bounded by curvature minima and classified by number of contained curvature zeroes. Curves are represented by lists of constituent codons. Concavity count is derivable from codon representations, however the converse is false since concavity count does not encode curvature extrema. Thus concavity count is less informative than representation via codons.

To this point nothing has been said concerning selection of maximum neighborhood size M. As M increases, lines traversing linear neighborhoods must intersect more pixels, yielding tighter orientation bounds and better concavity count estimates. However M can not be increased arbitrarily since the number of corresponding linear neighborhoods which must be stored grows with M. In order to explore the tradeoff between orientation ambiguity and number of stored neighborhoods, all linear neighborhoods of sizes 3, 5, 7, and 9 were tabulated and their orientation bounds calculated.

The major difficulty in doing so was the number of different pixel neighborhoods of given size m, namely 2^{m^2-1} (since we are interested only in neighborhoods whose center pixels are activated). As m increases, exhaustive search for linear neighborhoods becomes inappropriate. In order to avoid exhaustive search, pruning is required. Effective reduction is realized by recalling that all subsets of linear configurations must also be linear. Thus it is necessary to consider only size m configurations whose size $m-2$ neighborhoods are linear (see Figure 2-10), allowing prospective configurations of size m to be obtained by tiling linear neighborhoods of size $m-2$. Compilation of linear neighborhoods proceeds iteratively starting with m=5, the results for m=3 having been obtained previously (see Figure 2-4).

[4]Here the term "concavity" is independent of any notion of "inside" or "outside." Instead it refers to a curve interval of monotonic orientation variation, independent of the direction of variation.

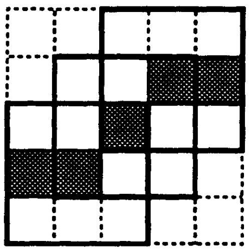


Figure 2-10: Subsets of linear neighborhoods must be linear. Thus all size 3 neighborhoods of the pictured size 5 linear neighborhood are linear, and can be found in Figure 2-4.

In order to evaluate the variation in orientation ambiguity with respect to m, see Figure 2-11 where the minimum and maximum orientation range sizes over all linear neighborhoods for each value of m are plotted. Note that orientation ambiguity decreases slowly in the vicinity of m=7 and 9, suggesting that large increases in m are required to significantly reduce uncertainty. But Figure 2-12 suggests that each increase in m results in an order-of-magnitude increase in number of linear neighborhoods. Thus close to m=7 and 9 it appears that small decreases in ambiguity are accompanied by huge increases in number of linear neighborhoods, implying that a reasonable compromise between ambiguity and linear neighborhood count is obtained by choosing M=7 or 9. Thus in all that follows M is set to 9.

Before proceeding to an example, we consider the mechanics of linear neighborhood retrieval. First note that only neighborhoods completely traversed are of interest, meaning those containing two outer-layer clusters. Thus only 48 size 3 neighborhoods need be stored even though 65 are pictured in Figure 2-4. Second, neighborhoods must be represented in a standard form for lookup. Canonical representations are obtained by simply translating neighborhoods so that center pixels have coordinates (0,0). Third, the number of stored neighborhoods can be reduced substantially by appealing to 8-way symmetry (90 degree rotations plus 90 degree rotations of reflection about x-axis). Thus the respective neighborhood counts for m=3, 5, 7, and 9 (which are all needed if $M = 9$) are reduced from 48, 816, 9968, and 105956 to 9, 111, 1272, and 13321. Finally there is the issue of parallel versus serial implementation. Since each neighborhood is an independent entity, each can be assigned to

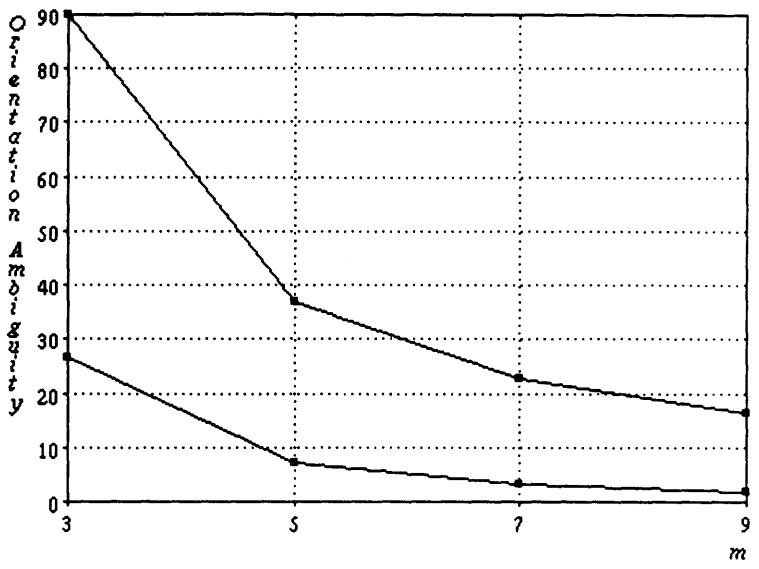


Figure 2-11: Bounds on orientation ambiguity (in degrees) as a function of neighborhood size m.

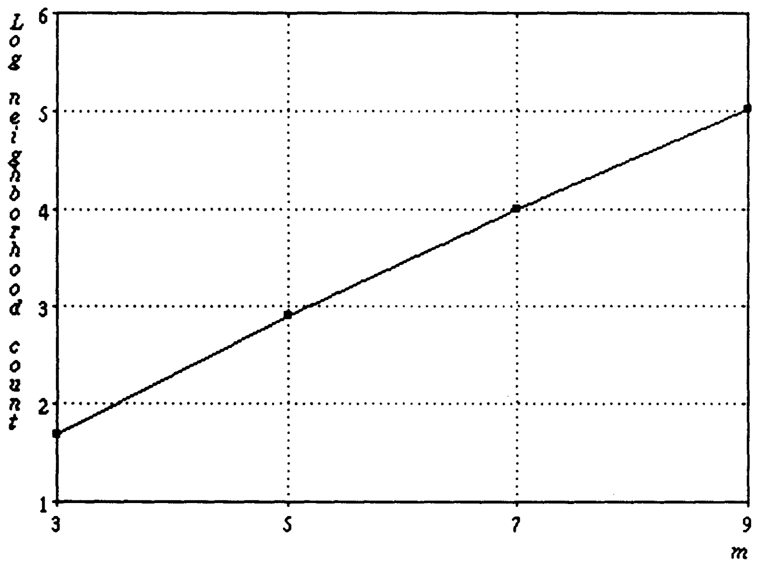


Figure 2-12: Logarithm (base 10) of number of linear neighborhoods as a function of neighborhood size m.

a different processor in a parallel computer, allowing simultaneous consideration of all stored neighborhoods. Alternatively, neighborhoods could be represented in tree structures (one for each neighborhood size) whose nodes represent pixel coordinates to facilitate fast retrieval on a serial machine. The latter was chosen since my parallel implementation using 4K processors of a Connection Machine model CM-2 parallel supercomputer configured for 16K virtual processors[5] resulted in performance about thirty times slower than tree search on a Symbolics model 3650 Lisp Machine.

To illustrate the performance of the shape description procedure discussed above we consider two curves, both pictured in Figure 2-13. In both cases the assumed direction of travel is from left to right. The first example involves curve (a), and demonstrates the effectiveness of the procedure in detecting subtle curvature. The calculated bounds on total curvature are $-.3070 < C < .1419$, but more importantly the signed concavity count is -2, indicating that two concavities were detected, the first having negative curvature. The second example involves curve (b), and demonstrates the ability of the procedure to classify a jagged collection of pixels as being linear, in spite of the curvature sensitivity demonstrated in the previous example. The procedure yields a concavity count of 0, which is correct since the curve was generated by a straight line.

To extend the above procedure to those curves containing pixels which are not 3-linear is trivial. In this case the curves are segmented into 3-linear segments bordered by clusters of pixels which are not 3-linear. All that needs to be done is to order the 3-linear segments consistently, and append the sequences of raw orientation ranges obtained for each segment accordingly to form one large sequence of ranges. This range sequence can then be handled exactly as the sequence obtained from a purely 3-linear curve would be, first by compensating for the orientation branch cut, and then by extracting the bounds on total curvature and the concavity description. In this way we completely ignore the clusters containing pixels which are not 3-linear, which is debatable since such clusters can signal corners. However, their ability to do so is clouded by many misses and false alarms, suggesting that if necessary other

[5]The number of virtual processors must be a power of 2.

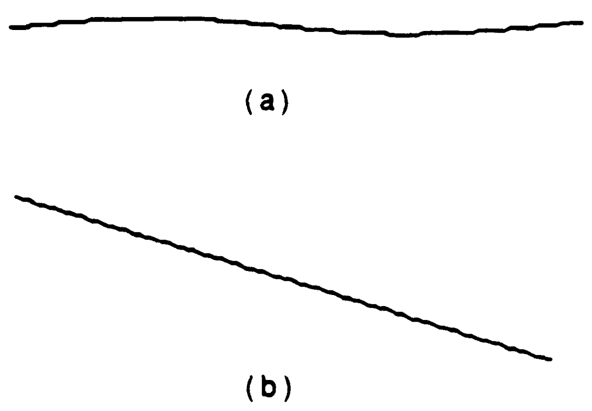


Figure 2-13: Examples considered in text.

means for detecting corners should be found.

2.4 Application to Line Drawings

2.4.1 Segmentation

Application of the above techniques to digitized line drawings as in Figure 2-14 requires segmentation into curves. This can be accomplished via curve tracing initiated at curve ends. In order to detect curve ends two situations must be addressed, namely isolated termination and termination at junctions.

Terminations at junctions of three or more digitized curves are easily detected since branching occurs there. Thus neighborhoods of junction pixels can not be ordered in a manner consistent with linear traversal. Since 3×3 linear neighborhoods result in consistent ordering, detection of junction pixels is performed by searching for nonlinear 3×3 neighborhoods. As discussed earlier, this reduces to detection of neighborhoods containing both diagonal pairs in Figure 2-3. Because junction pixels can appear in groups (see Figure 2-15), junction pixel clusters (8-connected)

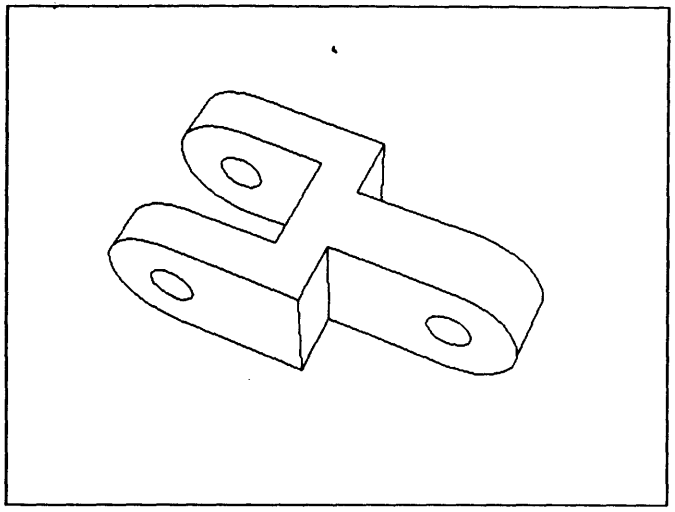


Figure 2-14: Sample digitized line drawing used to demonstrate extraction of curvature information.

are assumed to represent individual junctions[6].

The junctions found in Figure 2-14 are boxed in Figure 2-16. Note that all 3-junctions are found. Two 2-junctions (corners) are found as well (labelled "a" and "b"). But one is missed (labelled "c"), and one false corner is found (labelled "d"). This illustrates the difficulty in finding corners via nonlinear 3×3 neighborhoods as discussed above, and is consistent with the fact that corners are orientation discontinuities and are thus not preserved in a straightforward way by digitization.

Given a set of detected junctions represented by junction pixel clusters, the next step is to find the end pixels of intersecting lines. This is accomplished by finding those 3-linear pixels whose 3×3 neighborhoods suggest consecutive ordering with junction pixels (see Figure 2-15 where end pixels are labelled "1"). Detection of these pixels occurs quickly since only 3×3 neighborhoods of junction pixels need be searched.

Isolated termination pixels are found as discussed previously for the pixel configuration in Figure 2-7. Terminating pixels are signalled by linear 3×3 neighborhoods having single outer-layer clusters. The drawing in Figure 2-14 does not contain any isolated terminations.

[6]Note the contrast between multiple pixel junctions of digitized curves and single-point junctions of continuous curves.

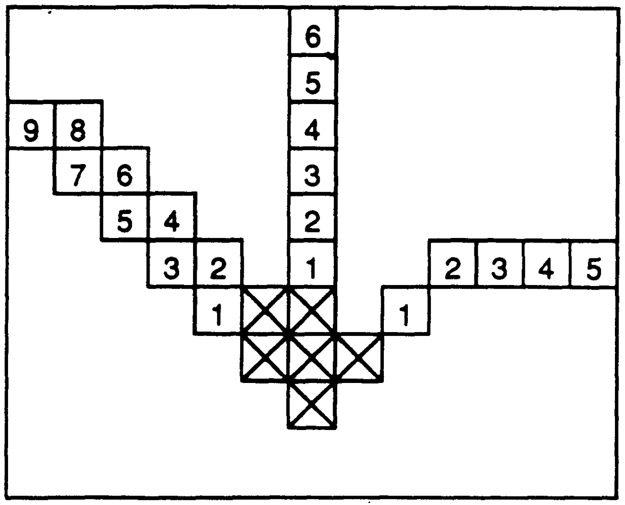


Figure 2-15: A magnified junction. Vertex pixels contain diagonals. Curve pixels are numbered assuming direction of travel is away from junction.

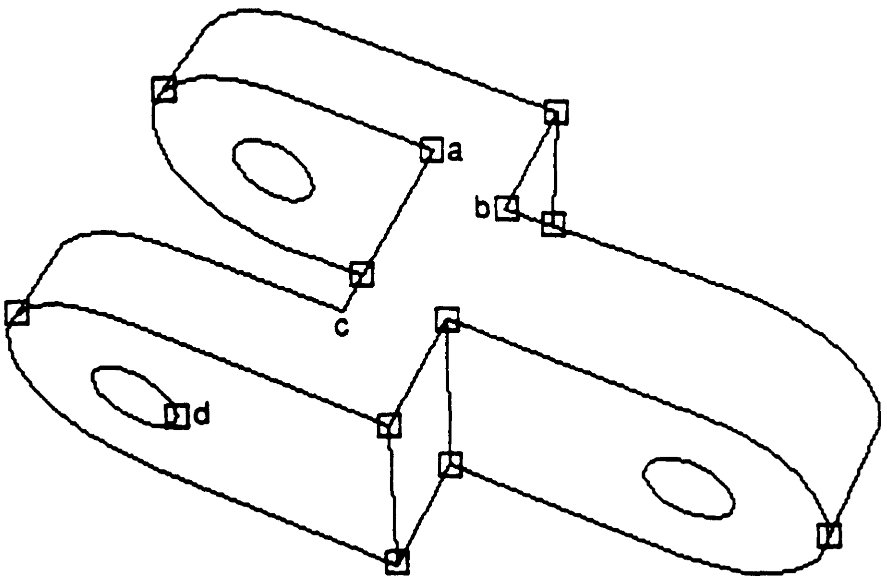


Figure 2-16: Junctions detected in line drawing in Figure 2-14.

With the end pixels of curves known, tracing occurs by ordering pixels as in Figure 2-7. Tracing initiates and terminates at end pixels. A minor complication arises when closed curves without detected junctions, as is the case for two of the three closed curves in Figure 2-14, are present because they do not have end pixels. In order to extract these curves presence of 3-linear pixels not belonging to previously detected curves must be checked. If one is found, extraction of a closed curve begins there. The process repeats until all pixels are accounted for, requiring three steps to find two closed curves in Figure 2-14 and to verify that no others are present.

The major computational expense in segmenting line drawings is finding junction pixels and isolated terminations, and checking for closed curves not containing detected junctions. The former involves only examination of 3×3 neighborhoods, specifically detection of those containing opposing diagonal pairs and those containing single outer-layer clusters. The later requires simple detection of activated pixels assuming removal of junction pixels and pixels belonging to previously detected curves. Thus the problem is not due to excessive computation at each pixel, but instead to the large number of pixels that must be considered. For example, the size of the array containing Figure 2-14 was 512 by 387 pixels, meaning that 198144 pixels had to be checked to find terminations and junctions, and to find closed curves.

Because required operations at different array elements are independent, parallel implementation is suggested. And since such operations require little computation, implementation on fine-grain hardware is feasible. Thus detection of junction pixels, isolated terminations, and closed curve pixels was implemented on a model CM-2 Connection Machine configured as a 2D grid of processors, one for each array element. The improvement in performance over serial array scanning on a Symbolics model 3650 Lisp Machine is substantial. Consider for example the processing of the array containing the line drawing in Figure 2-14. Because the size of this array was 512 by 387 pixels, 8K processors of the Connection Machine were configured as a 512 by 512 grid of virtual processors. Each array element was assigned to a virtual processor. The time required for detection of junctions and terminations decreased from 191 seconds for serial array scanning to 1 second for the parallel implementation. The time required for extracting curves decreased from 68 seconds to 8 seconds due to replacement of serial with parallel search for closed curve pixels. The total time required for segmentation dropped from 260 seconds for the serial implementation to 10 seconds for the parallel implementation.

2.4.2 Intermediate Representation for Recognition

Once line drawings are segmented, curve description can proceed. Above two descriptors whose coarseness is consistent with the loss of information due to digitization are considered, namely bounds on total curvature and concavity count. Since we are interested in qualitative descriptions (those resulting in a finite number of classifications), only concavity count will be employed. Bounds on total curvature could be converted to a qualitative description via quantization, but that will not be considered.

Extraction of concavity count from segmented curves occurs exactly as stated above, with one minor complication. Because of the difficulties discussed previously in detecting corners via nonlinear 3×3 neighborhoods, the interpretation of junctions associated with two curve ends (belonging to one closed or two open curves) is unclear. Thus in processing the drawing in Figure 2-14 whose detected junctions are pictured in Figure 2-16, junctions a, b, and d are ignored and their associated curves combined (except for junction d, which is associated with a single closed curve). Concavity count extraction from combined curves occurs exactly as stated above for curves whose pixels are not all 3-linear.

The results for Figure 2-14 are pictured in Figure 2-17, where the signed concavity counts determined for each curve are indicated along with their assumed directions of travel. The time required for calculation of all concavity counts was 56 seconds. Thus the total time required for processing the line drawing in Figure 2-14 was 316 seconds for the serial implementation and 66 seconds for the parallel implementation.

In considering the richness of representations based on qualitative feature characterizations in the remaining chapters, it is concluded that view descriptions encoding only concavity count of constituent curves are inadequate. Instead additional information regarding view structure is required. Specifically the relationships among

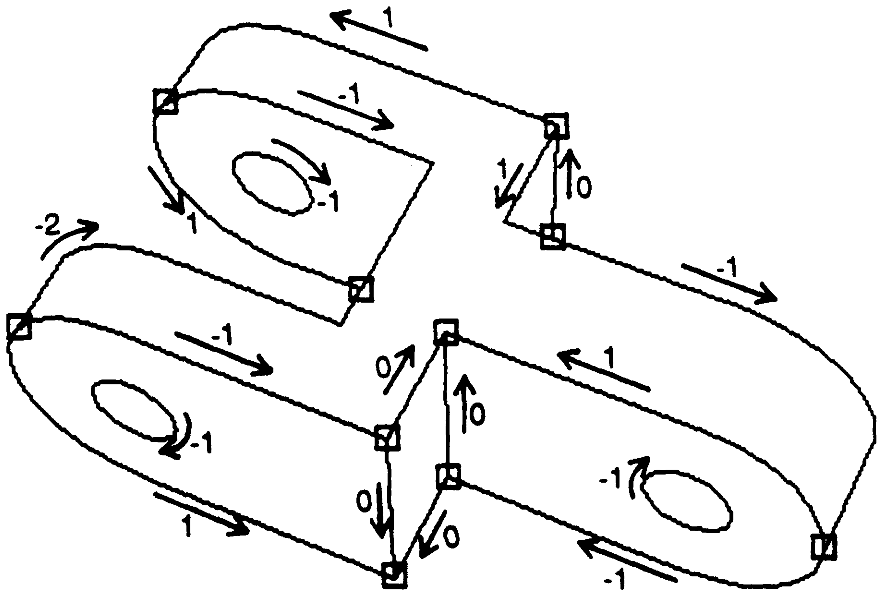


Figure 2-17: Signed concavity counts for curves in indicated directions.

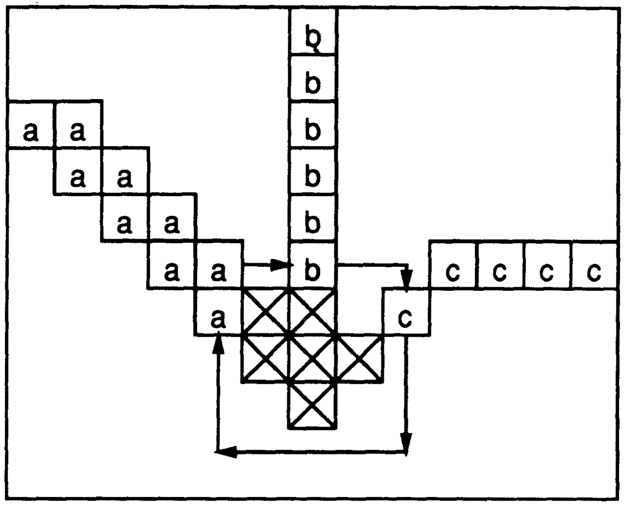


Figure 2-18: Angular ordering of curves at a junction is obtained by traversing its bounding rectangle in the clockwise direction. The clockwise ordering found here is a→b→c.

curves at vertices is exploited. Salient information already calculated includes the starting and ending orientation bounds of curves. Additional information required includes the angular ordering of curves at vertices, which can be extracted by noting the order of curves encountered as bounding rectangles for junctions are traversed in the clockwise direction (see Figure 2-18).

An intermediate view description from which structural information required for recognition can be extracted is then a graph structure whose edges encode curve information (concavity count, and starting and ending orientation ranges) and whose nodes encode the clockwise sequence of curves at junctions. Such descriptions can be viewed as data-reduced versions of line drawings obtained by coarsely describing line drawing entities.

2.5 Conclusion

We have seen that spatial quantization of continuous curves results in orientation ambiguity. Any pixel configuration generated by a straight line actually corresponds to an open interval of possible orientations. Thus to describe the curvature of a digitized curve, we must refer to a range of orientations at a particular pixel instead of a single value. With this limited information, curvature descriptions are necessarily coarse. One discussed at length above is concavity count.

In order to describe line drawings, segmentation into curves must be accomplished. This is easily performed by finding junctions and terminations at which curve tracing can be initiated. Because segmentation requires spatially independent operations over line drawing arrays, execution can be accelerated via parallel implementation.

Once segmentation is complete, extraction of curvature information from line drawings can proceed. Here extraction of concavity count from constituent curves has been emphasized. The issue addressed by the remainder of this work is whether such coarse information can result in effective recognition. In the next chapter the ability of representations constructed from qualitative information to distinguish objects is considered.

Chapter 3

Capacity of Qualitative Representations

View descriptions encoding qualitative information (such as curve concavity count) discard information, possibly resulting in identical descriptions for different objects. Thus the primary issue regarding qualitative representation is ability to distinguish objects of interest. To address this concern recognition is viewed here as a communication problem. Specifically, view descriptions are interpreted as object identity codes which must be rich enough to allow reliable decoding in the presence of distortion (possibly due to occlusion or imperfect object segmentation). In this chapter richness is measured by *capacity*, meaning the maximum number of decodable objects. An upper bound on capacity as a function of a description's constituent features is obtained, indicating a trade-off between uniqueness and allowable distortion. If this value is very large compared to the number of objects to be recognized, unknown objects will likely be identified with little ambiguity. However, distortion can be great enough to result in a value too low to yield effective recognition. In this case different descriptive features or additional recognition steps using different information must be employed, as discussed in the next chapter.

3.1 Recognition as Communication

As noted in Chapter 1, object-centered representations result in slow recognition of restricted object classes because of the dimensional mismatch between 3D object-centered and 2D image reference frames. An alternative is to use viewer-centered object representations based on qualitative description of observed features. Because a finite number of qualitatively described view features results in a finite number of possible view descriptions, object appearance can be captured in a finite number of views. However this also implies that the number of distinguishable objects is finite. Thus when considering the infinite variety of conceivable objects, sharing of view descriptions among different objects is inevitable, signifying loss of information which eliminates the possibility of unique recognition. In this case arbitrarily precise view descriptions, meaning those recording arbitrarily small changes in appearance, must be used.

But we need not concern ourselves with this situation. Although interested in recognition of *large* object sets, the reference is human performance where recognizable sets are considerably smaller. Biederman [5] for example estimates 30,000 readily distinguishable objects. Assuming finite object sets, the qualitative approach can result in different view descriptions for different objects, eliminating the need for arbitrarily precise descriptions. Clearly the important issue is not loss of information with respect to infinitely large object sets, but whether remaining information can distinguish objects in finite sets of interest.

Essentially recognition is being viewed here as communication, which classically is posed as the problem of transmitting symbols belonging to a finite set [51, 23] from one place to another. In general, a communication system can be decomposed into three basic components, namely a coder, channel, and decoder as in Figure 3-1. Simply stated, the role of the coder is to produce representations which need only contain enough data to allow discrimination by the decoder. Ideally coders are chosen to minimize the average amount of data needed per symbol to maximize transmission

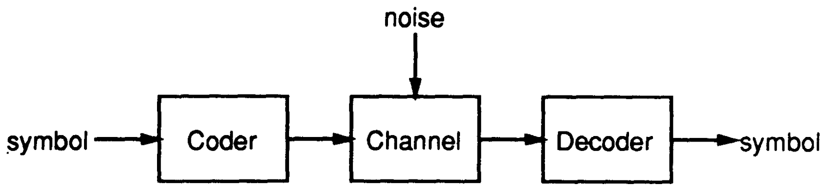


Figure 3-1: Communication system model.

rate. Assuming no channel noise, the minimum amount of data[1], usually measured in bits, depends only on the number of symbols and their relative frequencies. But because channels (cables, radio links, etc.) are typically noisy, coders must generate additional data to provide symbol representations with the redundancy[2] required for noise tolerance.

Similarly, view descriptions can be interpreted as codes which when decoded yield object identity. Thus they need only contain sufficient data for discrimination by the recognizer. View descriptions must be kept small not for the sake of transmission speed, but instead because the computational burden of recognition can be expected to increase with the amount of data present. Clearly the minimum amount of data required depends on the number of known objects, as well as the redundancy necessary for tolerance to distortion (due possibly to occlusion or imperfect object segmentation).

In order to make the analogy between recognition and communication explicit, consider Figure 3-2 where recognition is structured as the communication system in Figure 3-1. There the transmitted symbols are object identities, the coder implements imaging physics, and the channel medium is light. The goal of the observer is to decode imagery to obtain object identity.

We are concerned with the flow of information within the observer in order to minimize the computational burden of recognition. Thus view descriptions used by the observer when matching stored object descriptions to image contents must not contain excessive amounts of redundant data. A certain amount of redundancy is of course needed to counteract image distortion due to noise and occlusion. But redundancy in

[1]This value is the ***entropy*** of the transmitted symbols.

[2]An example of redundant data is the parity bit used in serial communication systems.

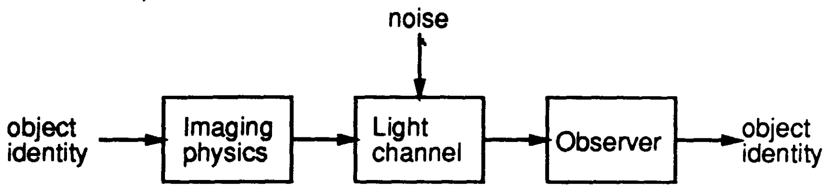


Figure 3-2: Recognition structured as a communication problem. Imaging physics codes object identity for transmission via light to the observer, which must decode the received image to obtain object identity.

excess of the amount needed to handle expected distortion yields unnecessarily large view descriptions resulting in wasted computation during matching.

The view description used could of course be the raw image received by the observer. But it appears that images contain huge amounts of redundancy, unnecessary except in extreme viewing conditions involving unusual levels of object image distortion. For example, humans can recognize most objects from their line drawings (projected depth and surface orientation discontinuities), suggesting that color, intensity, and texture data are unnecessary when such information is available. In fact, Biederman presents experimental data indicating that line drawings can be recognized almost as fast as color images [5, page 55].

Thus it appears prudent for the observer to perform data reduction on incoming imagery to strip away superfluous redundancy. This is pictured in Figure 3-3, where the observer has been decomposed into a data reducer, internal channel, and recognizer. Here it will be assumed that line drawing extraction takes place within the reducer in accordance with Biederman's results. But more importantly, we shall assume that line drawings are further reduced via qualitative description. The relevant point for the remainder of this chapter is that the final product of the data reducer is a qualitative view description. The information being qualitatively categorized is not really of concern: here line drawing information is used, but that does not exclude other types (for example color).

In terms of communication, the data reducer must produce qualitative descriptions which encode images in a manner which allows objects to be distinguished by the recognizer, which acts as a decoder. Qualitative view descriptions are passed

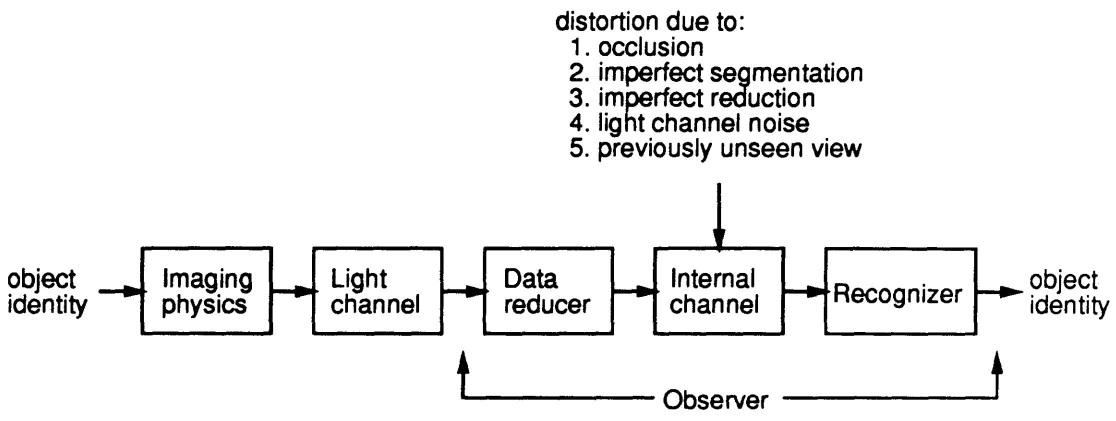


Figure 3-3: Recognition system from Figure 3-2 with observer decomposed into a data reducer, internal channel, and recognizer.

to the recognizer, which contains representations of the objects to be recognized. Since qualitative description allows object representation in terms of finite numbers of views, the role of the recognizer is to match observed view descriptions corrupted by the internal channel to those stored, thereby decoding view descriptions to yield object identity. The internal channel is primarily conceptual, serving mainly as a means of representing view description distortion in a communication framework. As mentioned previously, possible sources of distortion include occlusion and imperfect object segmentation. Additional distortion could be caused by imperfect data reduction and noise to the light channel, which is now represented as noise to the internal channel for conceptual unity. And although not an example of distortion, object views yielding previously unseen descriptions will be modeled here as such to allow unified consideration of all factors resulting in deviation of observed from stored view descriptions.

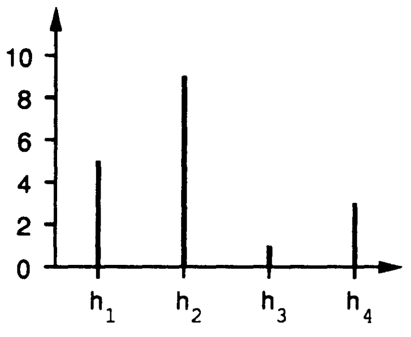
Because distortion is possible, recognition is not guaranteed to yield unique identification unless qualitative view description provides sufficient redundancy for distortion tolerance. But this may not be possible since similar objects are often distinguished by fine details not captured by coarse characterization. Thus it is not possible to guarantee in advance that an arbitrary object set is distinguishable. Instead the approach taken here is to provide enough redundancy so that substantially different objects can be distinguished in the presence of distortion, allowing ambiguity when considering similar objects. This approach is in accord with the strategy for handling large object sets outlined in Chapter 1, which suggested using coarse information to greatly reduce the set of possible objects, followed by a second recognition step using more detailed information to distinguish similar objects. Here we shall not address the second step, instead concentrating on the use of coarse information to index into large object sets, hopefully resulting in a small number of possibilities for the observed object.

One way to characterize the redundancy present in a view description is to determine its *capacity*, meaning the number of views that can be unambiguously represented in the presence of distortion[3]. Ideally, view descriptions having capacity much larger than the object set of interest should be used, suggesting that if ambiguity occurs, it will involve only a small number objects(except in unusual cases).

3.2 The View Histogram

Before considering the calculation of capacity, the exact nature of qualitative view descriptions must be defined. Because they are obtained by selecting a set of features and describing them in a manner which results in a finite number of possible categories, qualitative view descriptions can take the form of histograms whose bins represent qualitative categorizations and whose counts represent number of occurrences. For example, consider the line drawing in Figure 2-17. If the chosen features are constituent curves, they can be qualitatively characterized via concavity count as discussed in the previous chapter. In particular, concavity counts should be unsigned since a given curve can be plausibly traversed in either direction. Additional categorization is obtained by noting whether curves are open or closed. The resulting view histogram is presented in Figure 3-4.

[3]View description capacity is so-called because of its similarity to the notion of communication channel capacity, which is the maximum number of distinct messages that can be transmitted over a noisy channel without confusion. View description capacity can be thought of as the capacity of the internal channel in Figure 3-3, restricting message coding to the qualitative view characterization produced by the data reducer. Strictly speaking, though, channel capacity is a quantity independent of the particular message coding technique used.



Label	Category
h_1	0 concavities, open
h_2	1 concavity, open
h_3	2 concavities, open
h_4	1 concavity, closed

Figure 3-4: View histogram for drawing in Figure 2-17 assuming features are constituent curves categorized according to closure and concavity count.

In order to discuss view histogram capacity, the distortion to be tolerated must be quantified. This requires specification of a metric for view histograms. Because histograms with n bins can be represented as n-dimensional points, one possibility is to choose the Euclidean metric. But this choice is not particularly meaningful. Instead we might consider choosing a metric whose corresponding norm indicates in some fashion the complexity of described object views. One measure of complexity is simply the number of observed features, which is the sum of histogram bin counts. This corresponds to a metric $d(h,g)$ defining the distance between histograms h and g defined by

$$d(h,g) = \sum_{i=1}^{n} |h_i - g_i| \tag{3.1}$$

where h_i and g_i are the ith bin counts of h and g respectively. Note that as desired the size of histogram h specified by $||h|| = d(h,0)$ is

$$||h|| = \sum_{i=1}^{n} h_i \tag{3.2}$$

3.3 Capacity of View Histograms

Let histogram h be distorted as discussed above to yield perturbed histogram $\hat{h}$. Here distortion will be measured by $d(h,\hat{h})$. Assuming maximum distortion t, all histograms g such that $d(h,g) < t$ must be associated with h to guarantee correct matching of $\hat{h}$ with h. Consider for example the case where $n = 2$, as pictured in

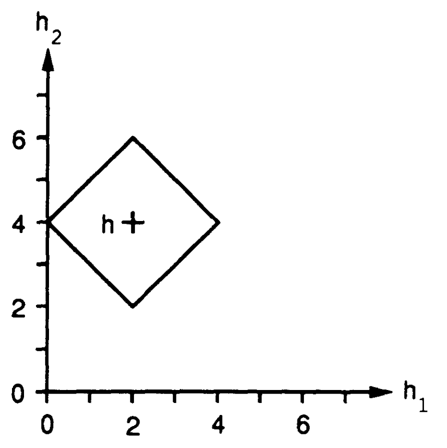


Figure 3-5: Error region for $t = 2$. Distortions of histogram $h = (2,4)$ can lie anywhere within the indicated boundary.

Figure 3-5. The indicated point represents histogram h. Since the only constraint on distortion is $d(h,\hat{h}) < t$, $\hat{h}$ can lie anywhere within the indicated region. Thus the recognizer must associate all view histograms within the error region with h in order to avoid missing correct matches.

Now consider the situation in Figure 3-6, where the error regions of two distinct histograms intersect. Since the histograms in the intersection can correspond to both h and h', ambiguity results. Thus the capacity of a view histogram representation, meaning the number of view histograms unambiguously decodable in the presence of distortion, is the maximum possible number of histograms having mutually disjoint error regions. More specifically, since a finite number of view histograms are required to represent any object, histogram size will be bounded if considering a finite object set. Assuming the maximum histogram size is M ($\|h\| \leq M$), all unperturbed view histograms must lie in the region indicated in Figure 3-7 (for $n = 2$). The problem is to determine the maximum number of histograms in that region whose error regions are mutually disjoint. This can be done approximately by noting that the error regions of all histograms indicated in Figure 3-7 must lie completely inside the region in Figure 3-8. Thus an upper bound on view histogram capacity can be obtained simply by dividing the area of the region in Figure 3-8 by the area of the error region in Figure 3-5.

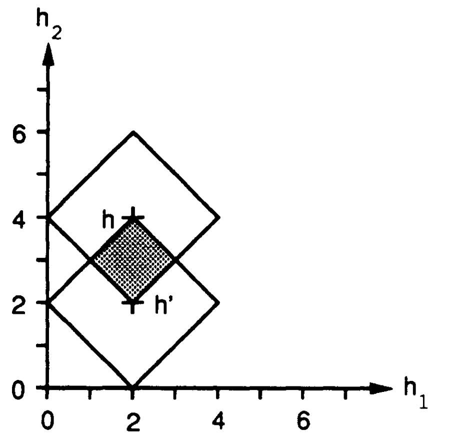


Figure 3-6: Overlapping error regions result in ambiguous assignment of histograms in intersection.

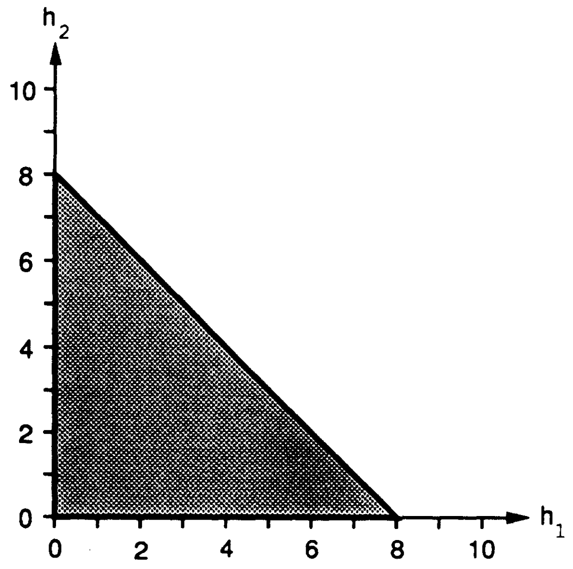


Figure 3-7: Region defined by $||h|| \leq M$ $(M = 8)$.

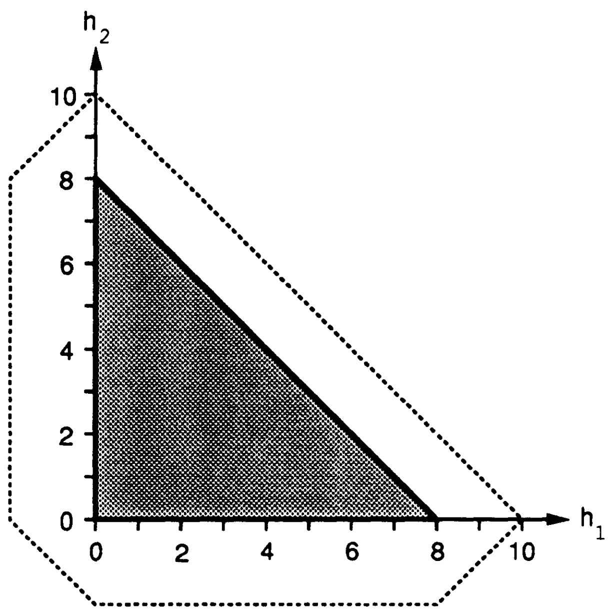


Figure 3-8: Error regions for histograms in shaded region must lie completely within dashed boundary ($t = 2$, $M = 8$).

First consider the error region. In the general case where view histograms have n bins (qualitative classification yields n categories), the error region is actually an n-dimensional solid, and we are concerned with its volume. In order to calculate volume it suffices to consider the error region centered at the origin, summing the volumes residing in each n-dimensional octant. Specifically we need only consider the octant in which all coordinates are positive (quadrant 1 when $n = 2$) because by symmetry all other volumes will be equal. Let $W_n(t)$ denote this volume, defined by $h_1 \dots h_n \geq 0$ and $\|h\| = \sum_{i=1}^{n} h_i < t$. Then

$$W_n(t) = \int_0^t dh_n \int_0^{t-h_n} dh_{n-1} \cdots \int_0^{t-\sum_{i=2}^{n} h_i} dh_1 \tag{3.3}$$

It is easy to prove that

$$W_n(t) = \frac{t^n}{n!} \tag{3.4}$$

Proceeding by induction, first assume $n = 1$. Then

$$W_1(t) = \int_0^t dh_1 = t$$

in agreement with equation 3.4. Now assume that 3.4 is valid for $n = n_1 - 1$ $(n_1 > 1)$. Then if $n = n_1$,

$$\begin{aligned} W_{n_1}(t) &= \int_0^t dh_{n_1} \int_0^{t-h_{n_1}} dh_{n_1-1} \cdots \int_0^{t-h_{n_1}-\sum_{i=2}^{n_1-1} h_i} dh_1 \\ &= \int_0^t \frac{(t-h_{n_1})^{n_1-1}}{(n_1-1)!} dh_{n_1} \\ &= \frac{1}{(n_1-1)!} \int_0^t u^{n_1-1} du \\ &= \frac{t^{n_1}}{n_1!} \end{aligned}$$

also in agreement with 3.4. Thus 3.4 is true for all $n \geq 1$. Note that since there are 2^n n-dimensional octants, with $W_n(t)$ indicating the volume contained in each, the total volume $T_n(t)$ of the error region is

$$T_n(t) = \frac{(2t)^n}{n!} \tag{3.5}$$

Now consider the region in Figure 3-8. Again this will be an n-dimensional solid in the general case where view histograms have n bins. To calculate the volume of this solid, we again consider the volume residing in each n-dimensional octant. But in this case volume will not be identical for all octants. Specifically, the volume in a given octant will depend on the number of positive coordinates present. Assume for a given octant that m and only m positive coordinates are present, where $0 \leq m \leq n$. Without loss of generality assume these coordinates are labelled $h_1 \ldots h_m$. In this case octant volume is defined by $h_1 \ldots h_m \geq 0$, $h_{m+1} \ldots h_n \leq 0$, $-t < \sum_{i=m+1}^n h_i$, and $\|h\| = \sum_{i=1}^m h_i - \sum_{i=m+1}^n h_i < M + t$. Letting $V_{n,m}(M,t)$ denote this volume, we have

$$\begin{aligned} V_{n,m}(M,t) &= \int_{-t}^0 dh_n \int_{-t-h_n}^0 dh_{n-1} \\ &\cdots \int_{-t-\sum_{i=m+2}^n h_i}^0 dh_{m+1} \int_0^{M+t+\sum_{i=m+1}^n h_i} dh_m \int_0^{M+t+\sum_{i=m+1}^n h_i - h_m} dh_{m-1} \\ &\cdots \int_0^{M+t+\sum_{i=m+1}^n h_i - \sum_{i=2}^m h_i} dh_1 \end{aligned} \tag{3.6}$$

Without too much difficulty it can be shown that

$$V_{n,m}(M,t) = \frac{1}{n!}\left((M+t)^n - \sum_{i=m+1}^{n}\begin{pmatrix} n \\ i \end{pmatrix} t^{n-i}M^i\right) \tag{3.7}$$

for all $m \geq 0$ and $n \geq m$.

First assume $n > m$. Then for $m \geq 1$ the innermost m integrals in equation 3.6 together equal $W_m(M+t+\sum_{i=m+1}^{n} h_i)$ (see equation 3.3). Employing 3.4, equation 3.6 becomes

$$V_{n,m}(M,t) = \int_{-t}^{0} dh_n \int_{-t-h_n}^{0} dh_{n-1} \cdots \int_{-t-\sum_{i=m+2}^{n} h_i}^{0} \frac{(M+t+\sum_{i=m+1}^{n} h_i)^m}{m!} dh_{m+1} \tag{3.8}$$

Coincidentally this is also correct if $m = 0$ because then the innermost integrand is 1. Proceeding by induction over n, first let $n = m+1$. Then

$$\begin{aligned} V_{m+1,m}(M,t) &= \int_{-t}^{0} \frac{(M+t+h_{m+1})^m}{m!} dh_{m+1} \\ &= \int_{M}^{M+t} \frac{u^m}{m!} du \\ &= \frac{1}{(m+1)!}\left((M+t)^{m+1} - M^{m+1}\right) \end{aligned}$$

in agreement with 3.7. Next assume 3.7 is valid for $n = n_1 - 1$ where $n_1 > m+1$. Then if $n = n_1$,

$$\begin{aligned} V_{n_1,m}(M,t) &= \int_{-t}^{0} dh_{n_1} \int_{-(t+h_{n_1})}^{0} dh_{n_1-1} \\ &\quad \cdots \int_{-(t+h_{n_1})-\sum_{i=m+2}^{n_1-1} h_i}^{0} \frac{(M+t+h_{n_1}+\sum_{i=m+1}^{n_1-1} h_i)^m}{m!} dh_{m+1} \\ &= \int_{-t}^{0} V_{n_1-1,m}(M, t+h_{n_1}) dh_{n_1} \\ &= \int_{-t}^{0} \frac{1}{(n_1-1)!}\left((M+t+h_{n_1})^{n_1-1}\right. \\ &\qquad \left. - \sum_{i=m+1}^{n_1-1}\begin{pmatrix} n_1-1 \\ i \end{pmatrix} (t+h_{n_1})^{n_1-1-i}M^i\right) dh_{n_1} \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{(n_1-1)!}\left(\int_{-t}^{0}(M+t+h_{n_1})^{n_1-1}dh_{n_1}\right.\\
&\qquad\qquad \left. - \sum_{i=m+1}^{n_1-1}\begin{pmatrix} n_1-1 \\ i \end{pmatrix} M^i \int_{-t}^{0}(t+h_{n_1})^{n_1-1-i}dh_{n_1}\right)\\
&= \frac{1}{(n_1-1)!}\left(\frac{(M+t)^{n_1}-M^{n_1}}{n_1} - \sum_{i=m+1}^{n_1-1}\begin{pmatrix} n_1-1 \\ i \end{pmatrix} M^i \frac{t^{n_1-i}}{n_1-i}\right)\\
&= \frac{1}{n_1!}\left((M+t)^{n_1} - M^{n_1} - \sum_{i=m+1}^{n_1-1}\begin{pmatrix} n_1 \\ i \end{pmatrix} t^{n_1-i}M^i\right)\\
&= \frac{1}{n_1!}\left((M+t)^{n_1} - \sum_{i=m+1}^{n_1}\begin{pmatrix} n_1 \\ i \end{pmatrix} t^{n_1-i}M^i\right)
\end{aligned}$$

in agreement with 3.7. Thus 3.7 is valid for all $m \geq 0$ and $n > m$.

The proof is completed by considering $n = m$. Note from equation 3.6 that in this case $V_{m,m}(M,t)$ is equivalent to $W_m(M+t)$, so that

$$V_{m,m}(M,t) = \frac{(M+t)^m}{m!}$$

in agreement with 3.7. Thus the validity of 3.7 is extended to all $n \geq m$ where $m \geq 0$.

Now consider the total volume $V_n(M,t)$ of the region in Figure 3-8 generalized to n dimensions. First note that since there are $\begin{pmatrix} n \\ m \end{pmatrix}$ n-dimensional octants with m positive coordinates, and $V_{n,m}(M,t)$ is the volume contained in each, the volume contributed by such octants is $\begin{pmatrix} n \\ m \end{pmatrix} V_{n,m}(M,t)$. Total volume can be obtained by summing over m:

$$\begin{aligned}
V_n(M,t) &= \sum_{m=0}^{n}\begin{pmatrix} n \\ m \end{pmatrix} V_{n,m}(M,t)\\
&= \frac{1}{n!}\sum_{m=0}^{n}\begin{pmatrix} n \\ m \end{pmatrix}\left[(M+t)^n - \sum_{i=m+1}^{n}\begin{pmatrix} n \\ i \end{pmatrix} t^{n-i}M^i\right]
\end{aligned}$$

Recall that an upper bound on histogram capacity $C_n(M,t)$ is obtained by dividing the volume of the region in which the error region must lie by the volume of the error region. Thus

$$\begin{aligned} C_n(M,t) &\leq \frac{V_n(M,t)}{T_n(t)} \\ &= \sum_{m=0}^{n} \binom{n}{m} \left[\left(\frac{M+t}{2t}\right)^n - \sum_{i=m+1}^{n} \binom{n}{i} \frac{1}{2^n} t^{-i} M^i \right] \\ &= \frac{1}{2^n} \sum_{m=0}^{n} \binom{n}{m} \left[\left(\frac{M}{t}+1\right)^n - \sum_{i=m+1}^{n} \binom{n}{i} \left(\frac{M}{t}\right)^i \right] \end{aligned}$$

Note that the last expression depends on M and t only in terms of their quotient. Labeling this expression $\widehat{C}_n(M/t)$ to distinguish it as an upper bound on $C_n(M,t)$, further reduction is possible by replacing $\left(\frac{M}{t}+1\right)^n$ by its Binomial Theorem expansion:

$$\widehat{C}_n\left(\frac{M}{t}\right) = \frac{1}{2^n} \sum_{m=0}^{n} \binom{n}{m} \left[\sum_{i=0}^{n} \binom{n}{i} \left(\frac{M}{t}\right)^i - \sum_{i=m+1}^{n} \binom{n}{i} \left(\frac{M}{t}\right)^i \right]$$

which yields

$$\widehat{C}_n\left(\frac{M}{t}\right) = \frac{1}{2^n} \sum_{m=0}^{n} \binom{n}{m} \sum_{i=0}^{m} \binom{n}{i} \left(\frac{M}{t}\right)^i \tag{3.9}$$

Of course an exact expression for capacity would be preferable to the upper bound given by $\widehat{C}_n$, but there appears to be no straightforward method for deriving one. In particular, a volume-packing method for spaces of arbitrary dimension would be required, capable of taking into account location of error volumes exclusively at discrete points. Thus, in the remainder of this work $\widehat{C}_n$ will be used as an indication of view histogram capacity.

As stated above, the goal is to choose features whose qualitative categorizations result in a histogram capacity much larger than the object set to be recognized. Of importance is not only the number of categories resulting from qualitative description (n), but also the ratio of histogram size to distortion denoted by M/t. This quantity can in a sense be regarded as a *signal to noise ratio*, which in agreement with intuition

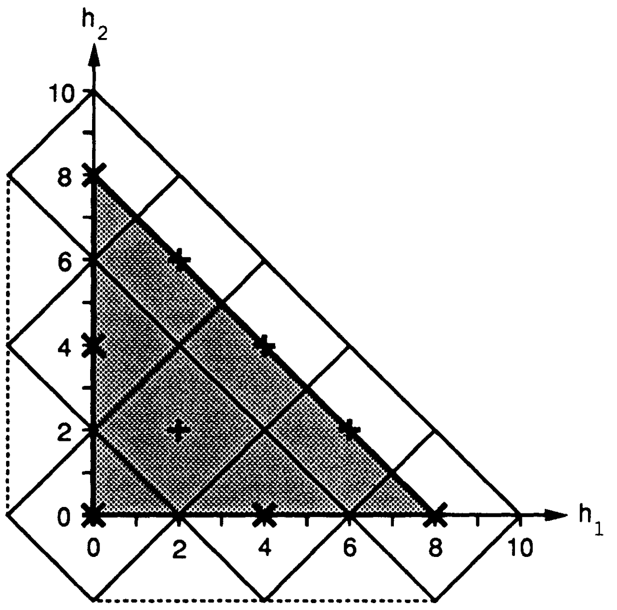


Figure 3-9: Nine histograms with mutually exclusive error regions (t = 2,M = 8).

decreases capacity when decreased itself. In other words, increasing distortion with respect to histogram size results in progressive deterioration of the recognizer's ability distinguish objects, as demonstrated in the next chapter.

Above the role of redundancy in providing tolerance to view distortion was discussed. In order to further explore the notion of view histogram redundancy consider the case where $t = 2$ and $M = 8$ as in the figures accompanying the text above. For this case equation 3.9 yields an upper bound on capacity of 11, which appears to be a reasonable estimate since at least 9 histograms with mutually exclusive error regions can be found (see Figure 3-9). Thus at most 11 objects can be distinguished. Now consider the distortion-free case, where every histogram can represent a distinct object. When $M = 8$ there are $1+2+\ldots+9$ possible histograms, allowing representation of 45 objects. Since there is at least a factor of four decrease in number of distinguishable objects when distortion is present, roughly 2 bits of histogram data must be used for distortion tolerance. And as distortion increases the amount of redundant data needed will increase, reducing the data available for actually distinguishing objects, thus increasing recognition ambiguity.

3.4 Conclusion

The significant result of this chapter is the upper bound on view histogram capacity given by equation 3.9. Because it is not likely that stored view histograms will be nicely distributed in histogram space as in Figure 3-9, we can reasonably expect that capacity must be much larger than the object set of interest if qualitative recognition is to provide reasonable resolution. Thus one of the main goals of the next chapter is to seek out features yielding view histograms with large capacities, and evaluate resulting recognition performance.

The mechanics of recognition to be employed there essentially have already been discussed. The general idea is to compare view histograms constituting stored object representations with those from observed objects. If the difference between an observed histogram $\hat{h}$ and stored histogram h defined by $d(\hat{h}, h)$ is less than maximum tolerable distortion t, then the object to which h belongs is considered a possibility for the viewed object.

To be more precise, view histograms will be obtained by sampling the viewing sphere of each object in Figure 1-4. Thus each will represent the appearance of an object as seen from a small patch on its viewing sphere. If recognition is then posed as the problem of distinguishing view histograms instead of objects, recognition results in orientation estimation in addition to identification. This approach requires histogram capacity to be much larger than the total number of views sampled for all objects of interest, as discussed next.

Chapter 4

Selection of Features

In the previous chapter an upper bound on view histogram capacity was obtained. The significance of capacity is that it in a sense measures representational richness. Large capacity histograms can distinguish many objects even when distorted (by imperfect segmentation, for example). The goal here is to choose features whose histogram capacity is much larger than the object set of interest, hopefully resulting in little recognition ambiguity.

The features considered consist of line drawing curves whose extraction and qualitative classification (via concavity count) were described in Chapter 2. It is shown that simple features consisting of individual curves do not yield sufficient capacity. Instead curves must be grouped into compound features whose qualitative description results in significantly increased capacity.

The performance of high-capacity histograms in distinguishing objects is demonstrated on scenes containing one and two objects. When a single object is present, recognition typically results in little ambiguity, meaning that observed histograms fall in the error regions of a small number of stored histograms. With multiple objects, reliable segmentation is needed to prevent segmentation error from enlarging error regions to the point where ambiguity becomes unacceptable.

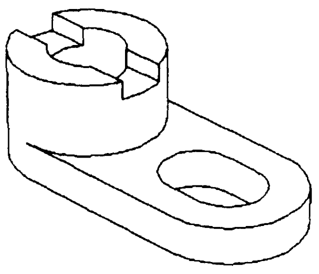


Figure 4-1: Object 15 as modeled.

4.1 The Object Set

Of interest in this chapter are the objects in Figure 1-4, which were taken from a mechanical drawing textbook [24]. Admittedly this is not a very large set by human standards, but it is much larger than those typically encountered in the recognition literature, which typically contain one object, or perhaps a few quite simple ones (see Chapter 1).

These objects were chosen primarily for three reasons. First, they are not trivial geometrically. They have holes and curved surfaces, and none are convex. Second, even though complicated they are similar enough to avoid discrimination by trivial test. For example, counting number of holes would not be an effective recognition strategy. Thus they present a reasonable proof-of-concept test for recognition based on coarse, qualitative information: if similar objects can be distinguished, then success can be anticipated for much larger sets containing substantially different objects. And third, the objects can be modeled by the Symbolics S-Geometry 3D geometry package [64], used by the author to generate object views from arbitrary viewpoints[1].

Models are exact duplicates of the objects in Figure 1-4 with one exception: the recessed hole in object 15 was not modeled because of difficulties in doing so with S-Geometry construction tools. Instead it was replaced by a single hole as in Figure 4-1.

Since S-Geometry can handle only polyhedra, curved surfaces are approximated

[1]Hidden line removal was performed with code supplied by the author.

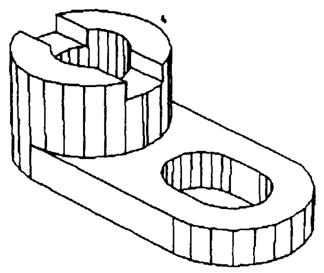


Figure 4-2: Polyhedral approximation of object 15.

as in Figure 4-2. But approximating facets are small enough so that planar approximation is not discernible in line drawings considered.

4.2 The Viewing Sphere

As mentioned previously, qualitative object representations are obtained by collecting view histograms obtained at sampled viewpoints. The approach taken here is to sample a given object's *viewing sphere*, which contains the object at its center and defines all viewpoints at a certain distance. The observer is assumed to reside on the sphere, its lens pointed toward the center. The size of the viewing sphere is identical for every object, yielding the scale shown in Figure 4-3 for the 50mm focal length used during sampling.

To discuss viewpoint sampling a viewing sphere coordinate system must be defined. To do so, first associate with each object a rigidly-attached 3D coordinate system oriented as in Figure 4-4 for the object attitudes shown in Figure 1-4. For a particular object, let the origin of its coordinate system lie roughly at the centroid of its bounding box (the one having faces parallel to the coordinate planes). With the origin defining the center of the viewing sphere, position on the sphere can be specified by latitude and longitude defined as follows. First let the equator of the viewing sphere reside in the x-z plane, with latitude defined such that the intersec-

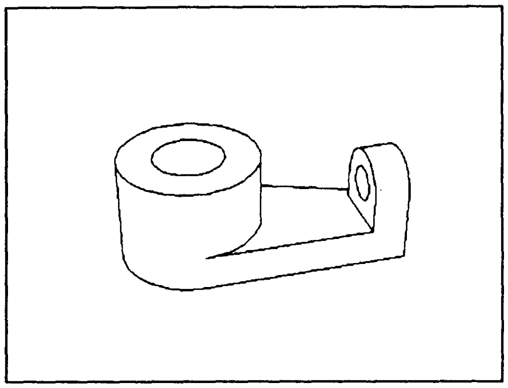


Figure 4-3: Object scale during viewpoint sampling. Image size is 512×387 pixels.

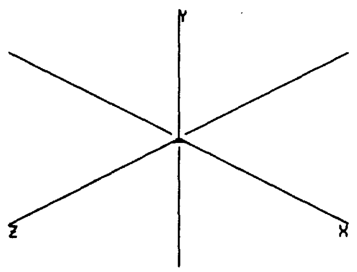


Figure 4-4: 3D coordinate system assumed for objects in Figure 1-4.

tion of the sphere and the positive (respectively negative) y-axis occurs at +90 (-90) degrees latitude. Furthermore, let longitude be defined such that it increases in the counter-clockwise direction when the viewing sphere is viewed from above on the positive y-axis, with the intersection of the sphere and the positive z-axis occurring at 0 degrees longitude.

The sampling pattern chosen allows samples to represent viewing sphere patches of roughly equivalent area. The basic idea is to sample latitude parallels in a manner such that the number of samples obtained is proportional to the cosine of latitude. To be more precise, assume ℓ viewpoint samples are obtained from the equator via uniform sampling, resulting in a longitude sampling interval there of $2\pi/\ell$ radians. The number of longitude samples taken from a parallel of latitude ϕ is then $\lceil \ell \cos(\phi) \rceil$ ($\lceil \cdot \rceil$ rounds its argument to the next higher integer), resulting in approximately the same inter-sample distance found at the equator, namely $2\pi/\ell$ (normalized for viewing sphere radius 1). If the sampled parallels are separated by latitude intervals of $2\pi/\ell$ radians, each viewpoint sample then accounts approximately for a viewing sphere

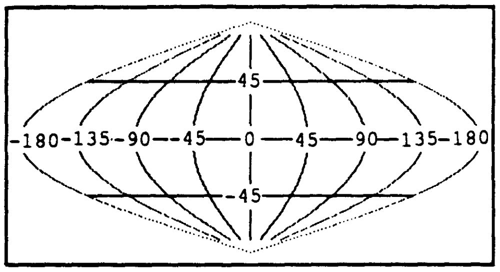


Figure 4-5: The Sinusoidal projection. Parallels and meridians are marked at 45 degree intervals.

patch of normalized area $(2\pi/\ell)^2$.

In the following sections it will prove useful to represent the viewing sphere schematically. Specifically the distribution of viewpoints whose view descriptions are captured by the above sampling of the viewing sphere will be of interest. In order to represent different regions on the viewing sphere equally, an area-preserving projection[2] of the viewing sphere is indicated. Here the Sinusoidal projection will be employed, as pictured in Figure 4-5. The Sinusoidal projection is convenient because of its similarity to the viewing sphere sampling strategy described above. In particular, projected parallels are uniformly spaced, their length varying with the cosine of latitude.

In Figure 4-6 an example of sampling the viewing sphere as discussed above is presented. The views so obtained are structured in a manner reminiscent of the Sinusoidal projection.

4.3 Unstable Viewpoints

Accidental viewpoints occur when view topology is unstable with respect to observer position on the viewing sphere. They are always found when object surfaces line up with the viewer's line of sight because then infinitesimal travel can result either in surface exposure or occlusion.

Because accidental viewpoints occupy a zero-area subset of the viewing sphere,

[2]An area-preserving projection is one in which the relative areas of different regions are preserved.

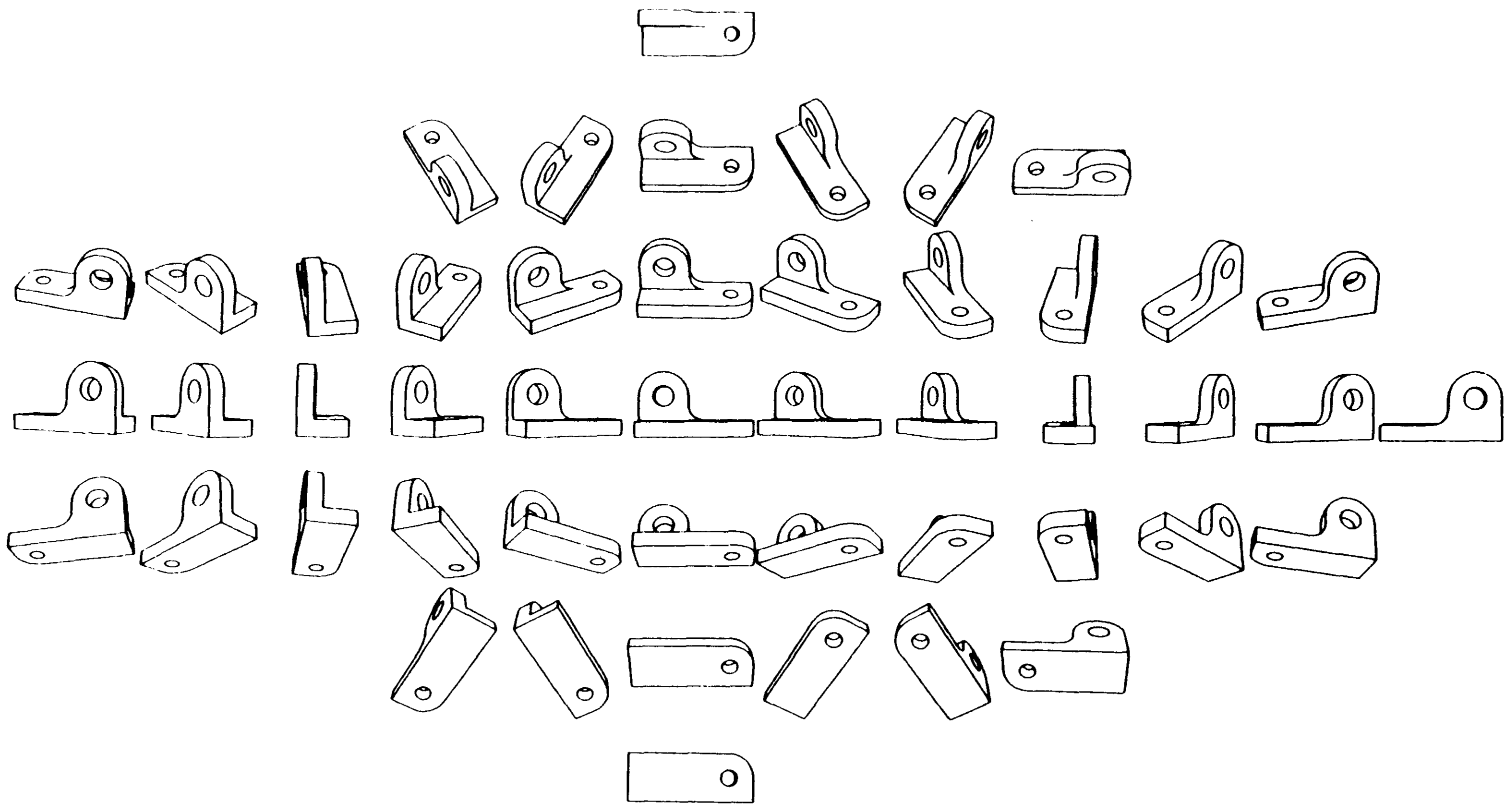


Figure 4-6: Views obtained by sampling the viewing sphere of object 3 ($\ell = 12$).

they are not of concern if the imaging sensor is spatially continuous. But in our case the imaging sensor is spatially discrete, which causes instability on the viewing sphere to spread to viewpoints in the vicinity of accidental viewpoints. Consider for example the effect on a planar face. As the face approaches alignment with the viewer, foreshortening causes face boundaries to interact in a manner that yields an unstable configuration of false curves and junctions, as demonstrated in Figure 4-7. The result is an unstable region of non-zero area on the viewing sphere. The extent of the unstable region will be indicated in Section 4.5, but for now it suffices to note that it can be significant.

This is of concern when evaluating the number of viewpoint samples needed to capture the appearance of a 3D object from all perspectives. One possible approach for doing so is to increase the number of samples until new view descriptions are no longer found. Such would be reasonable approach if viewpoint instability was limited to a zero-area region of the viewing sphere. But in our case it is not, so we can expect the fraction of view descriptions corresponding to unstable viewpoints to increase with the number of viewpoints tested, forcing storage of a great many descriptions varying from those obtained at stable viewpoints in a not particularly significant manner. Instead the approach taken here is to choose what appears to be a reasonable number of samples and set the distortion tolerance t discussed in the last chapter to take into account that certain observed view descriptions may not exactly match those stored in the recognizer. Specifically the number of equatorial samples ℓ used will be 24, resulting in 188 viewpoint samples for each object as plotted in Figure 4-8. Selection of t will be discussed in a later section.

4.4 Recognition Strategy

The representation used for recognition of each object in Figure 1-4 will consist of view histograms obtained at the viewpoints specified in Figure 4-8. Thus there will be a total of $16 \times 188 = 3008$ stored histograms, each corresponding to an object and orientation. The histograms are determined by qualitative characterization of selected

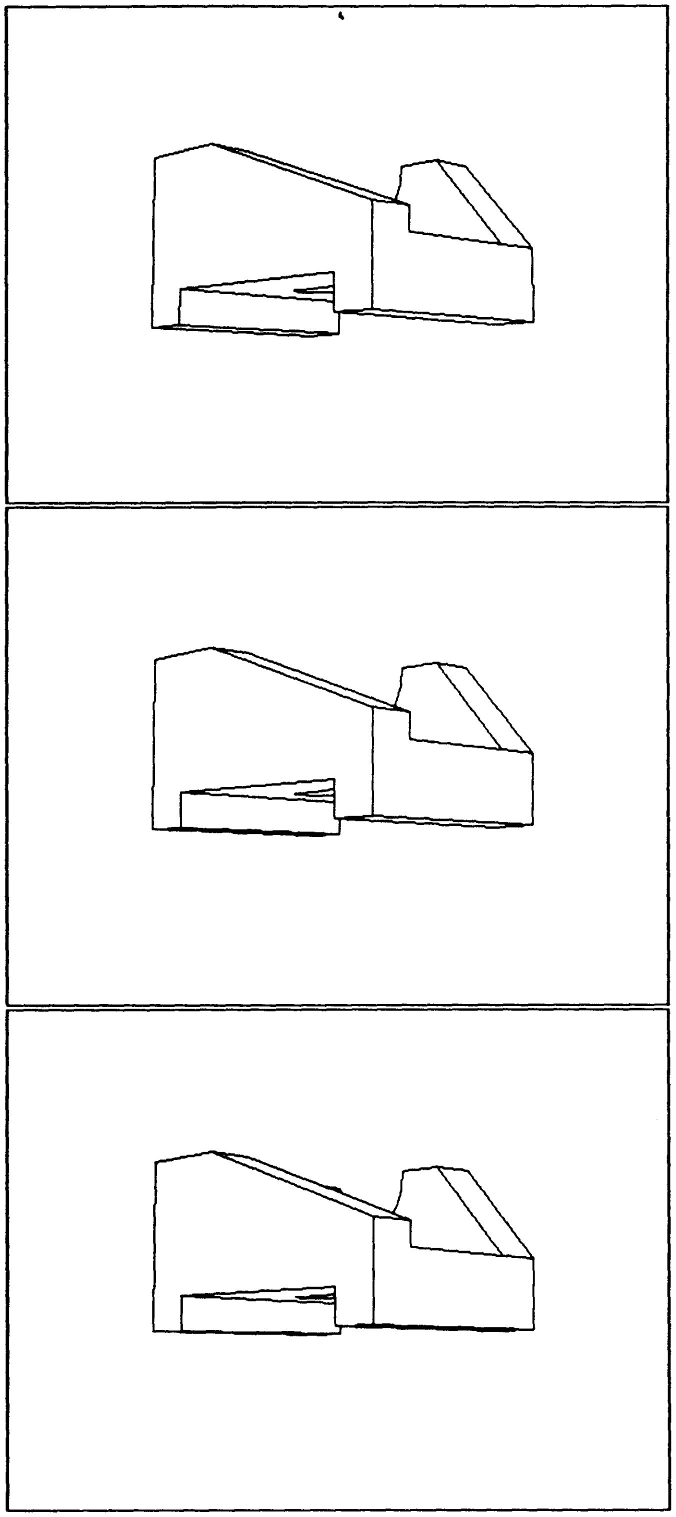


Figure 4-7: Base faces of object 9 approach alignment with the viewer (from top down). Note the creation of false junctions and curves due to interaction of digitized face boundary curves.

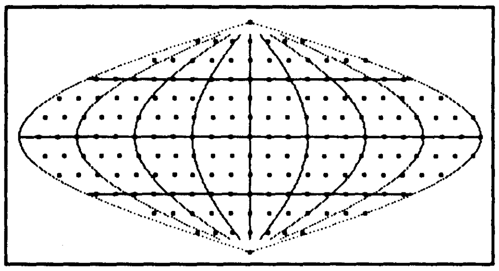


Figure 4-8: Dots indicate sampling of viewing sphere used in generating representations for the objects in Figure 1-4. There are 188 samples, corresponding to $\ell = 24$.

features. Here features will consist of segmented line drawing curves as discussed in Chapter 2. Simple features will consist of single curves described qualitatively via unsigned concavity count and closure. Compound features will consist of 2 curves meeting at line drawing junctions, serving to introduce information regarding line drawing structure.

Because object representations encode coarse information, we can not expect recognition to distinguish similar objects. But confusion among many objects is not acceptable, so the goal here is to associate unknown object views with a small number of stored histograms, resulting in ambiguity among only a few different objects. To uniquely identify an unknown object, a second recognition stage employing more precise object information could be applied to the small number of remaining possibilities. As discussed in Section 1.2, recognition then involves an *indexing* stage employing coarse information to reduce the object set to a small number of similar possibilities, followed by a verification stage employing detailed information to obtain unique identification. This two-stage paradigm is proposed for reasons of efficiency. Since it involves manipulation of simple information, indexing occurs quickly, leaving time-consuming verification to a greatly reduced set of objects.

The verification stage will not be addressed here. Instead emphasis will be placed on achieving adequate discrimination from indexing. Thus the goal of the next section is to find line drawing features whose qualitative categorization results in view histogram capacity much larger than the number of stored histograms constituting object representations. Since each stored histogram is associated with an object and a viewpoint, this will result in little ambiguity regarding not only identity, but orientation as well.

Before proceeding to a discussion on capacity, let us reconsider the procedure described in the previous chapter for associating an observed view histogram $\hat{h}$ with stored histogram h. There the association was made if $d(h, \hat{h})$ was less than tolerable distortion t where t was assumed constant for all h. This is not particularly reasonable since it assumes that views containing a few features will be distorted to the same extent as those containing many. Instead a more intuitive approach is to make t proportional to the number of view features present, namely $\|h\|$. Thus in all that follows $\hat{h}$ will be associated with h if $d(h, \hat{h}) < k\,\|h\|$, or alternatively if

$$\frac{d(h, \hat{h})}{\|h\|} < k \tag{4.1}$$

The left-hand term of this inequality will be known as the *relative distortion* for histogram h.

A relatively small fraction of stored view histograms will be significantly affected by this modification because the objects in Figure 1-4 are of roughly the same complexity, meaning that most views contain similar numbers of curves[3]. Then approximately speaking we expect $\|h\|$ to be constant for most stored views, resulting in constant distortion tolerance $t = k\,\|h\|$, and therefore approximate validity of histogram capacity upper bound $\widehat{C}_n$ defined in 3.9. To rewrite 3.9 in terms of k note that since M was defined in Chapter 3 as the maximum stored histogram size, we can let $M = \|h\|$ in this case, resulting in $\frac{M}{t} = \frac{M}{k\|h\|} = \frac{1}{k}$. Substituting into 3.9 we get

$$\widehat{C}_n(\frac{1}{k}) = \frac{1}{2^n} \sum_{m=0}^{n} \begin{pmatrix} n \\ m \end{pmatrix} \sum_{i=0}^{m} \begin{pmatrix} n \\ i \end{pmatrix} \left(\frac{1}{k}\right)^i \tag{4.2}$$

$\widehat{C}_n$ as defined in 4.2 will be used in the next section as an indication of histogram capacity.

[3]An exception is object 5, which in the view shown appears considerably simpler than the others.

4.5 Capacity Evaluation

The goal here is to find line drawing features whose histogram capacity is much larger than the total number of stored histograms, namely 3008. First to be considered are features consisting of single curves, qualitatively characterized according to closure and concavity count as discussed in Section 3.2. To apply the upper bound defined in 4.2, values for k and n are required.

First consider n. In the course of deriving 4.2 in the last chapter, it was implicitly assumed that the counts in n histogram bins exhibit significant variation, specifically over the range $[0, M]$. If violated, the actual region in which histograms are restricted is less than that assumed, resulting in an unreasonably high capacity estimate from 4.2. Thus a satisfactory approach in setting n is not to automatically choose the number of qualitative categories found during viewpoint sampling. Instead some provision must be made for preventing bins whose counts exhibit inadequate variation from contributing to n.

The variation measure used here will be *entropy* [51, 23], which is defined as follows. Letting probability measure P be defined on sample space $S = \{\omega_1, \ldots, \omega_m\}$, entropy H of S is defined as

$$H(S) = -\sum_{i=1}^{m} P(\omega_i) \log P(\omega_i) \tag{4.3}$$

If the base of the logarithm in this definition is 2, entropy is measured in *bits*.

In our case S is the set of counts observed for a particular histogram bin during viewpoint sampling, and P is defined such that all sampled viewpoints for all objects are given equal weight. Thus the probability of a given bin count is the number of stored histograms where it occurs divided by the total number of stored histograms (3008).

Entropy is a particularly good measure of variation in this case because it takes into account relative frequency. Noting only which bin counts are found is not satisfactory. For example, it is possible that counts for a particular bin are found throughout $[0, M]$, although tightly concentrated at only one value. Then the variation for this

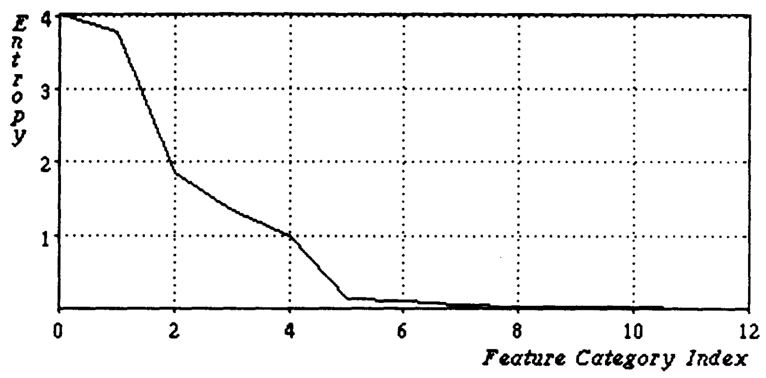


Figure 4-9: Bin entropies (in bits) for qualitative curve classification.

bin does not significantly expand the histogram region, and should not be included in n. In agreement, the bin count entropy for this case is approximately zero.

In order to exclude bins whose counts exhibit little variation, a lower threshold on entropy must be specified. Here a value of 1 bit is used, as would be obtained if only 2 equiprobable bin counts were found, which is considerably less than M considering that histogram size (number of curves) is typically close to 10 or higher for the library objects.

Thus only bins whose entropy is at least 1 bit contribute to n. In Figure 4-9 the entropies for all bins are plotted in decreasing order. This information is presented in tabular form in Table 4.1 to show the correspondence between curve category and entropy. Since only 4 of 13 categories have entropy of at least 1 bit, we shall use $n = 4$ in calculating histogram capacity.

Now consider k, the maximum tolerable relative distortion. We first address the situation where unknown object views are not distorted. In this case the only possible discrepancy between observed and stored view histograms is due to insufficiently dense sampling of the viewing sphere, resulting in incomplete collection of view histograms. As discussed in Section 4.3, no attempt is made to capture all possible view histograms when object representations are compiled. Instead the strategy is to sample the viewing sphere as specified in Figure 4-8, adjusting the distortion tolerance to capture additional viewpoints. To be precise, a viewpoint is considered captured if its histogram resides within the error region of a stored histogram, allowing association

Category Index	Concavities	Closure	Entropy (bits)
0	0	open	4.015
1	1	open	3.790
2	1	closed	1.858
3	2	open	1.345
4	3	open	0.983
5	3	closed	0.144
6	5	open	0.098
7	2	closed	0.050
8	4	open	0.029
9	4	closed	0.027
10	7	open	0.024
11	5	closed	0.008
12	7	closed	0.004

Table 4.1: Curve classification entropies.

with the corresponding viewing sphere sample.

Determining a plausible value for k thus requires evaluation of the extent of the viewing sphere captured as a function of k. The approach used here is to densely sample the viewing sphere, determining which viewpoints are captured for various values of k. To choose a uniformly acceptable value for k, this procedure should be applied to all library objects. This is not practical here, though, because dense sampling just one object involves considerable time due to lengthy hidden line removal required at each viewpoint. Instead, just one object is considered, namely object 9, with the hope that results so obtained are reasonably representative.

Dense sampling employed 120 equatorial samples, resulting in 4614 viewing sphere samples, or 25 times the number used for object representation. Figure 4-10 contains the Sinusoidal maps of captured viewpoints for various values of k (the map labelled $k = 0$ represents viewpoints whose histograms are present in the stored set). Note the banding that takes place about the equator and the -90, 0, +90, and 180 degree meridians in the plots for smaller values of k. This is a direct result of viewpoint instability as discussed in Section 4.3. There it was noted that instability occurs when planar faces are nearly aligned with the viewer. Because the dominant faces of object 9 are parallel to the coordinate planes in Figure 4-4, significant alignment takes place in the vicinity of the equator for faces parallel to the x-z plane, near the 0 and 180 degree meridians for faces parallel to the y-z plane, and near the -90 and +90 degree meridians for faces parallel to the x-y plane. Thus the banding seems to indicate large differences between histograms in these regions and those stored primarily due to viewpoint instability in the vicinity of accidental viewpoints.

It is not reasonable to require that the entire viewing sphere be captured because of the large values of k required by unstable viewpoints. Specifically, viewpoints away from unstable regions are captured by $k = .20$, whereas several unstable viewpoints remain free at $k = .35$. In fact, the value of k required for capture of all densely-sampled viewpoints is 2.0. Since our upper bound on capacity decreases rather sharply as k increases, here some of the unstable views will be conceded in favor of reduced recognition ambiguity elsewhere by choosing $k = .20$. This represents a departure from the earlier stated requirement of representing object appearance from all perspectives, although not too unreasonable in light of human difficulties in recognizing accidental views.

With $n = 4$ and $k = .20$, our upper bound on capacity $\widehat{C}_4(5)$ is 318. Since the number of stored viewpoints is 3008, we have no hope of achieving effective viewpoint indexing in this case. Histograms based on single-curve view features categorized according to concavity count and closure simply do not have the required capacity, even when recognition of accidental views is conceded.

In order to boost capacity several strategies are possible. Noting that our upper bound increases with n, we might expect a finer curve categorization to yield satisfactory capacity. But care must be taken here since increased fineness may force an offsetting increase in k due to decreased classification robustness.

An alternative approach which preserves curve classification coarseness is to group curves, thereby increasing the number of feature categories combinatorially[4]. Consider for example pairs of curves. If classified via classification of constituent curves,

[4]Biederman [5] similarly suggests using grouping to increase the number of objects representable by 3D "components".

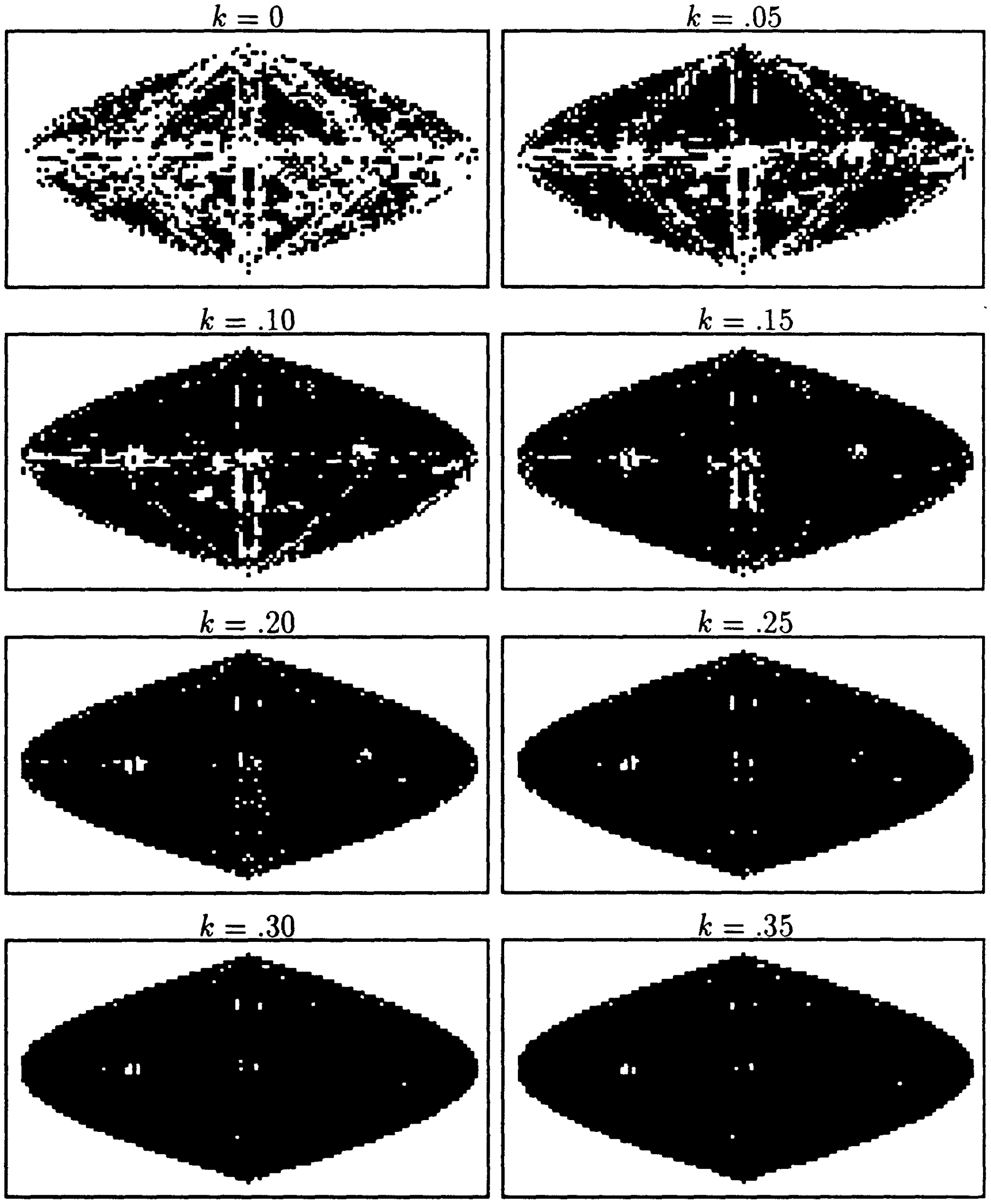


Figure 4-10: Captured viewpoint maps for single-curve view histograms (black dots indicate capture).

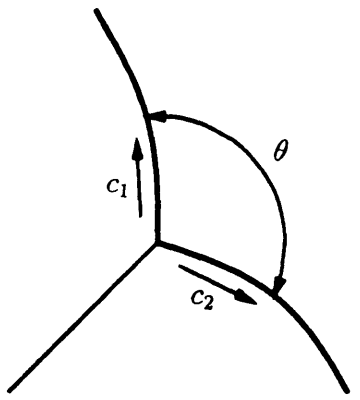


Figure 4-11: Adjacent curves at line drawing junctions form compound view features. c_1 and c_2 are signed concavity counts for curve travel in the indicated directions.

the number of pair categories is the number of curve categories squared.

But care must be taken in forming pairs. It is not satisfactory for example to simply collect all possible pairings from views because resulting histograms can then be derived from corresponding single-curve histograms, and therefore contain no additional information. More informative view descriptions can be obtained by pairing only intersecting curves, thereby encoding information regarding line drawing structure. Such an approach is taken here, first by ordering the curves at each line drawing junction in the clockwise direction as discussed in Section 2.4.2, then by pairing consecutive curves as in Figure 4-11.

Assuming constituent curves are labelled 1 and 2 to indicate clockwise ordering, a given pair can be categorized according to signed concavity counts c_1 and c_2, calculated assuming direction of travel away from the curve junction[5]. Additional classification is obtained via qualitative description of angle θ separating curves 1 and 2, in particular indicating whether θ is less than π, possibly π, or greater than π. This evaluation is made on the basis of the starting orientation ranges obtained during concavity count calculation for curves 1 and 2, as discussed in Chapter 2. Letting $(\theta_{1,min}, \theta_{1,max})$ and $(\theta_{2,min}, \theta_{2,max})$ denote these ranges, we have $\theta < \pi$ if $\theta_{1,max} - \theta_{2,min} < \pi$, $\theta > \pi$ if $\theta_{1,min} - \theta_{2,max} > \pi$, and θ is possibly π otherwise.

[5]Recall from Chapter 2 that the sign of a signed concavity count indicates whether the first detected concavity in the assumed direction of travel has positive or negative curvature.

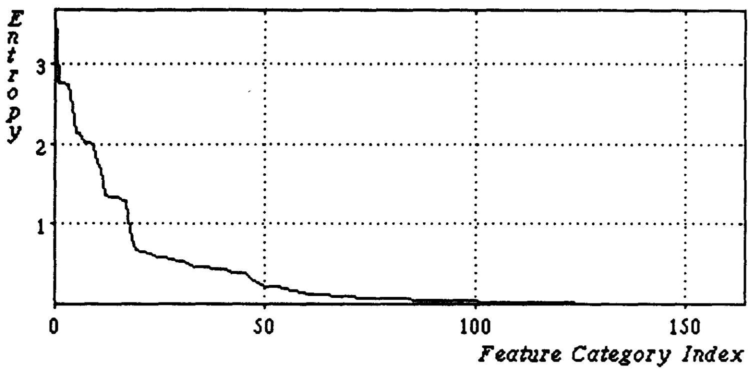


Figure 4-12: Bin entropies (in bits) for qualitative classification of curve pairs.

In considering histogram capacity for curve pairs so classified we proceed in exactly the same manner as discussed for the single-curve case. We first consider bin entropies for the histograms obtained as a result of viewing sphere sampling. In Figure 4-12 is a plot of the entropies in decreasing order, indicating that only 18 of 165 histogram bins have entropy of at least 1 bit. Thus n is set to 18. The corresponding feature classifications are listed in Table 4.2.

To determine a reasonable value for k, captured viewpoint maps for various values of k are presented in Figure 4-13. Note again the banding that takes place about the equator and the -90, 0, +90, and 180 degree meridians, especially for k = .20 and .30. It seems to be a bit more pronounced than in the single-curve case, perhaps because grouping as well as curve classification are disrupted in the vicinity of accidental views.

It appears that viewpoints away from the unstable regions are captured by $k = .4$. With $n = 18$ and $k = .40$, our upper bound on capacity $\widehat{C}_{18}(2.5)$ is 7.5×10^8, well in excess of the number of stored histograms (3008) as desired. Thus histograms based on the above categorization for curve pairs should be able to discriminate object views much more effectively than those based on single curves, resulting in better indexing. This is demonstrated in the next section.

Category Index	c_1	c_2	θ	Entropy (bits)
0	0	0	$< \pi$	3.671
1	-1	1	$< \pi$	2.766
2	0	1	$< \pi$	2.759
3	-1	0	$< \pi$	2.752
4	1	-1	$> \pi$	2.471
5	-1	-1	$< \pi$	2.159
6	1	1	$< \pi$	2.157
7	0	-1	$< \pi$	2.037
8	1	0	$< \pi$	2.017
9	0	0	$\sim \pi$	1.977
10	1	-1	$\sim \pi$	1.759
11	1	-1	$< \pi$	1.689
12	0	-1	$> \pi$	1.352
13	0	1	$\sim \pi$	1.344
14	-1	0	$\sim \pi$	1.343
15	1	0	$> \pi$	1.340
16	0	-1	$\sim \pi$	1.288
17	1	0	$\sim \pi$	1.286

Table 4.2: Classification entropies above 1 bit for curve pairs.

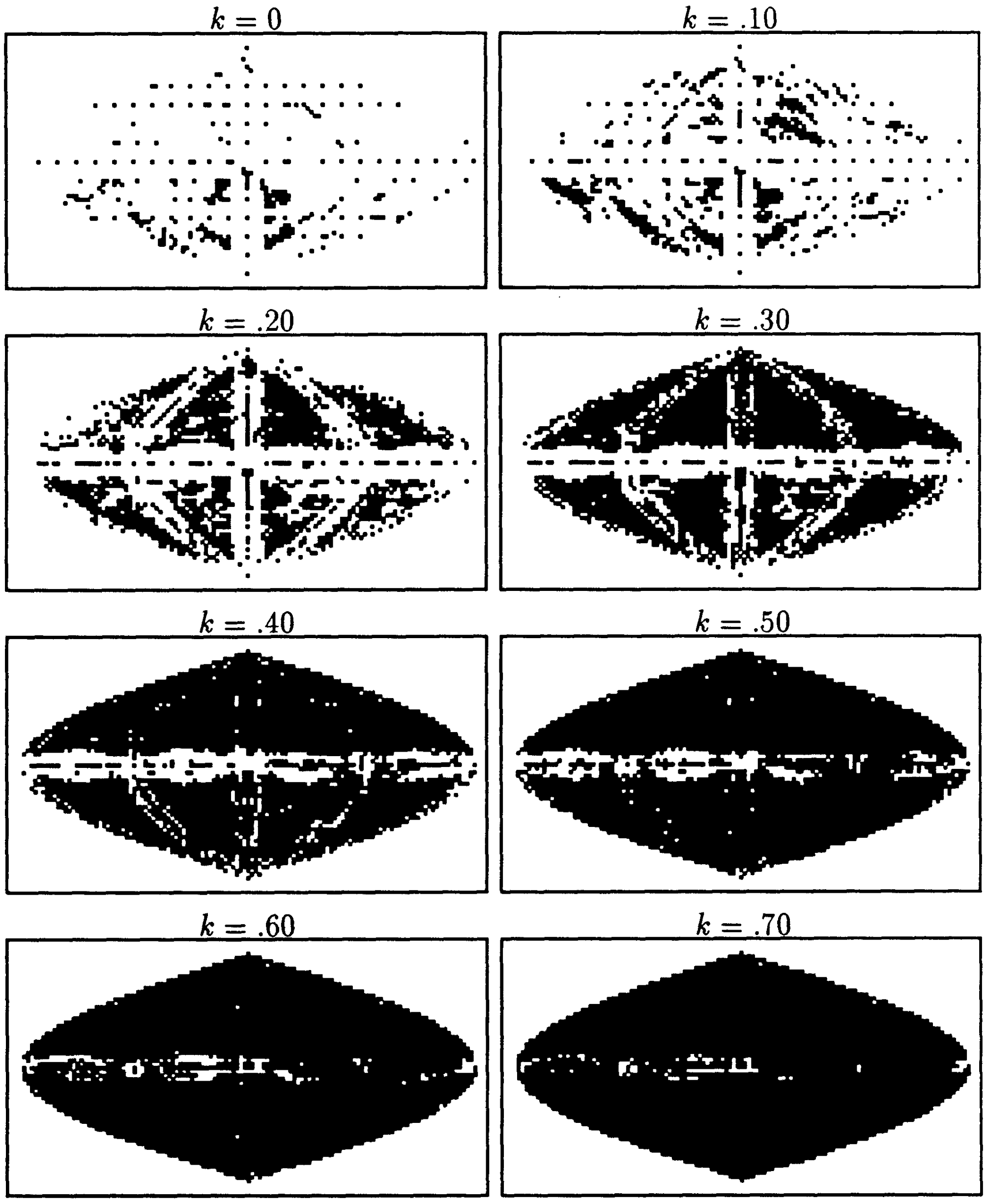


Figure 4-13: Captured viewpoint maps for two-curve view histograms (black dots indicate capture).

4.6 Examples

4.6.1 Implementation

In Chapter 1 much significance was placed on obtaining a recognition algorithm capable of considering known objects simultaneously, at least in its initial stages when objects dissimilar to observed objects are eliminated from further consideration. The indexing algorithm defined by inequality 4.1 realizes this goal because of its simplicity. It implies that, an unknown object view is matched to a sampled viewpoint if the relative distance between corresponding histograms is small enough. Since calculation of relative distance is a trivial operation, parallel implementation on fine-grain hardware is possible by assigning every sampled histogram to a distinct processor. For the examples which follow, parallel implementation employed 3008 processors of a model CM-2 Connection Machine.

4.6.2 Isolated Objects

We initially consider the case where single objects are observed. Our first example demonstrates the potential of indexing via high-capacity histograms. The view to be recognized is in Figure 4-14. It was observed from the viewing sphere of object 7 with focal length equivalent to that used during histogram sampling. When indexing based on single-curve histograms is attempted, 56 sampled viewpoints corresponding to 11 of 16 library objects are retrieved. In contrast, indexing based on two-curve histograms results in retrieval of only 2 views, both associated with the correct object as shown in Figure 4-15. Thus histogram capacity correlates nicely with indexing ambiguity, as expected.

Furthermore, indexing is fast as a result of parallel implementation, requiring for the two-curve case .05 seconds of Connection Machine processing. The corresponding time for serial consideration on a Symbolics model 3650 Lisp Machine is 5.5 seconds, for a two order of magnitude difference.

The next example demonstrates the robustness of qualitative description to per-

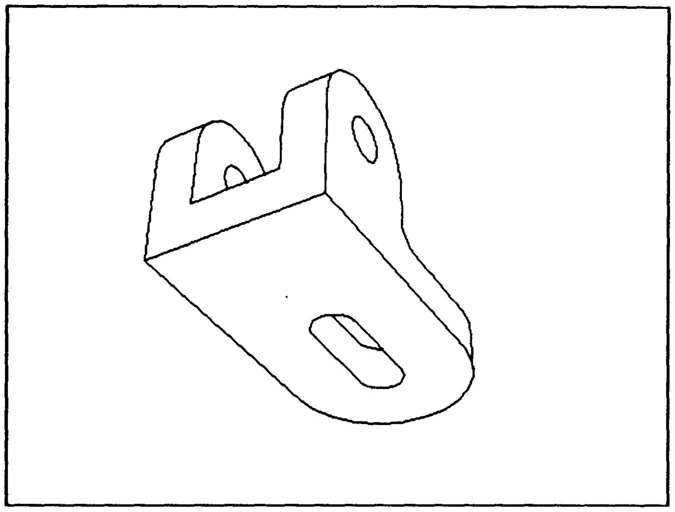


Figure 4-14: View of object 7 from its viewing sphere using a focal length of 50mm.

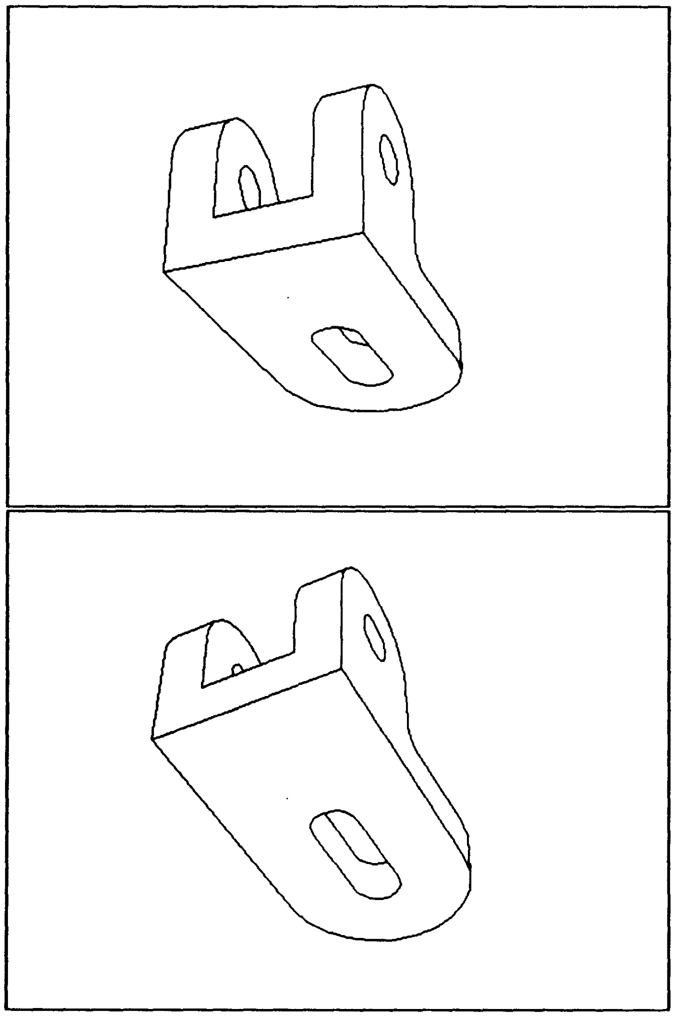


Figure 4-15: Results of two-curve histogram indexing for the view in Figure 4-14.

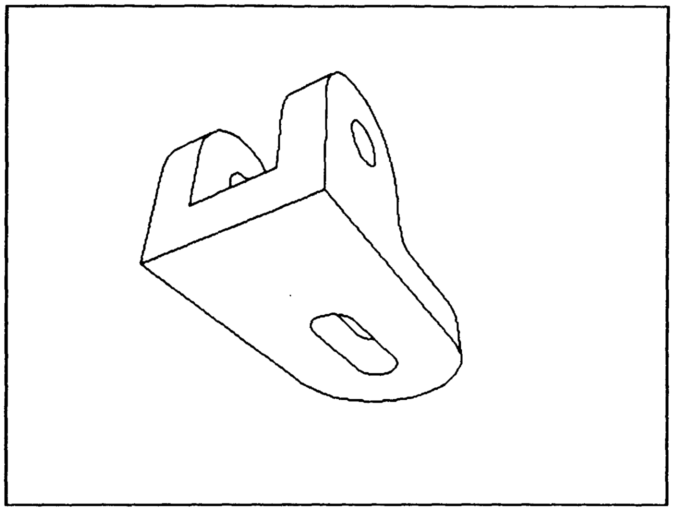


Figure 4-16: View of object 7 with focal length reduced from 50mm to 25mm.

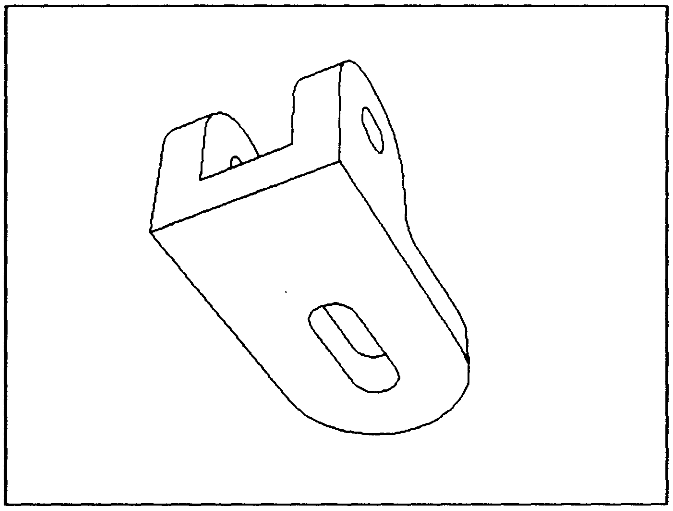


Figure 4-17: Result of two-curve indexing for view in Figure 4-16.

spective effects. We consider the object view used in the previous example with focal length reduced from 50mm to 25mm, as in Figure 4-16. In this case indexing based on single-curve histograms returns 74 sampled viewpoints, again corresponding to 11 objects. But two-curve indexing returns a single view of the correct object, pictured in Figure 4-17. The decrease in ambiguity is similar to that observed in the previous example. And halving the focal length compared to that used for histogram sampling does not seem to hinder performance.

The previous two examples are unusual in that indexing exhibits insignificant ambiguity. Typically a greater number of views are retrieved, often corresponding to

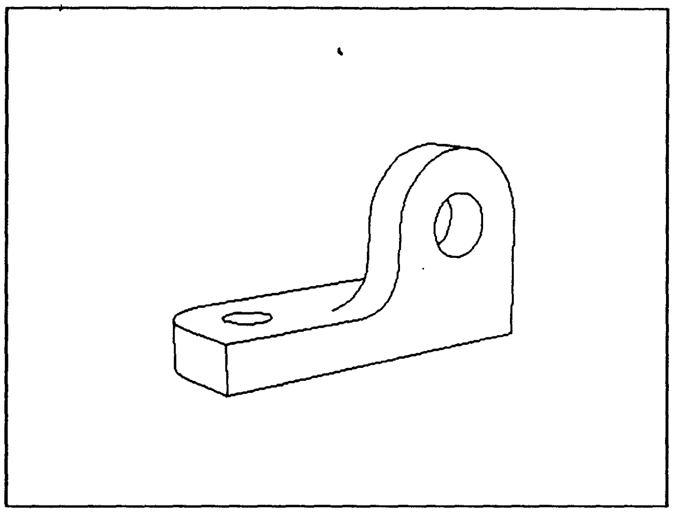


Figure 4-18: View of object 3 from viewing sphere with 50mm focal length.

more than one object. Consider for example the view of object 3 in Figure 4-18. In agreement with the previous examples single-curve indexing is ineffective, yielding 73 views of 11 objects. As expected, two-curve indexing performs much better, but this time with increased ambiguity. Specifically, 8 views of two objects are retrieved, as pictured in Figure 4-19. But the views retrieved of object 3 are quite similar to each other, and to the three views of of object 16 present. Thus ambiguity is increased, but since like views are retrieved, performance of indexing is as desired.

We now consider the behavior of indexing when object views are in the vicinity of accidental viewpoints. Figure 4-20 contains such a view of object 7. As discussed earlier we can not expect indexing to succeed here because tolerable distortion was set too low to capture all unstable viewpoints. In fact, two-curve indexing fails to retrieve any views of any objects.

4.6.3 Multiple Objects

So far we have considered only isolated objects. When multiple objects interact in the field of view, two approaches are possible. The first simply assumes bound p on the number of objects present, and without prior object segmentation attempts to match observed view histograms with all possible combinations of up to p sampled histograms. Because view histograms for scenes containing multiple isolated objects

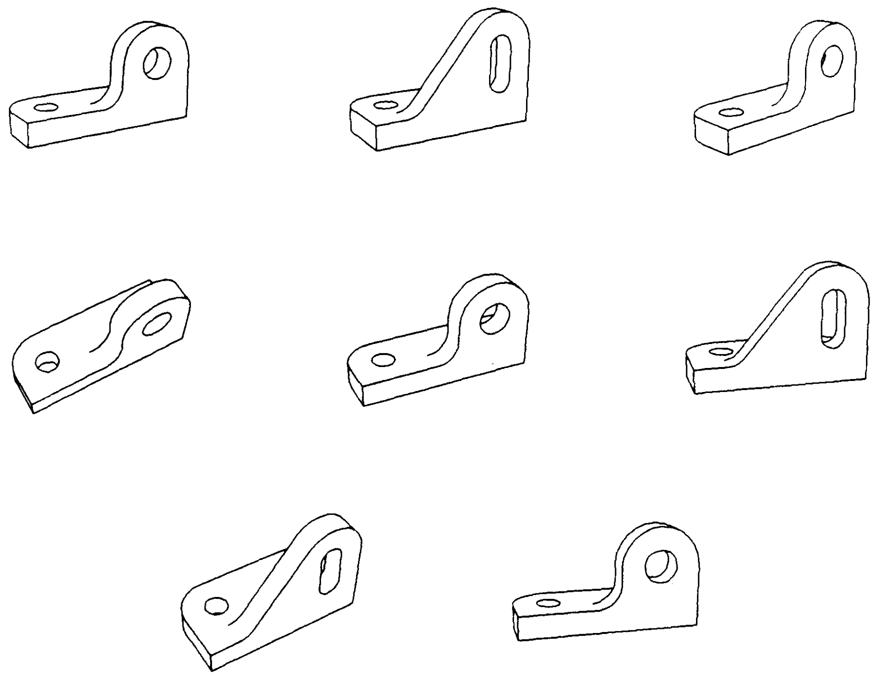


Figure 4-19: Results of two-curve histogram indexing for view in Figure 4-18.

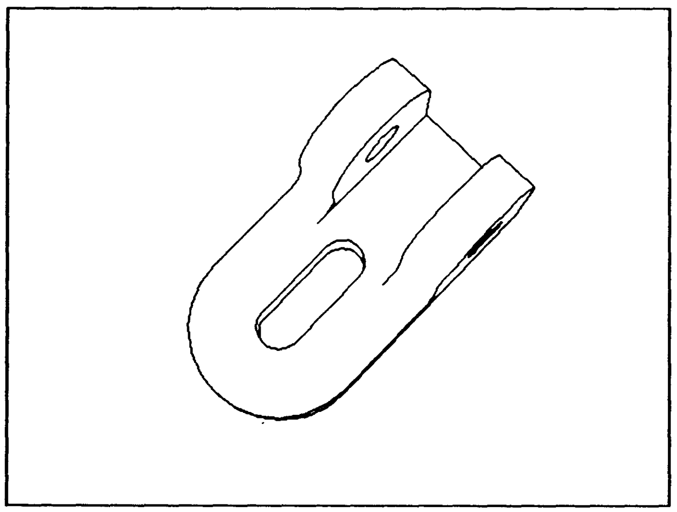


Figure 4-20: Nearly-accidental view of object 7 from viewing sphere with 50mm lens.

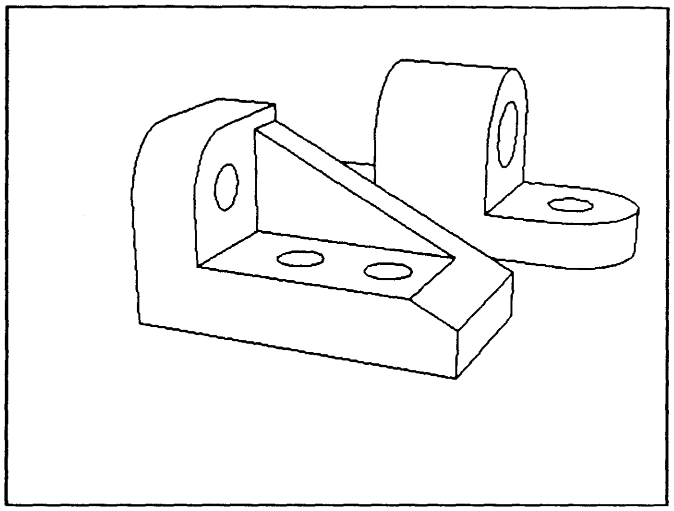


Figure 4-21: Object 8 partially occluding object 4.

are equivalent to the sum of histograms for individual objects, sampled histograms can be combined by summing, assuming distortion tolerance is large enough to handle the effects of occlusion. But combinatorially this approach is not very attractive. And because view histograms must be assigned to combinations of p sampled histograms with little ambiguity, histogram capacity must be much larger than the number of these combinations, namely the number of stored histograms raised to the pth power. Because of the huge increase in required capacity, this approach is not expected to work very well. As an example, consider the two-object view in Figure 4-21. With $p = 2$, the number of viewpoint pairs retrieved by two-curve indexing was in excess of 65,000.

Apparently, qualitative indexing requires prior segmentation of non-isolated objects. Thus the second approach assumes that segmentation has been attempted. The important issue then is the accuracy required of segmentation for successful indexing. To address this issue we consider three examples simulating successive degrees of segmentation error. First consider the view in Figure 4-22, which is assumed by the recognizer to contain only one object. Actually it contains two, with object 8 heavily occluding object 4. The small visible portion of object 4 simulates segmentation error. Figure 4-23 contains the results of two-curve indexing, indicating that distortion due to the minimal segmentation error in this case is handled nicely.

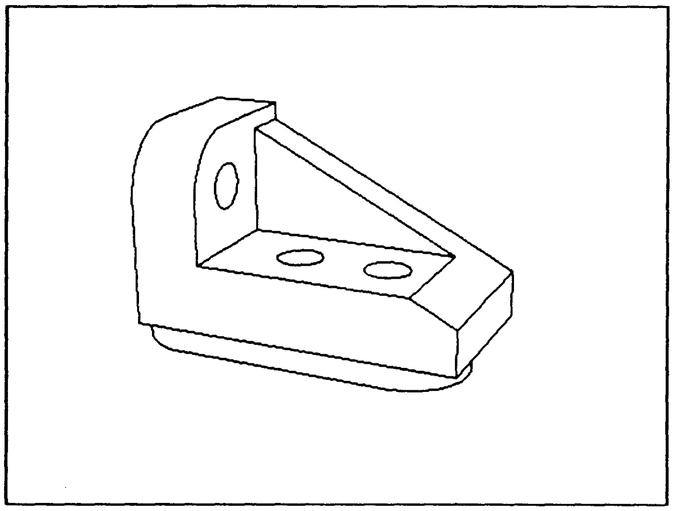


Figure 4-22: Segmentation error simulated by occluding object 4 with object 8.

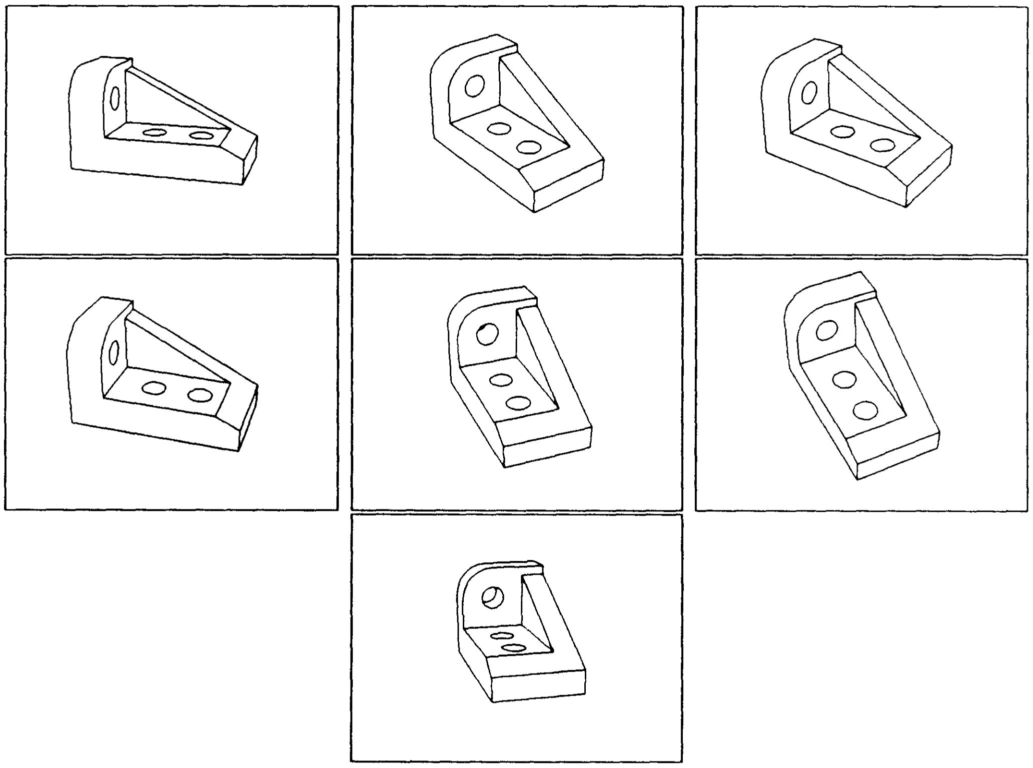


Figure 4-23: Results of two-curve indexing applied to drawing in Figure 4-22.

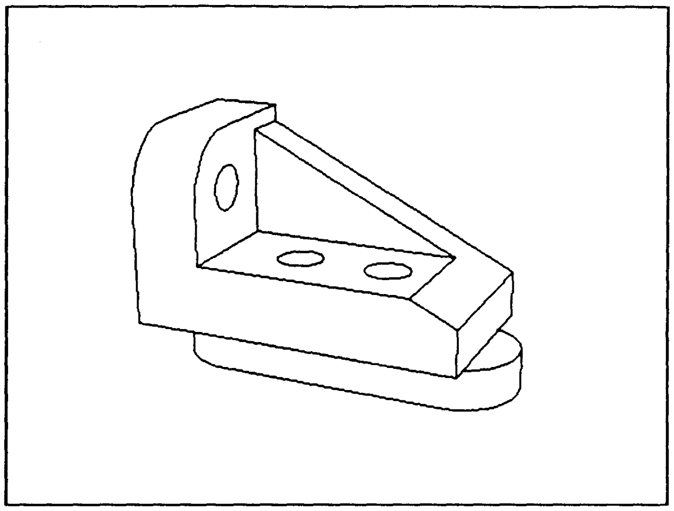


Figure 4-24: Increased segmentation error simulated by increased exposure of object 4.

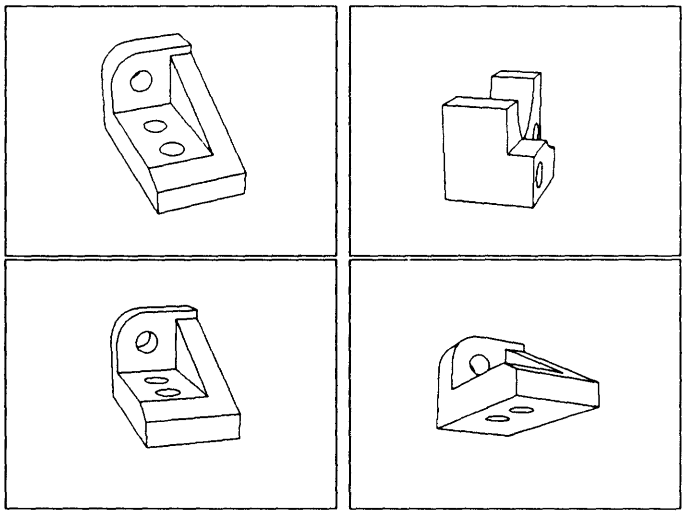


Figure 4-25: Results of two-curve indexing applied to drawing in Figure 4-24.

Now consider Figure 4-24, which represents a small increase in segmentation error. The viewpoints retrieved by indexing are presented in Figure 4-25. Note that increased distortion due to segmentation error results in significant viewpoint error in views corresponding to object 8.

Finally consider Figure 4-26, which represents increased segmentation error distortion primarily due to partial appearance of the hole in the base of object 4. Seven viewpoints were retrieved by two-curve indexing, only one corresponding to object 8. As pictured in Figure 4-27 this view is substantially different from the observed view. Thus the performance of indexing appears to decline rapidly with increased

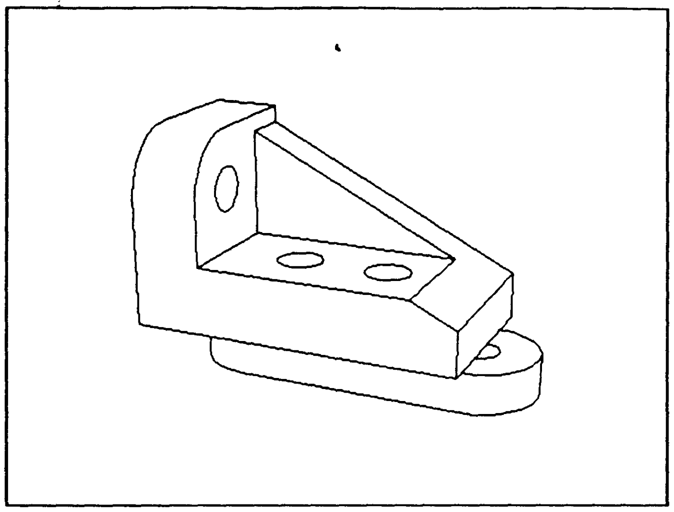


Figure 4-26: Appearance of hole in base of object 4 significantly increases distortion due to segmentation.

segmentation error.

One solution to this problem might be to simply increase distortion tolerance in order to retrieve more similar viewpoints. But it appears that the required tolerance is too large to result in acceptable indexing ambiguity, even for a moderate amount of segmentation error. For example, of the 72 stored histograms closest to the view in Figure 4-26, only the view cited belongs to object 8.

Obviously distortion can not be allowed to grow to the point where capacity becomes insufficient. But the big problem here is that view features are quite sensitive to segmentation error, particularly because extraction of object curves is disrupted by intersection of spurious curves.

Thus if view features insensitive to segmentation error cannot be found, success of qualitative indexing depends on reliable object segmentation. This perhaps is the major obstacle regarding application of indexing to real images of cluttered scenes, one frankly that does not exist for recognition techniques using object-centered models. In contrast, these techniques produce scene segmentation instead of requiring it.

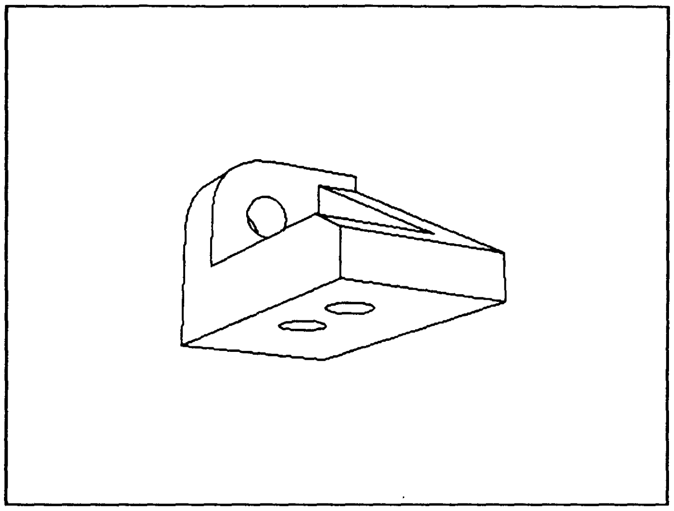


Figure 4-27: Two-curve indexing applied to Figure 4-26 retrieves only one view of object 8.

4.6.4 Additional Information

Significant ambiguity can sometimes result in spite of large histogram capacity. For example, two-curve indexing applied to the view of object 14 in Figure 4-28 results in retrieval of 24 sampled viewpoints, with 13 corresponding to object 14, 7 to object 13, and 4 to object 9. Thus it appears that the view histogram for Figure 4-28 resides in a rather congested region of histogram space. A possible solution is to employ a second indexing step based on alternate information to further filter retrieved viewpoints.

One source of information yet to be exploited is curve length[6]. In order to be used as part of a qualitative description scheme it must be quantized to yield a finite number of curve descriptions. Furthermore, it must be normalized somehow to provide scale independence. If we are interested in qualitative description of curve pairs, then these two requirements are easily satisfied.

Assuming line drawing curves are paired as described above, qualitative description based on length can be obtained by replacing concavity counts c_1 and c_2 with qualitative length descriptions d_1 and d_2 obtained as follows. First let $L_{\max}$ indicate the maximum of curve lengths L_1 and L_2. Then division by $L_{\max}$ normalizes L_1 and

[6]Here lengths of digitized curves were obtained by summing lengths of segments obtained via recursive linear approximation [2, pages 234–235].

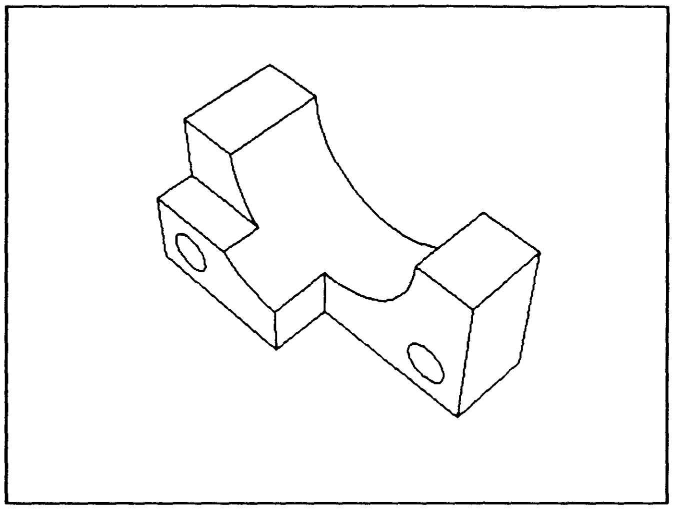


Figure 4-28: View of object 14 from viewing sphere with 50mm lens.

L_2. Assuming q quantization intervals, the quantization interval in which normalized lengths reside can be determined by rounding up the product of q with each. Thus normalized and quantized length description d_i for curve i is obtained as follows:

$$d_i = \left\lceil q \frac{L_i}{L_{\max}} \right\rceil \tag{4.4}$$

In order to successfully apply this manner of qualitative pair description to the indexing problem, a value for q yielding large histogram capacity must be determined. Here $q = 4$ is used since it can be shown to yield an upper bound on capacity of the same order of magnitude as that obtained from the previously considered pair classification scheme.

To do so, first consider histogram bin entropies. From the plot in Figure 4-29, we see that all 21 bins have entropy greater than 1 bit (category entropies are listed in Table 4.3). Thus we set $n = 21$. Now consider the captured viewpoint maps in Figure 4-30. It appears that views away from accidental viewpoints are captured by $k = .50$. Thus our upper bound on capacity in this case is $\widehat{C}_{21}(2.0) = 1.8 \times 10^9$, compared to 7.5×10^8 for the previous pair classification scheme.

In applying our new pair classification scheme to the view in Figure 4-28, indexing results in ambiguity similar to that found for classification based on concavity count.

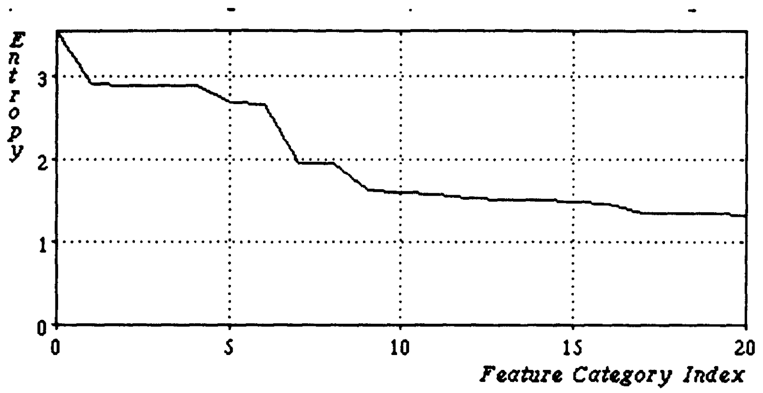


Figure 4-29: Bin entropies (in bits) for qualitative length classification of curve pairs.

Category Index	d_1	d_2	θ	Entropy (bits)
0	4	4	$< \pi$	3.566
1	1	4	$< \pi$	2.908
2	4	1	$< \pi$	2.905
3	2	4	$< \pi$	2.896
4	4	2	$< \pi$	2.889
5	3	4	$< \pi$	2.691
6	4	3	$< \pi$	2.671
7	4	1	$\sim \pi$	1.946
8	1	4	$\sim \pi$	1.943
9	4	4	$\sim \pi$	1.631
10	4	4	$> \pi$	1.608
11	2	4	$\sim \pi$	1.564
12	4	2	$\sim \pi$	1.526
13	4	1	$> \pi$	1.506
14	4	2	$> \pi$	1.502
15	2	4	$> \pi$	1.484
16	1	4	$> \pi$	1.449
17	3	4	$> \pi$	1.354
18	3	4	$\sim \pi$	1.337
19	4	3	$\sim \pi$	1.335
20	4	3	$> \pi$	1.321

Table 4.3: Entropies for length-classified curve pairs.

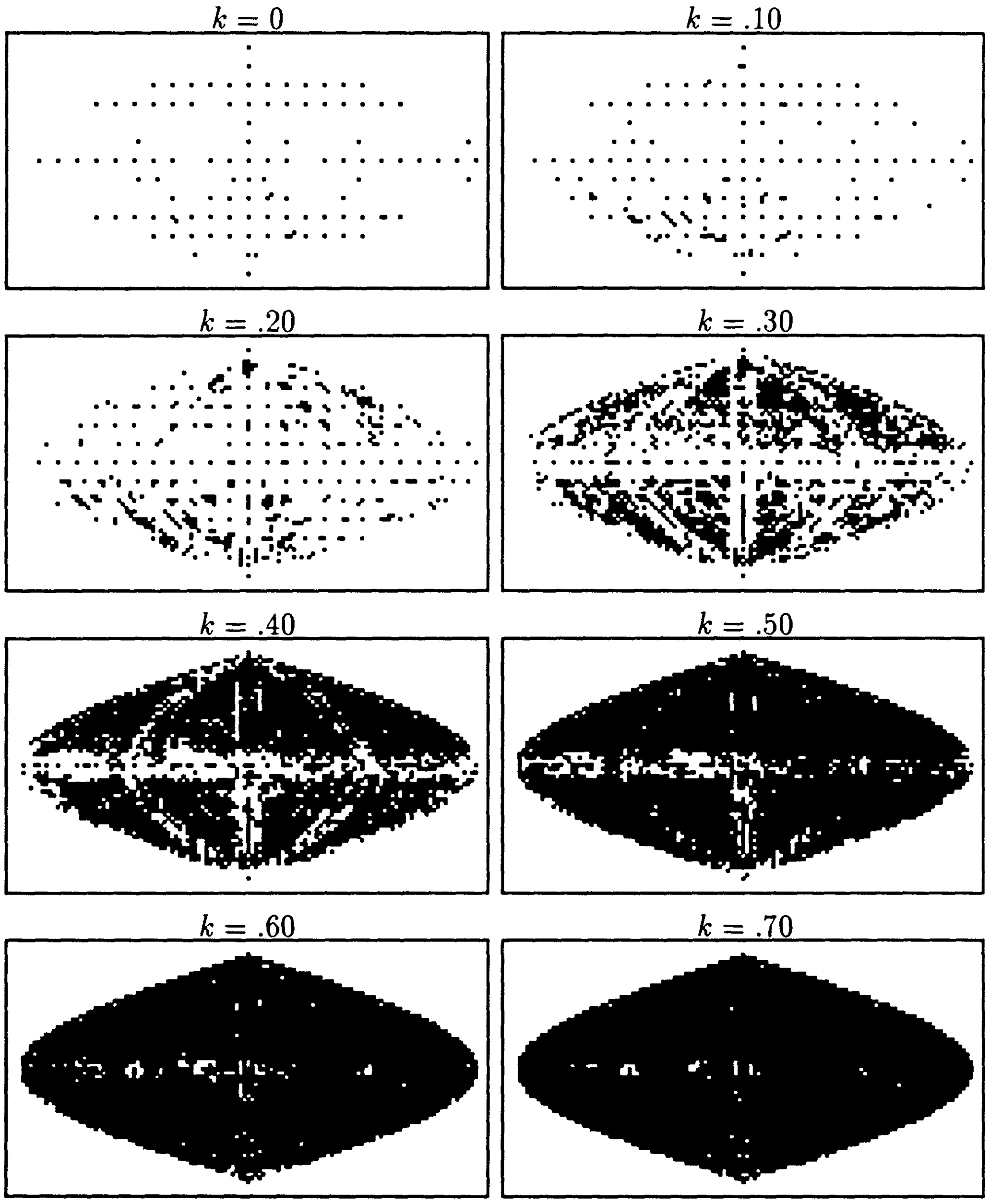


Figure 4-30: Captured viewpoint maps for view histograms based on qualitative length description of curve pairs. (black dots indicate capture).

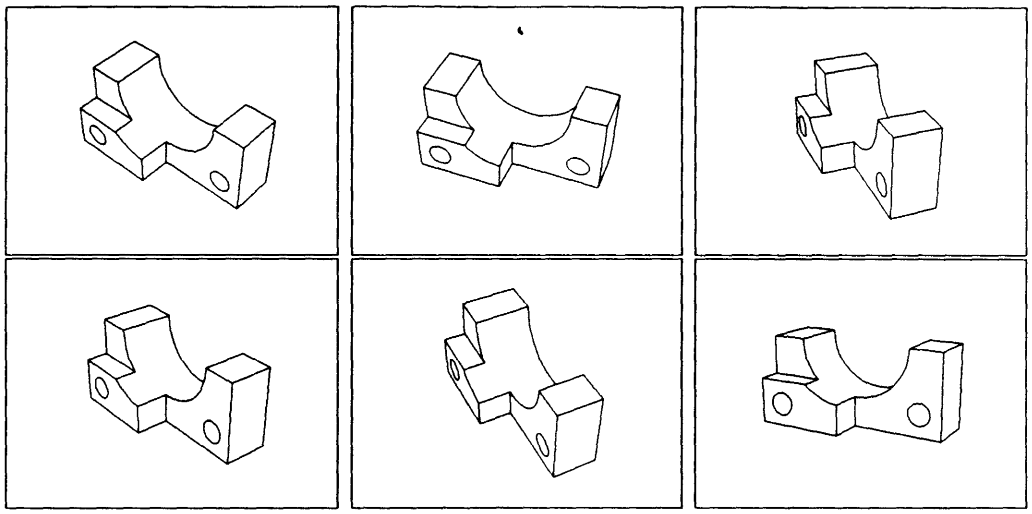


Figure 4-31: Results of two-stage indexing for view in Figure 4-28 using pair descriptions based on curvature and length.

Specifically, 15 view samples are retrieved, of which 7 correspond to object 14, 4 to object 15, 3 to object 9, and 1 to object 11. But what really is of interest here is the intersection of the views retrieved for the two classification schemes. These views are presented in Figure 4-31, indicating no remaining ambiguity regarding object identity. Thus we see that two indexing steps, one applied to the results of the other, can be used to reduce ambiguity to an acceptable level.

4.7 Conclusion

We have seen that qualitative information when properly encoded results in effective indexing of the objects in Figure 1-4. In particular, view descriptions encoding only concavity counts and three-way angular categorizations (less than π, possibly π, or greater than π) yielded good results, which might seem surprising since this information is quite coarse.

That such simple information is so effective in distinguishing objects (and viewpoints) is explained via consideration of representational capacity, meaning the maximum number of objects unambiguously representable in the presence of distortion. Assuming capacity is much larger than the number of items to be distinguished, effective indexing can be expected.

In considering capacity, the dependence of representation on number of objects to be distinguished becomes explicit. Since capacity increases with number of feature categories, it is not prudent to specify object representations without regard for the number of objects to be recognized because simpler representations involving fewer feature categories could be sufficient.

In our case capacity was not sufficient initially because qualitative characterization of line drawing curves yielded too few categories. In order to increase the number of feature categories in a manner which preserved the coarseness of extracted information, grouping was employed, demonstrating its usefulness as a mechanism for increasing representational capacity[7]. In particular, curves were grouped into pairs, yielding much higher capacity than provided by single-curve features. As expected, indexing using curve pairs was far superior, in general resulting in little ambiguity.

However, large capacity is not a guarantee against unacceptable ambiguity. When this occurs, several indexing steps can be cascaded to avoid time-consuming verification of a large number of views. This technique was demonstrated by considering an example where indexing based on curvature description of curve pairs yielded unsatisfactory results. As desired, a second indexing step based on length description of pairs resulted in unique identification.

Consideration of scenes involving multiple objects suggested that object segmentation is required by qualitative indexing. Furthermore, it appears that segmentation must be quite accurate in order to preserve histogram capacity. Thus application of qualitative indexing to real images is conditional upon the development of robust object segmentation techniques.

[7]Previous work has employed grouping, but typically as a pruning technique for the feature correspondence problem [49, 45].

Chapter 5

Conclusion

A review of the previous chapters is given, followed by elaboration of some of the more important issues addressed.

5.1 Review

The use of qualitative information in visual object recognition is attractive for several reasons. Most important is the ability to capture qualitative appearance in a finite number of views. With no dimensional mismatch between image and model data, the computationally intensive task of converting to a common reference frame for matching is eliminated, allowing recognition to proceed with reduced computational effort. Learning is similarly simplified, since qualitative object representations consist of collections of view descriptions directly encoding object appearance.

But because qualitative characterization can fail to capture fine distinctions among objects, unique identification can not be guaranteed. Instead the best we can hope for is that objects are *indexed* by qualitative information, allowing retrieval of a small number of similar objects for further consideration by schemes employing more precise information.

Thus a general framework for recognition inspired by consideration of qualitative information was proposed. Instead of applying computationally expensive detection algorithms to library objects uniformly, a two stage algorithm was suggested involving initial application of fast and cheap indexing, followed by more detailed and therefore expensive verification of remaining objects.

In this framework it is obviously of great importance for indexing to yield very selective results, so that only a few objects need to be considered for verification. Thus the primary issue regarding qualitative representation is ability to distinguish objects of interest. This issue was addressed in Chapter 3 via consideration of representational capacity, defined as the maximum number of objects unambiguously representable in the presence of view distortion. Assuming capacity is much larger than the number of items to be distinguished, effective indexing can be expected.

Determining an exact value for capacity is difficult, so an upper bound on capacity as a function of expected view distortion and number of qualitative feature categories was derived. As expected, this result indicates capacity decreases with increasing distortion. It also suggests that capacity increases with number of feature categories.

In applying these ideas to the objects in Figure 1-4, the initial features considered were line drawing curves segmented at junctions. In Chapter 2 qualitative characterization of digitized curves was considered, resulting in derivation of an algorithm for reliable extraction of concavity count in spite of orientation uncertainty due to spatial quantization.

Resulting view descriptions were considered in Chapter 4, where capacity evaluation yielded an insufficient value. To increase capacity, grouping was used to increase the number of feature categories in a manner which preserved feature classification coarseness. In particular it was demonstrated that indexing via curve pairs is much more effective than via single curves. Typically, indexing of the object set in Figure 1-4 in this manner yielded only a few sampled viewpoints corresponding to one or just a few objects.

However, performance was not uniformly acceptable. In spite of high representational capacity, it is possible for indexing to result in significant ambiguity. Instead of proceeding immediately to intensive verification, additional indexing steps using alternate sources of information can be used to achieve desired selectivity. This was demonstrated by cascading two indexing steps, one corresponding to qualitative curvature description, the other to qualitative length description. Indexing in this case can be thought of as a sequence of filters, each using different information to exclude objects from further consideration, in a sense reducing ambiguity through "process of elimination."

The results cited above concern isolated scene objects. When multiple objects are present matters become complicated, especially when occlusion is present. In this case it was concluded that qualitative indexing requires object segmentation. And if view features are sensitive to segmentation error, as was the case with the line drawings considered here, segmentation must be accurate and reliable. This perhaps indicates the primary difficulty in applying qualitative indexing to real images.

5.2 Representation

The core of this work is Chapter 3, where recognition is viewed as a communication problem. By interpreting view descriptions as codes for object identity, object representation becomes a function of the number of objects to be recognized. Thus the representation problem is moved from the domain of philosophical speculation where the *type* and *structure* of information that *should* be encoded is of concern, to a more pragmatic plane where any information is acceptable as long as it results in sufficient representational capacity.

This of course does not represent revolutionary thought. In practice one is much more likely to find representations encoding coarse information such as number of holes, rather than the precise representations currently in vogue in the literature. But the primary contribution here has been to formalize the analysis of qualitative information in an attempt to predict its effectiveness in distinguishing objects of interest. The major payoff was the correct but surprising suggestion made by capacity analysis that coarse description of curve pairs results in view descriptions capable not only of distinguishing objects, but object viewpoints as well.

5.3 Learning

With few exceptions (eg. [15]), the object learning problem has been neglected in the computer vision literature. To my knowledge, current recognition systems are not capable of generating representations of realistic objects from input imagery. More specifically, they are not able to "learn from experience" [3], meaning that if presented with an unknown object, they are not able to extract enough information to recognize it if seen again from a similar viewpoint. Instead these systems depend on models provided by humans.

For systems employing 3D object-centered models, the problem is directly attributable to difficulties associated with forming object-centered representations from viewer-centered data. No such difficulties exist for qualitative representations, which consist of collections of view descriptions. "Learning by experience" is thus trivially enabled, allowing representations for the objects in Figure 1-4 to be obtained simply by describing appearance at various viewpoints.

5.4 Generality

The results presented here regarding application of coarse information to the object recognition problem apply to any feature class that can be qualitatively categorized. For illustrative purposes line drawings were examined, with features consisting of extracted curves classified primarily by concavity count and categorized angular separation. But the analysis in Chapter 3 regarding capacity of qualitative view descriptions does not exclude other object attributes, for example color, surface shape, or texture. All that is required is qualitative description of feature classes of interest. Then capacity can be examined and manipulated in exactly the manner discussed in Chapter 4.

5.5 Further Work

In Chapter 4 an example considering the effects of imperfect object segmentation was presented with the result that segmentation error, small in terms of image area, significantly disrupted qualitative indexing. The reason was that introduction of spurious junctions caused faulty extraction and description of view features. Sensitivity with respect to segmentation error therefore is expected since introduction of false junctions is a phenomenon independent of segmentation error area.

This represents a serious problem for qualitative indexing. Since qualitative representation depends on categorization of discrete features, any disruption of feature segmentation is bound to have a serious if not catastrophic effect. But it is probably worth overcoming since the results presented for isolated objects indicate that indexing is potentially an extremely valuable tool for increasing recognition speed and effectiveness.

In considering solutions to the sensitivity problem several options exist. One is to attempt to isolate segmentation error by performing hierarchical indexing. That is, parts identified by indexing on primal features could be used to index into the object library. In so doing, a certain amount of part misidentification could be tolerated, thus compensating for localized segmentation error. But this approach requires a repeatable procedure for extracting parts, representing a significant drawback.

Another approach is to use features of small spatial extent, thus reducing the chance that any one would be affected by segmentation error. For example, curves could be divided at regularly spaced intervals, resulting in smaller curves less likely to contain spurious junctions.

Finally, we might simply consider trying to find feature classes such that spurious features are not created by segmentation error. One possibility is to consider surface patches, which do not suffer from the same difficulties as curves.

Consideration of these sensitivity-reduction techniques, as well as others if they exist, is required to avoid reliance on extremely reliable segmentation techniques which currently do not exist. Thus their development can be considered a prerequisite for practical application of the indexing scheme proposed in this work.

Bibliography

[1] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111-122, 1981.

[2] D. H. Ballard and C. M. Brown, *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

[3] P. J. Besl and R. C. Jain, "Three-dimensional object recognition," *Computing Surveys*, vol. 17, no. 1, pp. 75-145, March 1985.

[4] B. Bhanu and J. C. Ming, "TRIPLE: A multi-strategy machine learning approach to target recognition," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 537-547.

[5] I. Biederman, "Human image understanding: recent research and a theory," *Computer Vision, Graphics, and Image Processing*, vol. 32, pp. 29-73, 1985.

[6] T. O. Binford, "Survey of model-based image analysis systems," *International Journal of Robotics Research*, vol. 1, no. 1, pp. 18-64, Spring 1982.

[7] R. A. Brooks, "Symbolic reasoning among 3-D models and 2-D images," *Artificial Intelligence*, vol. 17, pp. 285-348, 1981.

[8] J. B. Burns and L. J. Kitchen, "Rapid object recognition from a large model base using prediction hierarchies," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 711-719.

[9] T. A. Cass, "Parallel Computation in Model-Based Recognition," S.M. thesis, Massachusetts Institute of Technology, May 1988.

[10] I. Chakravarty, "A generalized line and junction labeling scheme with applications to scene analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, no. 2, pp. 202-205, Apr. 1979.

[11] I. Chakravarty and H. Freeman, "Characteristic views as a basis for three-dimensional object recognition," in *SPIE Conference on Robot Vision*, Arlington, VA, May 1982, pp. 37-45.

[12] C. H. Chen and A. C. Kak, "A robot vision system for recognizing 3-D objects in low-order polynomial time," to be published in *IEEE Trans. SMC.*

[13] R. T. Chin and C. R. Dyer, "Model-based recognition in robot vision," *Computing Surveys*, vol. 18, no. 1, pp. 67-108, March 1986.

[14] M. B. Clowes, "On seeing things," *Artificial Intelligence*, vol. 2, pp. 79-116, 1971.

[15] J. H. Connell, "Learning Shape Descriptions: Generating and Generalizing Models of Visual Objects," S.M. thesis, Massachusetts Institute of Technology, 1985.

[16] C. I. Connolly et al., "Matching from 3D range models into 2D intensity scenes," in *Proc. First International Conference on Computer Vision*, London, England, June 1987.

[17] C. I. Connolly and J. R. Stenstrom, "Generation of face-edge-vertex models directly from images," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 1041-1046.

[18] B. M. Dawson and G. Treese, "Computing curvature from images," in *SPIE Vol. 504–Applications of Digital Image Processing VII*, 1984, pp. 175-182.

[19] M. Dhome and T. Kasvand, "Polyhedra recognition by hypothesis accumulation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 3, pp. 429-438, May 1987.

[20] S. Edelman, H. Bülthoff, and D. Weinshall, "Stimulus familiarity determines recognition strategy for novel 3D objects," A. I. Memo #1138, Massachusetts Institute of Technology, July 1989.

[21] J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics.* Reading, MA: Addison-Wesley, 1984.

[22] K. S. Fu, *Sequential Methods in Pattern Recognition and Machine Learning.* New York: Academic Press, 1968.

[23] R. G. Gallager, *Information Theory and Reliable Communication.* New York: Wiley, 1968.

[24] F. E. Giesecke et al., *Engineering Graphics.* New York: Macmillan, 1981.

[25] C. Goad, "Fast 3D model-based vision," in *Recent Advances in Computational and Robotic Vision*, A. P. Pentland, Ed. Norwood, NJ: Ablex, 1986.

[26] W. E. L. Grimson, "Sensing strategies for disambiguating among multiple objects in known poses," A. I. Memo #855, Massachusetts Institute of Technology, Aug. 1985.

[27] W. E. L. Grimson, "The combinatorics of local constraints in model-based recognition and localization from sparse data," A. I. Memo #763A, Massachusetts Institute of Technology, Mar. 1986.

[28] W. E. L. Grimson and D. P. Huttenlocher, "On the sensitivity of the Hough transform for object recognition," A. I. Memo #1044, Massachusetts Institute of Technology, May 1988.

[29] W. E. L. Grimson and T. Lozano-Pérez, "Model-based recognition and localization from sparse range or tactile data," *The International Journal of Robotics Research*, vol. 3, no. 3, pp. 3-35, Fall 1984.

[30] W. E. L. Grimson and T. Lozano-Pérez, "Recognition and localization of overlapping parts from sparse data," A. I. Memo #841, Massachusetts Institute of Technology, June 1985.

[31] A. Guzman, "Decomposition of a visual scene into three-dimensional bodies," in *Proc. AFIPS Joint Computer Conference*, Fall 1968, pp. 291-304.

[32] D. Harwood, R. Prasannappa, and L. Davis, "Preliminary design of a programmed picture logic," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 745-755.

[33] M. Hebert and T. Kanade, "3-D vision for outdoor navigation by an autonomous vehicle," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 593-601.

[34] D. D. Hoffman and W. A. Richards, "Representing smooth plane curves for recognition: implications for figure-ground reversal," in *Proceedings of the National Conference on Artificial Intelligence*, Pittsburgh, PA, Aug. 1982, pp. 5-8.

[35] R. Horaud, "New methods for matching 3-D objects with single perspective views," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-9, no. 3, pp. 401-412, May 1987.

[36] D. A. Huffman, "Impossible objects as nonsense sentences," *Machine Intelligence*, vol. 6, pp. 295-323, 1971.

[37] D. P. Huttenlocher, "Three-dimensional recognition of solid objects from a two-dimensional image," Ph.D. thesis, Massachusetts Institute of Technology, Apr. 1988.

[38] D. P. Huttenlocher and S. Ullman, "Object recognition using alignment," in *Proc. Image Understanding Workshop*, Los Angeles, CA, Feb. 1987, pp. 370-380.

[39] K. Ikeuchi and T. Kanade, "Modeling sensors and applying sensor model to automatic generation of object recognition program," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 697-710.

[40] D. W. Jacobs, "GROPER: A grouping based recognition system for two dimensional objects," in *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, Miami Beach, FLA, Nov. 1987, pp. 164-169.

[41] A. Kalvin et al., "Two-dimensional, model-based, boundary matching using footprints," *The International Journal of Robotics Research*, vol. 5, no. 4, pp. 38-55, Winter 1986.

[42] J. J. Koenderink and A. J. van Doorn, "The singularities of the visual mapping," *Biological Cybernetics*, vol. 24, pp. 51-59, 1976.

[43] J. J. Koenderink and A. J. van Doorn, "The internal representation of solid shape with respect to vision," *Biological Cybernetics*, vol. 32, pp. 211-216, 1979.

[44] S. J. Lee, R. M. Haralick, and M. C. Zhang, "Understanding objects with curved surfaces from a single perspective view of boundaries," *Artificial Intelligence*, vol. 26, pp. 145-169, 1985.

[45] D. G. Lowe, *Perceptual Organization and Visual Recognition.* Hingham, MA: Kulwer, 1985.

[46] J. Malik, "Interpreting line drawings of curved objects," Ph.D. thesis, Stanford University, 1985.

[47] D. Marr, *Vision.* San Francisco: Freeman, 1982.

[48] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," *The Psychological Review*, vol. 63, no. 2, pp. 81-97, March 1956.

[49] R. Mohan and R. Nevatia, "Perceptual grouping for the detection and description of structures in aerial images," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 512-526.

[50] J. L. Mundy, A. J. Heller, and D. W. Thompson, "The concept of an effective viewpoint," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 651-659.

[51] J. R. Pierce, *An Introduction to Information Theory.* New York: Dover, 1980.

[52] T. Poggio, personal communication, Aug. 1989.

[53] K. Rangarajan, M. Shah, and D. Van Brackle, "Optimal corner detector," in *Proceedings of the Second International Conference on Computer Vision*, Tampa, FLA, Dec. 1988, pp. 90-93.

[54] W. Richards, "How to play twenty questions with nature and win," A. I. Memo #660, Massachusetts Institute of Technology, Dec. 1982.

[55] W. Richards, B. Dawson, and D. Whittington, "Encoding contour shape by curvature extrema," *Journal of the Optical Society of America*, vol. 3, no. 9, pp. 1483-1491, Sept. 1986.

[56] L. G. Roberts, "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, Tippett et al., Eds. Cambridge, MA: MIT Press, 1965.

[57] J. T. Schwartz and M. Sharir, "Identification of partially obscured objects in two and three dimensions by matching of noisy characteristic curves," Robotics Report #46, New York University, 1985.

[58] Y. Shirai, *Three-Dimensional Computer Vision.* Berlin: Springer-Verlag, 1987.

[59] Y. Shirai, "Recognition of real-world objects using edge cues," in *Computer Vision Systems*, A. R. Hanson and E. M. Riseman Eds. New York: Academic Press, 1978.

[60] T. M. Silberberg, "Multiresolution aerial image interpretation," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 505-511.

[61] G. Strang, *Linear Algebra and Its Applications.* New York: Academic Press, 1980.

[62] K. Sugihara, *Machine Interpretation of Line Drawings.* Cambridge, MA: MIT Press, 1986.

[63] M. Swain, "Object recognition from a large database using a decision tree," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 690-696.

[64] Symbolics, Inc., *S-Geometry* reference manual. Apr. 1988.

[65] D. W. Thompson and J. L. Mundy, "Model-directed object recognition on the Connection Machine," in *Proc. Image Understanding Workshop*, Los Angeles, CA, Feb. 1987, pp. 98-106.

[66] K. Trustrum, *Linear Programming.* London: Routledge & Kegan Paul Ltd., 1971.

[67] S. Ullman, "Visual routines," *Cognition*, vol. 18, pp. 97-159, 1983.

[68] S. Ullman and R. Basri, "The alignment of objects with smooth surfaces," A. I. Memo #1060, Massachusetts Institute of Technology, July 1988.

[69] D. Walters, "Selection of image primitives for general-purpose visual processing," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 261-298, 1987.

[70] D. L. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, P. H. Winston Ed. New York: McGraw-Hill, 1975.

[71] Q. R. Wang and C. Y. Suen, "Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, no. 4, pp. 406-417, July 1984.

[72] D. Weinshall, "Qualitative vs. quantitative depth and shape from stereo," in *Proc. Image Understanding Workshop*, Cambridge, MA, Apr. 1988, pp. 779-785.