Pyramid Vision Using Key Features to Integrate Image-Driven Bottom-Up and Model-Driven Top-Down Processes

by

Ze-Nian Li and Leonard Uhr

Computer Sciences Technical Report #678

December 1986

Pyramid Vision Using Key Features to Integrate Image-Driven Bottom-Up and Model-Driven Top-Down Processes

Ze-Nian Li
Department of Computer Sciences
University of Wisconsin–Milwaukee

Leonard Uhr
Department of Computer Sciences
University of Wisconsin–Madison

Abstract

Pyramid-like parallel hierarchical structures have been shown to be suitable for many computer vision tasks, and to have the potential for achieving the speeds needed for the real-time processing of real-world images. We are developing algorithms to explore the pyramid's massively parallel and shallowly serial-hierarchical computing ability in an integrated system that combines both low level and higher level vision tasks.

Micro-modular transforms are used to embody the program's knowledge of the different objects it must recognize. This paper describes pyramid vision programs that, starting with the image, use transforms that assess key features to dynamically imply other feature-detecting and characterizing transforms, and additional top-down, model-driven processes to apply. Program performance is presented for four real-world images of buildings.

The use of key features in pyramid vision programs and the related search and control issues are discussed. To expedite the detection of various key features, feature-adaptable windows are developed. In addition to image-driven bottom-up and model-driven top-down processing, lateral search is used, and is shown to be helpful, efficient, and feasible. The results indicate that, with the use of key features and the combination of a variety of powerful search patterns, the pyramid-like structure is effective and efficient for supporting parallel and hierarchical object recognition algorithms.

This research was partially supported by NSF Grant DCR-8302397 to L. Uhr. To appear in *Systems, Man and Cybernetics*, in press.

1. Introduction

The importance of highly parallel but also hierarchical descriptions of image data has been emphasized by many computer vision researchers. In one of their early papers [15], Marr and Nishihara illustrated a good hierarchical organization of shape information in a 3-D model description. In recent years, neuroanatomists and neurophysiologists have provided extensive evidence demonstrating the hierarchical processing of sensory information in the visual cortex of animals [6, 7, 9]. The *recognition cone* proposed by Uhr [29, 30] loosely models human and animal vision systems in the form of a layered hierarchical structure. Features for object recognition are extracted and aggregated at levels of increasing abstraction, globality, and generality, and decreasing resolution and detail. In addition to its hierarchical nature, the recognition cone also incorporates the massive parallelism of local array processes which is another characteristic of human and animal vision systems. This kind of parallelism can be realized by micro-modular parallel processors; hence it is well suited for fast and efficient VLSI multi-computers.

The *pyramid* structure introduced by Tanimoto and Pavlidis [27] incorporates the same multi-resolution and hierarchical representation of image data. It has been shown [4, 8, 17, 23, 26, 28] that, using local windows and parallel computation, pyramid-like structures are very efficient for low level image processing, e.g., averaging, histograming, edge detection, median filtering, image segmentation. Several programs [8, 19, 24, 25, 30] have also used pyramid-like structures for object recognition. Pyramid programs are, potentially, extremely fast when executed by appropriate pyramid-of-arrays multi-computers.

Various observations have presented evidence of behavior and mechanisms for 'focus of attention' in human vision [16, 34]. Since a picture that is presented using only a one millisecond flash of light is perfectly recognizable, and recognition takes only a few hundred milliseconds, there is no time for glancing around or focusing attention. But the way people examine images when they are presented for longer periods is suggestive of

how processing might be organized in briefer periods as well. Human perceivers usually do not stare at the entire area with the same intensity. Rather, after a rough scan they focus on the 'interesting' portion of a scene. This is one of the factors which make the human vision system so effective. To emulate this, there are needs for: (1) a well-defined mechanism for the generation and use of 'focus of attention' in vision systems, (2) an efficient hardware structure that makes the approach feasible. *Key Features* are used in our program. The concept of key feature is not original. Similar concepts have been used by other researchers in computer vision. An interesting implementation of a 'Local-Feature-Focus' method is described by Bolles and Cain [3], where one local feature in an image is found, and is thereafter used to predict a few nearby features to look for. In the systems described in the present paper, key features are used as the central initiating threads for the control process. They embed naturally into the hierarchical pyramid structure, organizing image-driven bottom-up and model-driven top-down processes into what appears to be a reasonably well-integrated total structure.

The overall structure of the present program is that of a pyramid of arrays. Successively smaller two-dimensional arrays are stacked and linked via a tree to one another. An image is input to the large 'retinal' array of computers at the pyramid's base and stored in an image plane of its memory. Each pixel (or if the image array is larger than the computer array, each sub-array of pixels) is stored in the memory of the corresponding processor. Computers are linked to neighbors (typically the 4 or 8 near-neighbors). They are also linked in logarithmically converging tree-like fashion to parents (typically a 2-by-2 array of children link to each parent) in the next-higher array. All processors work in parallel, computing functions on information stored in their own and in near-neighbors' memories. Results can either be stored in a designated location of the processors' own memories, or passed (and converged) to parents or to children. Thus each layer can execute typical array operations of the sort executed by the CLIP [5], DAP [20], MPP [2], and also pyramid operations that converge, combine, and disseminate information [31].

Our program runs on a simulated pyramid machine (Table 1).

Level	Size	Local Memory (bit)
0	1×1	2048
1	2×2	2048
2	4×4	2048
3	8×8	2048
4	16×16	2048
5	32×32	2048
6	64×64	1024
7	128×128	256
8	256×256	64
9	512×512	16

Table 1 Configuration of the Simulated Pyramid Machine

The basic pyramid structure has n levels, each level k ($0 \leqslant k < n$) has $2^k * 2^k$ nodes (each a simple processor with its own memory). Each node at level k is hard-wired to its 13 neighbors, i.e., 1 parent, 8 siblings and 4 children.

The connectivity between layers of nodes in the pyramid structure makes the hierarchical data flow (bottom-up and/or top-down) very efficient. The results of our program indicate that the pyramid-like structure is successful in facilitating the development of mechanisms for 'focus of attention'.

Micro-modular 'transforms' [29, 30] are used to execute processes. Transforms are IF...THEN... production-like constructs that consist of a number of conditions to look for (e.g., a two-dimensional mask of features), each with an associated weight, a threshold with which to determine whether the transform has succeeded, and a number of implied results (e.g., features, parts of objects, objects, additional transforms to apply). A transform computes a relatively local probabilistic brain-like window-matching or combining function [32]. Successive transforms are applied in a parallel and hierarchical manner. This serves to embed the very complex set of processes needed to recognize real-world objects into a potentially very fast and efficient highly parallel process, and also to

model living visual systems, with their neurons organized in converging layers in the retina, lateral geniculate, and visual cortexes.

Transforms compute values or search for features in a set of cells in one level, and then output the values or implied features (or objects) into the same or an adjacent level. When an operation involves nodes in more than one level, it is called a *pyramid operation*. Sometimes, an *array operation* is also used, in which only nodes in the same level are involved. Transforms can imply features and characteristics, sub-objects and objects, and also additional transforms to apply and particular things to look for. Hence they have the advantage of being 'procedural'. On the other hand, since transforms are often coded to find certain patterns (features or objects), they also embody 'declarative' and 'structural' aspects.

Section 2 discusses search and control mechanisms for pyramid vision programs. Section 3 presents the program's results in analyzing house and building images. Section 4 contains a brief summary and discussion.

2. Search and Control in Pyramids

2.1. Searches in the Conceptual Hierarchy and Recognition Tree

In this paper images of building scenes are used as test images. Being man-made objects, buildings may have more obvious hierarchical structures than many natural objects. To describe them, *part-of* and *geometrical relations* are extensively used. This kind of hierarchy is sometimes referred to as *conceptual hierarchy* [1]. A graph can be constructed to depict the conceptual hierarchy, in which nodes represent objects with their own feature properties, upward arrows represent the 'part-of' relations and horizontal arrows represent geometrical relations (above, below, next-to, in, contain, ...) between parts in the image [11].

Features in images are characterized as global or local, size invariant or size variant. The implication of this is that the pyramid is a convenient structure for instantiating and

looking for objects, for example using conceptual hierarchies; but now different features should more appropriately be extracted at different levels in the pyramid. Geometrical relations are handled in whatever precision required by using the actual implicit relations between several pieces of information stored in the image array. Thus the system can make use of both the implicit iconic structure of the images and successively abstracted images, and also the explicit information that it extracts and stores. Coarse relations can be obtained at high levels of the pyramid, successively more detailed relations at lower levels. With multi-computer pyramid hardware and transformed images that are stored into memories at the appropriate hardware level to represent the different levels of abstraction, features can be extracted and combined at many levels simultaneously; they will be scattered around in nodes at different levels. From a structural point of view, these features form a *recognition tree* in the pyramid for each existing object. Hence a major process of object recognition can also be viewed as a search that applies 'recognition trees' to look for the objects they represent and model. The recognition trees for all the different objects about which the program has knowledge are all embedded in the pyramid. When they have transform nodes in common (as will often be the case, especially for lower-level features like edges, angles, and curves) the transform (with pointers to the adjacent nodes in all these trees) is executed only once.

The search in the tree hierarchies in the pyramid should not be confused with the graph search in the so called 'conceptual hierarchy'. These two searches are closely related. But there is no one-to-one correspondence between the levels in these two hierarchies. In other words, features at the same level of the 'conceptual hierarchy' do not necessarily appear at the same level in the 'recognition tree' in a pyramid. Also, a simple object like 'door' in a conceptual hierarchy may consist of features at several levels of the recognition tree, e.g., longer vertical 'door-side' at a higher level, shorter horizontal 'door-top' at a lower level. Nevertheless, the search of the 'recognition tree' can still be guided by the graph of the 'conceptual hierarchy'.

2.2. The Importance of Key Features

Key features are those that appear frequently and are especially helpful in the recognition of the object in question.

Key features can often be extracted from intrinsic images. In this paper we will mainly use features of shape, texture and color as examples. Other types of intrinsic features (e.g., surface orientation, occluding contour and disparity) should be investigated, especially if the program is extended to general three-dimensional domains.

The reason for introducing the notion of 'key features' is that they form a significantly smaller and especially important subset of the whole feature space. In real-world image analysis, the possible space of features that are occasionally useful is usually much too big to deal with. As a program like ours is given more and more transforms, the importance of first using a small set of key features increases. It is more efficient to first focus the system's attention on the more important features. Moreover, many psychological observations suggest that this is exactly the way human vision systems works [18, 21]. Yarbus reported his observations of the human eye movements during perception of complex objects in [34]:

> Records of eye movements show that the observer's attention is usually held only by certain elements of the picture. When looking at a human face, an observer usually pays more attention to the eyes, the lips, and the nose. The other parts of the face are given much more cursory consideration. ... Analysis of the eye-movement records shows that the elements attracting attention may, in the observer's opinion, contain information useful and essential for perception.

This is not to argue that eye movement is essential to recognition. Recognition often takes only a few hundred milliseconds, and can succeed without this kind of (slow) eye movement. However, psychological observations have certainly revealed that the brain's processes do focus on key features during the perception of complex objects.

2.3. Focus Areas

The discovery by the perceiver of some significant key features suggests the extraction of

other features in the conceptual hierarchy — to generate a hypothesis. Once the key features are located, the enclosed or neighboring areas will attract special attention. The spatial areas on which the attention is thus focused are named *focus areas*. For example, in houses the areas enclosed by pairs of parallel long edges are possible window, shutter or door areas. Features (e.g., colors and textures) in these areas are further studied, and, in some sense, verified.

For texture measures, *edgeness* and *edge separation* are used in our programs, because they are simple and effective. They can both be derived from the micro-edge map that the program builds up in its first application of transforms to the raw input image (Section 3 will present the details of the implementation). Edgeness is the number of edges in a certain unit area. Homogeneous regions, e.g., the sky, doors and window shutters (at the appropriate image resolution) have low edgeness measures. On the other hand, trees, windows and textured wall areas will have a relatively high edgeness measure. Edge separation refers to the distance measure between micro-edges with eight different orientations. Kjell [10] demonstrated good results using the edge separation method for classifying images with different textures, and for image segmentation. For digital images with good resolution, tree areas and brick wall areas will both have high edgeness measures, but they will have completely different measures on edge separation, since bricks have regular horizontal and vertical layouts whereas leaves do not.

In our implementation, simple statistical data (e.g., *mean*, *standard deviation* or *variance*) on the feature values within the focus areas are used. Pyramid operations are employed for gathering statistics in the following manner. At first, the higher level nodes that found the key features will specify the locations of the focus areas by passing the coordinate information down to the child nodes and marking the nodes inside the focus areas. Then the nodes inside the focus areas at an appropriate low level apply the local measures (e.g., on edgeness or edge separation).

If the variable representing the feature value is x, then for a focus area with n nodes, the *mean* (μ) for these x values is

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}.$$

The values of $\sum_{i=1}^{n} x_i$ and n can be aggregated by the parent nodes at consecutively higher levels. Pyramid operations with 2×2 windows are suitable and efficient for doing this. These are SIMD operations. Nodes outside the focus areas are masked out. Only one bottom-up pass is needed to capture the *mean* values.

The definition for the *variance* (σ^2) of the distribution of x is

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(x_i - \mu \right)^2}{n}.$$

If this definition is used to derive σ^2, then μ must be known before the calculation of σ^2. Therefore two passes are needed, the first for μ, the second for obtaining σ^2. Between these two passes, the μ values must be propagated down to the nodes in the original focus areas.

However, there is a 'short-cut' formula for the *variance* as below:

$$\sigma^2 = \frac{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i \right)^2}{n}}{n}.$$

With this formula there is no need to pre-determine the *mean* value. The values of $\sum_{i=1}^{n} x_i$, $\sum_{i=1}^{n} x_i^2$ and n can all be aggregated in a bottom-up way in the same pass. This allows the computation for both μ and σ^2 to be completed in just one pass. Although slightly more calculations are conducted, the control for data flow is simplified in this approach. In a complicated multilevel system like this, the control can sometimes be a bottleneck. To avoid more complicated control and data flow, the one-pass method was adopted.

From the process control point of view, the program's attention is initially on the key features. After the key features are extracted successfully in a bottom-up manner, attention shifts to the focus areas. The process now turns into a top-down model-driven mode to look for particular interior features in the focus areas. For example, window panes will be expected in possible window areas, whereas elongated homogeneous areas will be expected to indicate shutters. However, the parent (or grandparent) nodes need only specify the 'goal' and mark the 'focus areas'. The actual data for the measure of the features will, as just described, again flow in a bottom-up way. Thus the program combines bottom-up and top-down processing.

2.4. Lateral Search

In low level image pattern analysis, transforms are basically probabilistic template-like thresholding operations. More and more global features can often be built up in a hierarchical bottom-up way. Sometimes fairly simple, local transforms can yield good results. Good masks for low level features such as edges, curves, and corners can be relatively easily coded as several-level hierarchies executed over several layers of the pyramid, and applied to almost any image. At higher levels this type of mask is harder to find, since there are many more variations of feature combinations.

The key features and focus areas method makes use of cycles of bottom-up and top-down data flow to accumulate pieces of evidence. However, there is also a need to add more flexible search strategies to deal with more complex patterns in the real-world images. It is often too likely to result in error to increase the weight of the 'house' based only on some door-like features, say a rectangle. *Lateral search* is thus introduced to more effectively handle geometrical relations between objects in the outdoor-scene analysis. Based on the discovery of some key features, 'attention' can now also be directed to other key features at the same level in the conceptual hierarchy of the scene. The feature search routines are coded in the transforms, and can be triggered dynamically. In cases where such pairs of key features are found in the 'lateral search', the implied object gets a

significant increase in its weight.

There is a variety of neurophysiological evidence that supports the concept of "lateral interconnections between areas at the same hierarchical level" [6, 14].

The term 'lateral search' refers to the search in a conceptual hierarchy. For the corresponding search on a subpart of the recognition tree, whose nodes are usually scattered around at many different levels in the pyramid, several levels may be involved simultaneously in this so called 'lateral' search. Often, this kind of 'lateral search' is actually accomplished by certain pyramid operations where the parent nodes use elongated large windows (e.g, 2×6, or 6×2) to look for pairs of features in expected directions within the window.

The lateral search need not be executed serially (node by node). Each layer of the pyramid is assumed to be an SIMD array multi-computer. The search can be executed in parallel by all processors in the appropriate layer, with those processors not involved masked out. In the future, more powerful MIMD multi-computers with more sophisticated connections between them at the higher levels of the pyramid could be built that would significantly speed up processing.

2.5. Hypothesis Generation from Low Resolution Images

We have seen how the successful extraction of key features can lead to the generation of subsequent hypotheses that guide top-down and lateral processing for their verification.

However, in order to know what to start with, and where, we sometimes need, (a) an initial hypothesis of what key features to expect; and (b) an estimation of the object size. The latter is important, because it suggests the appropriate resolutions (levels) to extract and aggregate features for the recognition. These hypotheses can, and must, be generated by the higher level processors on the basis of an initial rough analysis on images with reduced resolutions, because (a) global information is needed to generate such hypotheses, and is available only at higher levels; (b) certain salient, coarse and size invariant features

can be extracted at reduced resolutions.

Once a hypothesis is generated, the system's attention will be focused on the hypothesized key features. This process is much like a human vision system glancing at an object in a scene using 'peripheral vision', thus allowing the perception of a wide angle in a low resolution, and then directing the attention to some details using the high-resolution foveal areas.

It takes little time to generate such a hypothesis. On the other hand, such a hypothesis is obviously not necessarily accurate. Although a pyramid algorithm for the relatively powerful median filtering operation is employed to generate the low resolution images, some important features still get lost. To ensure good performance, such hypotheses should include several alternatives (guesses), and be treated as though they are subject to later modifications.

3. The Performance of the Program

In this work only front views of building scenes are used. The camera (viewpoint) is sufficiently far from the object so that perspective projection need not be taken into account. Many three-dimensional image analysis problems are not addressed. By choosing a small subset of detailed TV images of real-world building scenes, our examples are limited to a relatively well-defined, yet fairly general domain.

The program for outdoor-scene analysis was developed on a simulated pyramid. It was initially tested on two house images, 'house34' and 'house35'. After the transforms were developed, a third image 'house.sri', which was kindly supplied by Wesley, who used it in [33], was also used. The program appears to exhibit some reasonable generality, since it works as well on that image. Finally, the program was also tested on an office building image ('building1') to see if it can distinguish an office (or apartment) building from an ordinary house by examining the number and the layout pattern of the windows. This section gives some of the results from the analysis of these images.

The digitized images all have a resolution of 512×512. They are input to and reside at level 9 in the pyramid. Fig. 1 shows these four images at this resolution. Color images are digitized with three component (RGB) images. The pyramidal median filtering technique [12] is used to obtain multi-resolution images at the levels 8, 7 and 6. Next, edge operators are applied for getting micro-edges at these levels, producing micro-edge maps with eight directions encoded by 0, 1, ..., 7. Afterwards an edge thinning algorithm is used to get micro-edges with single pixel width. Fig. 2(a) and (b) show such micro-edge maps for 'house35' and 'house.sri' at level 7 with a resolution of 128×128. The main steps for the analysis of the house images are described in the following sections:

3.1. Feature Extraction

3.1.1. The Selection of Proper Windows

In contrast to other methods, the pyramid approach makes possible the effective use of relatively small cascaded windows to assess global processes in recognizing objects. If the object is too large to 'see' at one level, it will be visible at some higher level of the pyramid, where the features will have been successively computed and pulled closer.

Since each parent node has direct links to its four children (direct children), windows of 2 by 2 are the most natural choice. Operations within such windows are the most efficient to execute and thus are often favored. But a 2×2 window has drawbacks. First, this type of window is a 'non-overlapped window'. Such non-overlapped windows have 'cracks' between them [22]. This means that small features may not be detectable except at very high levels. Hence the number of levels needed to detect a feature is not proportional to the logarithm of its size. Second, a 'conceptual hierarchy' is formed according to the part-of relation. Some features, e.g., long edges, are semantically low level information in the conceptual hierarchy. They may represent the side of a door, or the top of a roof. However due to their length, they may only be combined at the very top levels of the pyramid. This can lead to difficulties, since features which belong to the

same level in the conceptual hierarchy (e.g., the top and the sides of a door) are now separated by too many levels in the recognition tree in the pyramid.

In our program, windows of 2×2, 3×3, 4×4 and up to 6×6 are selected, according to the particular task. The pyramid operations for such relatively small window sizes can be implemented with reasonable efficiency and speed. For a parent P to access one of its 6×6 children, if this child C is not its direct child, P can use its sibling link to 'talk' to one of its 8-neighbors P', then P' can use its direct link to get this C node for the node P.

However, large windows have their own problems, in addition to longer execution time. Since the child sets of neighboring parents overlap a great deal, the same features will be found redundantly in many nodes at the neighborhood of the expected location. As illustrated in Fig. 3, with a 6×6 window, the long vertical edge at level k+1 can be completely detected at level k by nodes 4,5,6 and mostly detected by nodes 1,2,3,7,8,9. In our recognition scheme, there is no benefit if all the nodes extract the same information in such a redundant way. The redundancy becomes more severe, when features are combined through several levels.

Feature-adaptable windows are used to lessen this problem. As an example, consider again the window with size 6. For extracting the long vertical edge in Fig. 3 , only a 6×2 window, i.e., the central strip of the 6×6 window, need be defined. In this way, the entire edge falls only in the window of node 5, while part of the edge can be seen in nodes 2 and 8. Fig. 4 depicts some of the feature-adaptable windows for detecting long edges with different orientations.

3.1.2. Extraction of Collinear Long Edges

Since doors and windows are key features for houses, transforms were developed for locating them. A close examination of a number of buildings suggests that collinear long edges in orthogonal directions are often key features for windows, shutters and doors. This step illustrates the extraction of these long edges.

First, short edges are built from micro-edges. Pyramid operations are used: specifically, all the nodes at level 6 examine their 4×4 children at level 7. Take 'shortedge0' as an example. If a pattern of several micro-edge0's lining up vertically is detected in its window, the parent node claims a 'shortedge0'. In order to reduce the chance of wrong combinations of micro-edges, the actual handling of the 4×4 window is somewhat subtle. At first, the parent examines only its direct children (i.e., the center 2×2 children). Only when a micro-edge0 with a sufficient weight is found at some direct child node (e.g., at the upper left direct child), will the parent then look for other possible micro-edge0's in a 3×3 window centered around this child. Hence this 3×3 window is indeed a subwindow of the 4×4 window (e.g., the upper left part of the 4×4 window). In this way, the 4×4 window is made more adaptable to different orientations of short edges. The resulting shortedge0, ..., shortedge7 are stored at level 6.

Second, short edges are combined into long edges. As discussed in section 3.1.1, variations of 6×6 feature-adaptable windows are employed for this task. The transforms for the detection of short and long edges work in the same hierarchical way. The long edges (named 0 - 7) are extracted by level 5 nodes. Fig. 5 depicts the resulting 'longedge0' and 'longedge4' for 'house35', where the numbers indicate the column coordinates of these vertical long edges at level 7's resolution (128×128).

After these two steps, efforts are made to build still higher level features. Doors and windows are usually rectangular in the front views of buildings. Considering that doors and windows of houses can often be partially occluded by other objects, e.g., bushes or trees in front of the house, the program should also be able to deal with part of a rectangle, especially some antiparallel long vertical edges. The level 4 nodes use pyramid operations to extract pairs of such antiparallel long edges currently stored at level 5. For example, a 2×6 window is specified to search for pairs of vertical long edges ('longedge0' at the left and 'longedge4' at the right, or vice versa).

3.2. Search in Focus Areas

Once such collinear long edges are located, the enclosed regions are treated as focus areas. Usually the program will examine more details of the image at lower levels. The full program executes a number of sequences of this sort of shallowly serial searches.

Fig. 6 is a micro-edge map for the lower left window in 'house35' under the original resolution (512×512). In this particular example, window panes are not hard to recognize at this high resolution. Once attention is focused on this area, evidence for a window is abundant. Successive transforms (for short edges, Ljoints, rectangles) are employed from level 8 up to verify the hypothesis of 'window'.

The program also gathers statistics on features such as colors and textures within these focus areas. As an example, the study of 'edgeness' is shown. The motivation for studying 'edgeness' is to distinguish subparts with high 'edgeness' values (e.g., trees, windows, textured roofs) from others with low values (e.g., single-colored door, sky). At levels 5, 6 and 7, the focus areas are specified by the level 4 nodes which found the pairs of antiparallel long edges. Local measures on edgeness are then first made at level 7. Each of the level 7 nodes counts the total number of micro-edges of its 2×2 direct children at level 8. The maximum possible edgeness value is 4, the minimum is 0. Based on this edgeness count, nodes at levels 6, 5, and then level 4 get the statistical information (mean and variance of edgeness) at all focus areas. Fig. 7 (a) and (b) show the data acquired at level 4 for the focus areas in 'house35' and 'house.sri'. The areas are enclosed by pairs of long edges 4 and 0, with 'longedge4' either at the left or at the right. The boundary coordinates for these rectangular areas are also listed. (The data are scattered in different locations at level 4. The presentation of these global tables is for the sake of clarity.) Doors and windows in these images have mostly been identified as focus areas and have some interesting statistics. For example, in 'house35' doors and window shutters show near-zero mean and variance values due to their uniform color and intensity, whereas window areas have high edgeness values due to the edges of all the frames.

The search in the focus area is very powerful for studying properties of subparts with relatively big regions. Our program started with edge properties. By focusing in the areas bounded by long edges, the program is capable of combining region properties with edge properties.

3.3. An Example of Lateral Search

Although the transforms described are only a few examples of those actually used, the analysis needed for the focus areas, e.g., the ones in Fig. 7, is already non-trivial. The system can invoke further transforms on more features (e.g., size, elongation and color of the region) to disambiguate them. But to understand the subparts in the images, the more important information here is the geometrical locations and relations between the door and windows.

In the building images with front views, windows are often lined up horizontally and/or vertically, doors are usually at lower positions than windows, and the window shutters (if any) are very good indications of the locations of the windows. The 'lateral search' routines use these geometrical relations. For instance, every time a window is implied, the program will look for the door or other windows at a range of certain distances and directions.

As an example, consider the window-door lateral search for 'house35' in a little more detail. As usual, many of the entries in Fig. 7(a) are not important parts of the house. Some of them are even in the tree area. Since some lower left node at level 4 got a high weight for the left window (Fig. 6) in the previous step, search routines were triggered to laterally look for other windows and door. The routines succeeded in finding the door and lower right window at this time. As entries in Fig. 7(a) indicate, there are two 'window assemblies' (combinations of one window area and two shutter areas) in 'house35'. The entries marked 'window' and 'shutter' indicate that their top is at about row 72, and their bottom at about row 83. The left and right boundaries for the left window assembly are column 29 and column 46; for the right window assembly they are column 88 and column

106. Similarly, the door-assembly (combination of one door and two door frames) has its (top, bottom, left, right) at (72, 87, 59, 74). The shutters, doors, and door frames have basically zero mean and variance values on the edgeness measure, whereas the windows have high mean and variance values. For locating the shutters and doors, the program first finds 'non-tiny' and 'low-edgeness' areas. Focus areas whose *area* is larger than 10 and *mean* not larger than 0.5 are chosen, as shown in Fig. 8(a). After the lateral search routines find the matched shutter and door areas, they search further for the evidence of the two entire window assemblies and one door assembly, with their appropriate spatial relations. The result is shown in Fig. 8(b), where the first three rows show the lower left window assembly, and so on. The partial conclusions are: (a) The lower windows have clearly visible frames and panes, the shutters next to the lower windows have a low-edgeness texture measure; (b) a door is between two windows, the bottom of the door is lower than the bottom of the windows, their top is on the same line; etc. In this case, the weights for the door and windows, and also for house, are increased substantially, with confidence.

3.4. More Searches Initiated from High Levels

Due to the topology of the pyramid, the high level nodes can naturally be used to access global data.

As discussed in section 2.5, sometimes global features derived directly from low resolution images (e.g., 64×64) can be used by higher level nodes to generate hypotheses at the beginning of the analysis. In outdoor-scenes, colors of large regions are good size invariant features and useful clues for regions like sky and grass. The large sky-colored regions above some long horizontal or triangular roof-shaped edges, and the green grass-colored regions lower down, imply the possible existence of a house in between these two regions. The hypothesis thus generated suggests the possible set of key features (e.g., doors and windows) to look for and the possible 'best-levels' to start at.

However, what may be an even more important role that the high level nodes play is the combining and abstracting of features collected by lower level processes. A simple example is the recognition of the outlines and roofs of buildings. These are more global features and they are aggregated at higher levels in the pyramid. When a certain set of features has been implied with sufficiently high weights, some goal-driven search routines are invoked to look for other related features in the conceptual hierarchy. Now these features no longer have to be key features. By accumulating more evidence, the recognition process becomes more thorough. This type of search is especially useful for the features whose extraction heavily relies on global knowledge of the scene. For example, the chimney in 'House35' was initially not noticed by the program. It is recognized after there is already an indication of the shape and location of the roof.

As a more detailed example, the analysis for textures of walls in 'house34' and 'house35' will be examined. The walls of 'house34' are built of shingles that create many regularly arranged horizontal long edges ('longedge2' and 'longedge6') in the image. The walls of 'house35' have almost no trace of edges at the current image resolution. The evidence of the locations of the key features (windows and doors) of the houses is used to point out the possible locations of the wall areas. Then the statistics on the edgeness and edge separation for the wall areas are measured. Fig. 9 is an illustration of the data collected from these areas. In 'house35' there is evidence for a 'door-assembly' whose (top, bottom, left, right) is (72, 83, 59, 74), and one of the 'window-assemblies' at (72, 83, 88, 106). Two sample 'wall areas' are thus chosen to be at (64, 68, 88, 106) and (72, 83, 78, 84); one is above the window, and the other between the door and the window. The wall areas in 'house34' are chosen in a similar way. As seen in Fig. 9(a) the mean values for edgeness in 'house34' are relatively high, as they should be. The vertical distance between 'longedge2' and 'longedge6' is used as the measure of 'edge separation'. The statistics of edge separation are quite informative. The mean value of 2 indicates that horizontal long edges are separated on the average by two cells (at the resolution of 256×256) and the variance value of 0 indicates the extremely regular horizontal edge

separation pattern in the two chosen wall areas of 'house34'. In Fig. 9(b), the low mean 'edgeness' values indicate there are very few edges in the wall areas of 'house35'. Since there are no pairs of 'longedge2' and 'longedge6' existing within a certain maximum search distance, the mean and variance for edge separation in 'house35' get a value 'undefined'.

Generally, the program can handle image data that is noisier and more complex than what is presented in Fig. 9. However, this example is sufficient to illustrate how the high level nodes can effectively initiate more search routines, ones that improve the recognition process.

3.5. Combining Information Using Evidential Reasoning

To perceive real-world images, a vision system needs intelligent reasoning mechanisms. Our program uses its own knowledge representation technique, and the evidential reasoning mechanism based on key features with the application of the Dempster-Shafer theory to combine evidence. A detailed description is presented in [11, 13]. The program is capable of accumulating evidence and reasoning under uncertainty. Table 2 shows the partial results of the reasoning process from the analysis of the image 'building1'.

	Possible window areas												
	W1-6	W7	W8	W9	W10	W11	W12	4	5	9	15	17	18
Bel (elong)	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.3	0.5	0.5	0.3	0.5	0.5
Bel (text)	0.4	0.2	0.4	0.4	0.4	0.4	0.4	0.4	0	0.4	0.4	0	0
Bel (lt −bound)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0	0.1	0	0	0.6	0.3
Bel (rt −bound)	0.6	0.6	0.6	0.3	0.1	0.6	0.6	0.6	0.3	0.3	0.6	0	0.1
Bel (wnd)	.449	.409	.449	.407	.379	.449	.449	.335	.205	.262	.335	.285	.205
Bel (v −sibl)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0	0	0.6	0	0.6	0.6
Bel (h −sibl)	0.6	0.6	0.6	0.6	0.6	0.6	0.6	0	0.6	0	0	0	0
Bel '(wnd)	.492	.475	.492	.475	.462	.492	.492	.134	.203	.225	.134	.234	.203
Bel (non −wnd)	0	0	0	0	0	0	0	0.5	0.5	0	0	0	0.5
Bel "(wnd)	.492	.475	.492	.475	.462	.492	.492	.080	.166	.225	.134	.234	.166

Table 2 Belief Values for Some Possible Windows in 'Building1'

After extracting pairs of collinear horizontal long edges, 46 areas are hypothesized as possible window areas. To make this example clearer, 12 real window areas are labeled 'W1' to 'W12', the others '1' to '34'. Only the 12 real window areas and 6 'non-window' areas which carry relatively high weights and thus are not easy to discriminate from the real windows are shown in Table 2. At first, *belief* values (similar to weights) are assigned to each area according to the 'perfectness' of its *elongation*, *texture*, and the extracted *left-boundary* and *right-boundary*. These belief values are combined to obtain the belief value *Bel* (*wnd*) for each area. This step already gives higher belief values to the 12 real windows than any of the 'non-window' areas. However, it can be seen that the value for 'W10' (0.379) is fairly close to the value of non-window '4' and '15' (0.335). Contextual geometrical relations are thus used to improve the result. Since the windows of an office (or apartment) building are usually arranged in a horizontal or vertical alignment, the 'lateral search' for horizontal or vertical sibling windows is conducted. It turns out that the real windows are successful in finding their neighbors and hence have their *Bel* (*wnd*) increased; whereas the non-window areas are unable to find their neighbors in this way and thus get their *Bel* (*wnd*) decreased. The new values are indicated by *Bel* ' (*wnd*). Finally, the approximate locations of the outlines of the building are used (they are actually propagated down from a higher level in the pyramid). Some of the areas, e.g., '4', '5' and '18', are ruled to be probably out of the building boundaries due to their locations. Their chances of being real windows are further reduced. The final values *Bel* " (*wnd*) show that the program is very confident about the existence of the 12 real windows. Because the program found strong evidence of the number and the arrangement of the windows at this level, it combined this with other evidence (e.g., the shape and outline of the building), and succeeded in recognizing the object in 'Building1' as an office or apartment building.

4. Summary and Discussion

This paper describes the search and control strategy which is made feasible by the use of key features. Following the model of 'preattentive processes', the transforms that successfully extracted key features are used to trigger other transforms for 'focus of attention' in focus areas, and also for 'lateral search' in extracting other key features. Since the searches are concentrated in relatively small and especially important subsets in the feature space, recognition is accomplished efficiently and effectively. This is attested to by the program's success in recognizing buildings (houses) in four different test images. The pyramid-like hierarchical structure facilitates the combining of both bottom-up and top-down searches. By the application of key features and the combination of a variety of effective search patterns, the program accumulates evidence about the important components of buildings and their surroundings, e.g., windows and doors, as well as roofs, chimneys, walls, glasses, bushes, trees, sky, ... , and finally reaches the goal of recognizing different buildings in outdoor scenes.

The VISIONS system [8] was proposed as a general system for the interpretation of static outdoor scenes. The system is composed of a segmentation subsystem and an interpretation subsystem. The segmentation subsystem extracts features in a 'processing cone'. The results of 'boundary analysis' and 'region analysis' are merged to produce the segmentation of images. The interpretation subsystem interprets features in terms of world knowledge, and proposes new features to be looked for by the segmentation process. There have been many follow-on publications about segmentation techniques as well as several concerning knowledge representation and reasoning in the VISIONS system. The main difficulties seem to be: (a) good segmentations are exceedingly difficult to achieve (indeed we conjecture that they cannot be achieved except after objects have been recognized), (b) a suitable knowledge representation and reasoning technique for the complex outdoor-scene domain remains to be satisfactorily developed.

The present paper also examines the domain of static outdoor scenes. The pyramid programs developed have many similarities to the VISIONS system, since both take a parallel-serial hierarchical approach to object recognition in a pyramid-like structure. However, our approach attempts to combine lower- and higher-level processes into a well-integrated multi-level system. Our program does more processing (both low-level and high-level) that is highly parallel using neuron-like micro-modular transforms within the parallel structure and therefore should be substantially faster (given appropriate hardware). We do not believe that a complete and near perfect segmentation is possible, or necessary, on most image analysis tasks. Segmentation should serve to help recognition; it is not an end in itself. This paper describes how key features are used to alleviate the segmentation problem, and to initiate a smooth combination of bottom-up, top-down, and lateral flows of processes. The emphasis is on the search and control aspects related to the use of key features that process and help focus attention on important parts of the scene.

The present program was tested on only a small number of images; but these were complex real-world images. The program's knowledge of key features, and of the entire set of recognition trees of features that it uses to model all the objects are expressed as micro-modular transforms. So this program can be made substantially more general and more powerful, simply by adding more transforms. Given suitable parallel hardware, it will still be as fast; therefore it offers real promise of real-time perception of real-world objects.

5. References

[1] R. Bajcsy and D.A. Rosenthal, "Visual and conceptual focus of attention," pp. 133-150 in *Structured Computer Vision*, ed. S. Tanimoto and A. Klinger, (1980).

[2] K.E. Batcher, "Design of a massively parallel processor," *IEEE Trans. Computers* C-29(9) pp. 836-840 (1980).

[3] R.C. Bolles and R.A. Cain, "Recognizing and locating partially visible objects: the local-feature-focus method," *The International Journal of Robotics Research* 1(3) pp. 57-82 (1982).

[4] P.J. Burt, T.H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Trans. Systems, Man and Cybernetics* **SMC-11**(12) pp. 802-809 (1981).

[5] M.J.B. Duff, "CLIP4: a large scale integrated circuit array parallel processor," *Proc. 3rd IJCPR*, pp. 728-733 (1976).

[6] D.C. Van Essen and J.H.R. Maunsell, "Hierarchical organization and functional streams in the visual cortex," *Trends in NeuroSciences*, pp. 370-375 (1983).

[7] D.C. Van Essen, "Functional organization of primate visual cortex," pp. 259-329 in *Cerebral Cortex: Vol. 3. Visual Cortex*, ed. A. Peters and E.G. Jones, Plenum (1985).

[8] A.R. Hanson and E.M. Riseman, "Visions: a computer system for interpreting scenes," pp. 303-334 in *Computer Vision System*, ed. A.R. Hanson and E.M. Riseman, (1978).

[9] D.H. Hubel and T.N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *Journal of Physiology (London)* **160** pp. 106-154 (1962).

[10] B. Kjell, *Oriented edge separation texture measures*, Ph.D. Thesis, University of Wisconsin-Madison (1985).

[11] Z.N. Li, *Pyramid vision using key features and evidential reasoning*, Ph.D. Thesis, University of Wisconsin-Madison (1986a).

[12] Z.N. Li and L. Uhr, "A pyramidal approach for the recognition of neurons using key features," *Pattern Recognition* **19**(1) pp. 55-62 (1986b).

[13] Z.N. Li and L. Uhr, "Evidential reasoning in parallel hierarchical vision programs," *Proc. Second Workshop on "Uncertainty in Artificial Intelligence"*, pp. 175-182 (1986c).

[14] M.S. Livingstone and D.H. Hubel, "Anatomy and physiology of a color system in the primate visual cortex," *Journal of Neuroscience* **4** pp. 309-356 (1984).

[15] D. Marr and H.K. Nishihara, "Representation and recognition of the spatial organization of Three-Dimensional Shapes," *Proc. Royal Society of London* **Series B, 200** pp. 269-294 (1978).

[16] M. Mesulam, "The functional anatomy and hemispheric specialization for directed attention," *Trends in NeuroSciences*, pp. 384-387 (Sep. 1983).

[17] R. Miller, *Pyramid Computer Algorithms*, Ph.D. Thesis, Dept. of Math., SUNY Binghamton (1984).

[18] U. Neisser, *Cognitive psychology*, Appleton (1967).

[19] C.F. Neveu, C.R. Dyer, and R.T. Chin, "Two-dimensional object recognition using multiresolution models," *Computer Vision, Graphics, and Image Processing* **34** pp. 52-65 (1986).

[20] S.F. Reddaway, "DAP — a flexible number cruncher," *Proc. LASL Workshop Vector Parallel Processor*, pp. 233-234 (1978).

[21] I. Rock, *An introduction to perception*, MacMillan (1975).

[22] A. Rosenfeld, "Quadtrees and pyramids for pattern recognition and image processing," *Proc. 5th ICPR*, pp. 802-811 (1980).

[23] A. Rosenfeld, *Multiresolution image processing and analysis*, Springer-Verlag (1984).

[24] D. Sabbah, "Design of a highly parallel visual recognition system," *Proc. 7th IJCAI*, pp. 722-727 (1981).

[25] L.A. Schmitt, *A structured approach to computer image understanding: the use and representation of real-world knowledge in an artificial vision system*, Ph.D. Thesis, University of Wisconsin-Madison (1982).

[26] Q.F. Stout, "An algorithmic comparison of arrays and pyramids," in *Evaluation of Multicomputers for Image Processing*, ed. L. Uhr, K. Preston, S. Levialdi and M.J.B. Duff, Academic Press (1985).

[27] S.L. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing," *Computer Graphics and Image Processing* **4** pp. 104-119 (1975).

[28] S.L. Tanimoto, "Algorithms for median filtering of images on a pyramid machine," pp. 123-141 in *Computing Structures for Image Processing*, ed. M.T.B. Duff, Academic Press (1983).

[29] L. Uhr, "Layered 'recognition cone' networks that preprocess, classify and describe," *IEEE Trans. Computer* C-21 pp. 758-768 (1972).

[30] L. Uhr and R.J. Douglass, "A parallel-serial recognition cone system for perception: some test results," *Pattern Recognition* **11** pp. 29-39 (1979).

[31] L. Uhr, *Algorithm-structured computer arrays and networks*, Academic Press (1984a).

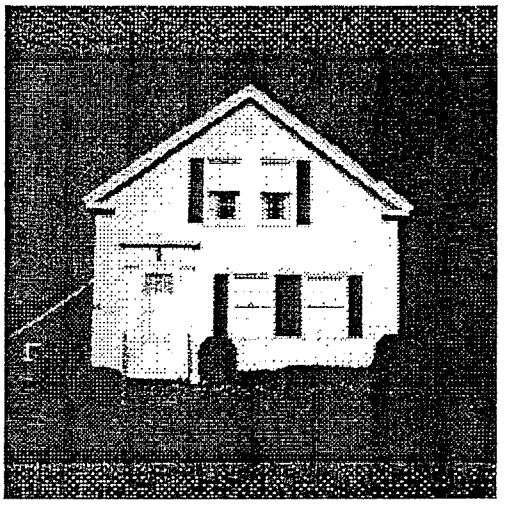
[32] L. Uhr and L. Schmitt, "The several steps from icon to symbol, using structured cone/pyramids," pp. 86-100 in *Multi-Resolution Systems for Image Processing*, ed. A. Rosenfeld, North-Holland (1984b).

[33] L.P. Wesley, J.D. Lowrance, and T.D. Garvey, "Reasoning about control: an evidential approach," SRI Technical Note 324 (1984).

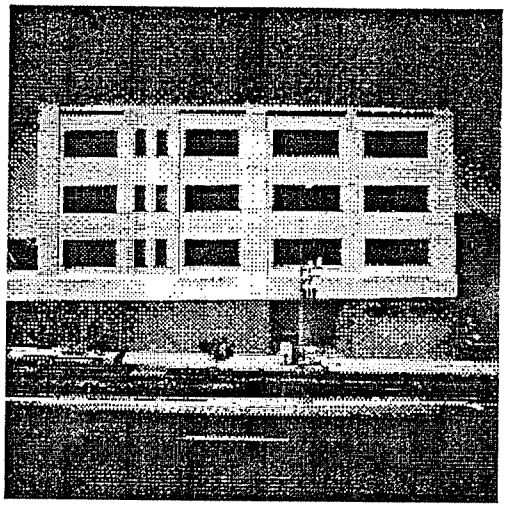
[34] A.L. Yarbus, *Eye movements and vision*, Plenum, New York (1967).

(a) House34

(b) House35



(c) House.sri



(d) Building1

Fig. 1 Four Building Images

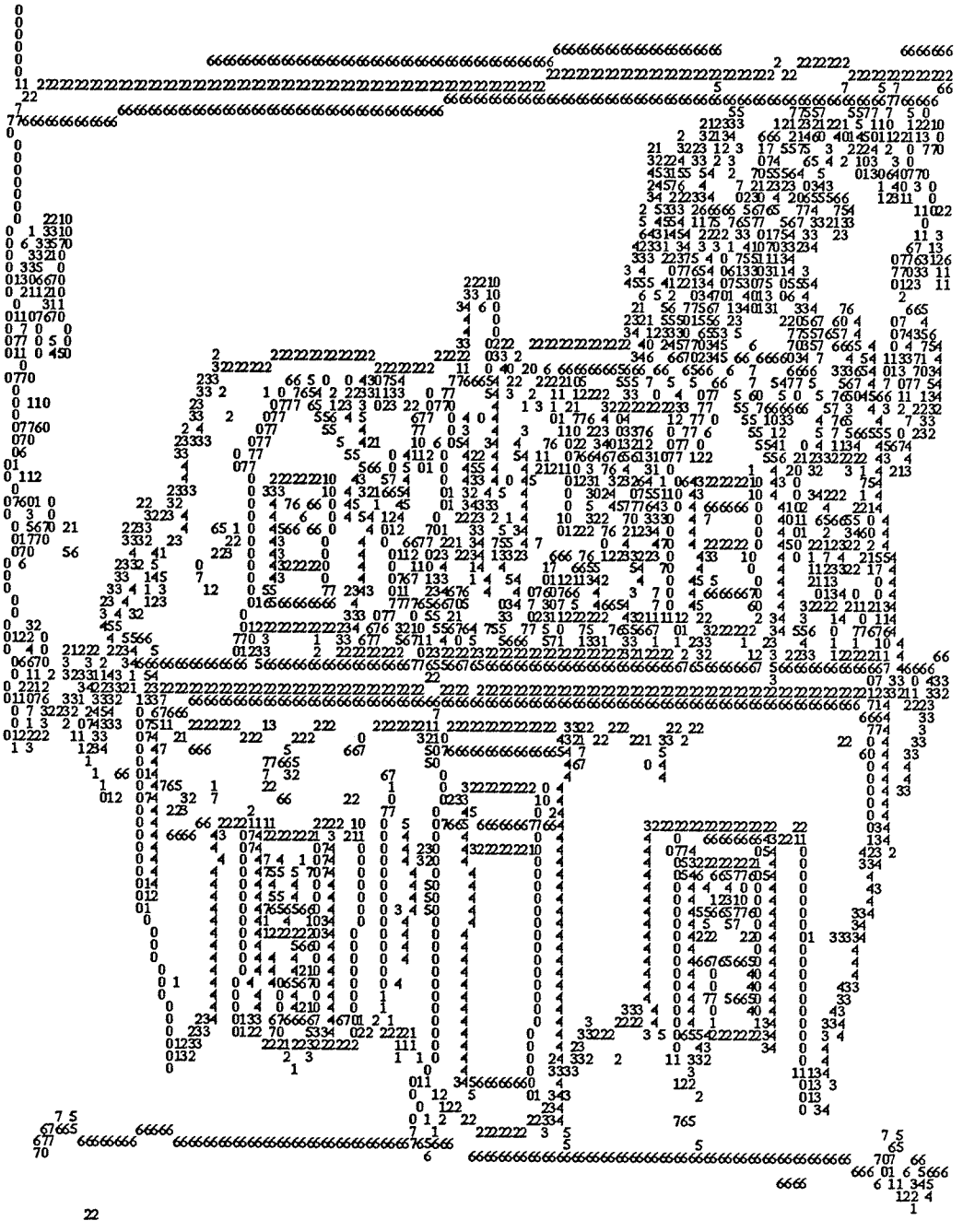


Fig. 2 (a) Micro-Edge Map for 'House35'

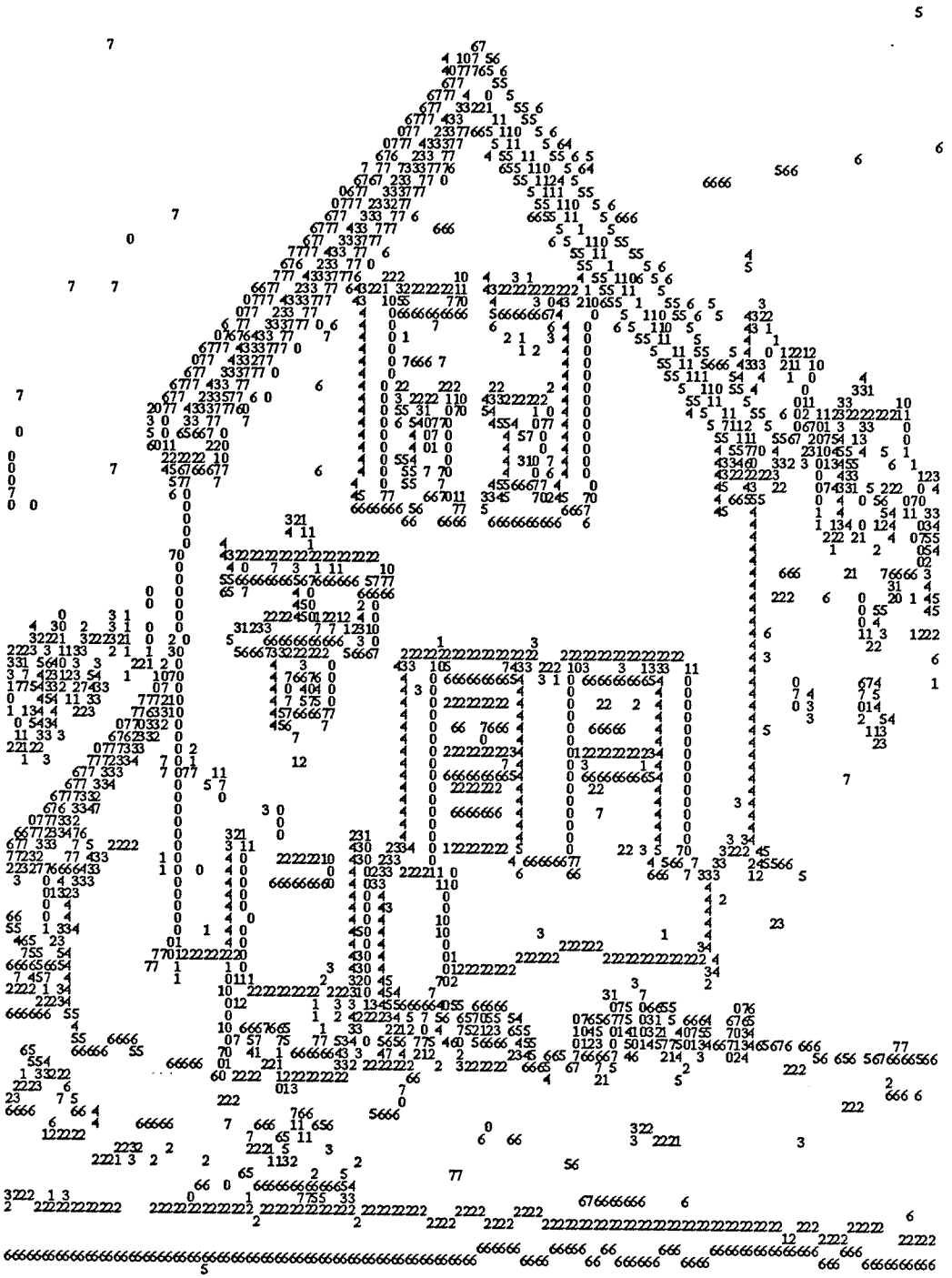


Fig. 2 (b) Micro-Edge Map for 'House.sri'

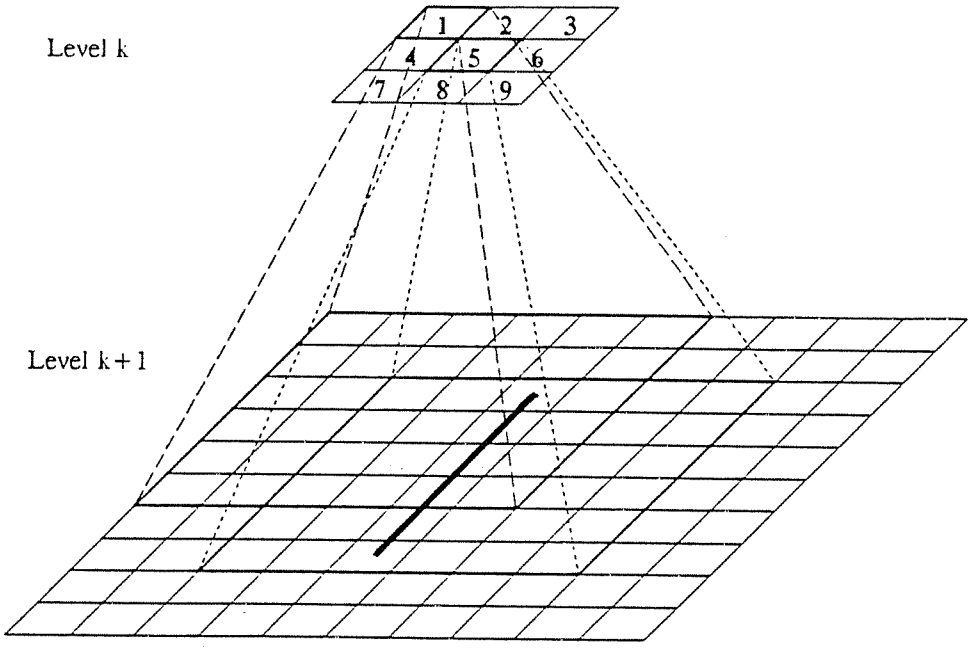


Fig. 3 Problems with Larger Windows

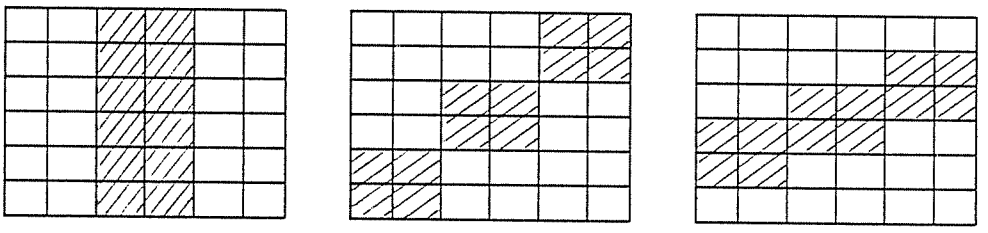
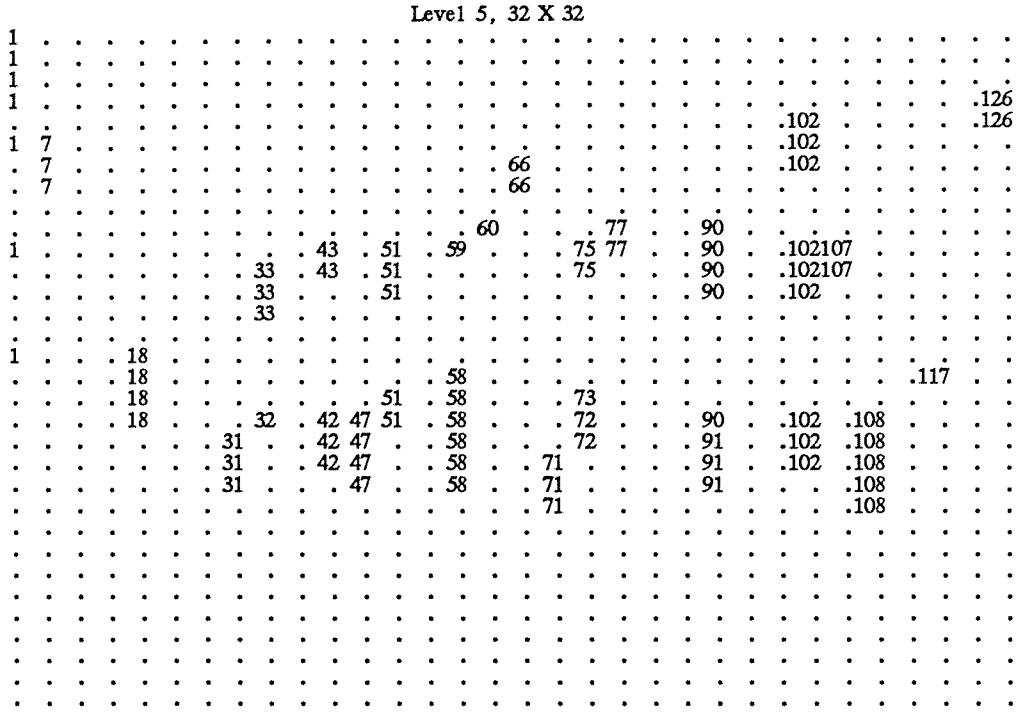
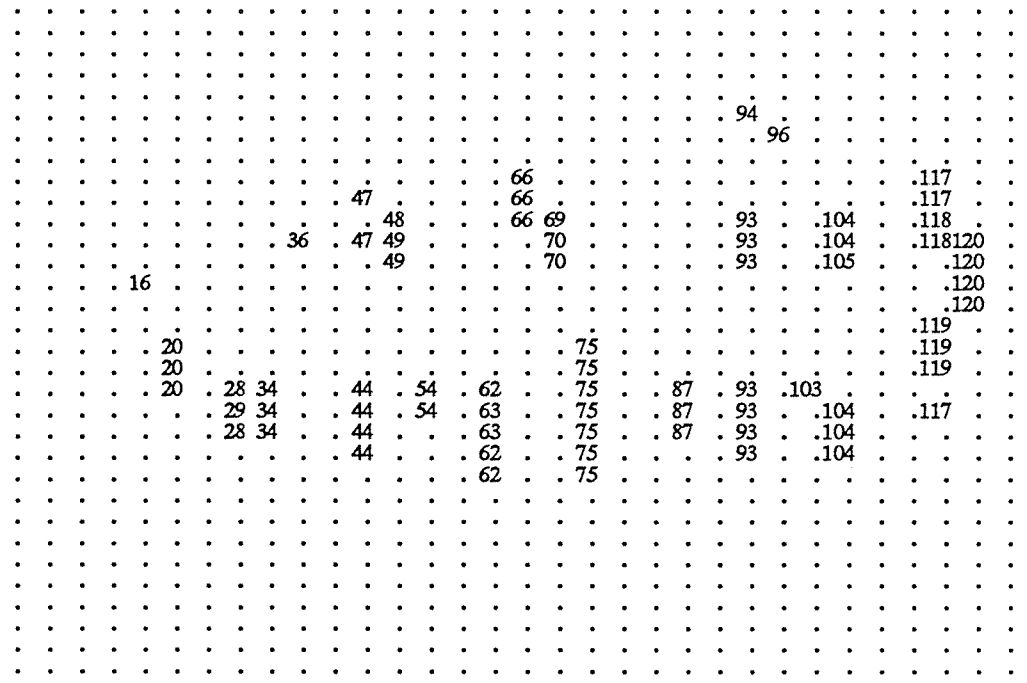


Fig. 4 Windows for Detecting Long Edges



(a) 'longedge0'



(b) 'longedge4'

Fig. 5 Long Edges 0 and 4 for 'House35'

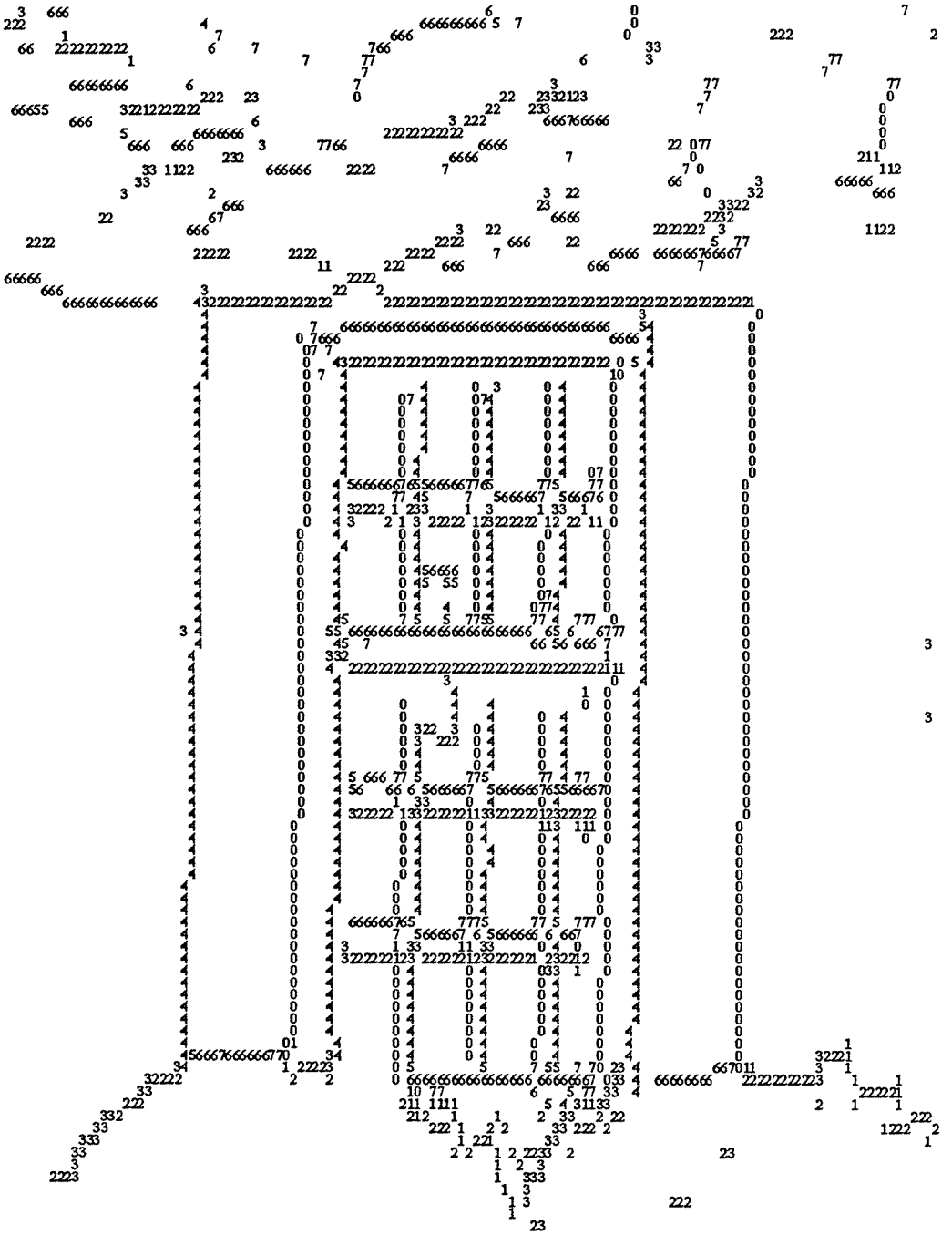


Fig. 6 Micro-Edge Map for the Lower-Left Window in 'House35'

Statistics on 'focus areas' enclosed by pairs of long edges 4-0:

area	μ	σ^2	top	bottom	left	right	
32	2.6	1.2	20	27	98	100	
40	1.5	1.2	36	39	49	58	
32	1.1	1.4	36	39	68	75	
20	1.1	0.9	44	47	38	42	--> upper window
20	1.3	1.7	40	47	72	73	
4	0.3	0.2	48	51	50	50	
72	1.6	1.3	40	51	95	100	--> upper window
56	0.2	0.4	52	55	18	31	
8	1.0	1.3	72	75	55	56	
8	0.0	0.0	76	79	55	56	
20	0.0	0.0	72	79	64	70	
24	0.0	0.0	72	83	29	30	--> lower shutter
60	2.4	0.9	72	83	36	40	--> lower window
24	0.0	0.0	72	83	45	46	--> lower shutter
32	0.0	0.0	72	83	88	90	--> lower shutter
76	1.8	1.4	72	83	95	100	--> lower window
8	0.8	0.7	84	87	45	46	
120	0.0	0.0	72	87	63	70	--> door
60	0.6	1.0	84	87	76	90	
32	0.0	0.0	72	87	105	106	--> lower shutter
32	0.5	0.8	88	91	63	70	

Statistics on 'focus areas' enclosed by pairs of long edges 0-4:

area	μ	σ^2	top	bottom	left	right	
8	1.3	0.7	40	43	45	46	
40	2.2	1.3	36	43	60	64	
4	0.0	0.0	44	47	35	35	--> dormer side
8	0.0	0.0	44	47	44	45	--> dormer side
64	2.4	0.8	40	47	109	116	
56	1.3	1.3	48	51	34	47	
148	2.3	1.0	40	51	53	68	
8	0.0	0.0	44	51	92	92	--> dormer side
12	0.0	0.0	44	51	103	104	--> dormer side
56	0.8	1.4	52	55	1	14	
4	0.0	0.0	64	67	118	118	
112	1.3	1.5	64	71	60	73	
4	0.0	0.0	72	75	52	52	
28	0.4	0.9	76	79	109	115	
44	1.8	1.2	84	87	32	42	
52	0.1	0.2	72	87	32	42	
48	0.0	0.0	72	87	59	61	--> door frame
40	0.0	0.0	72	87	72	74	--> door frame
12	1.1	1.4	88	91	72	74	

** σ^2 — variance.

*** The symbolic names pointed to by arrows are not program output at this stage of the recognition.

Fig. 7 (a) The Statistical Data for 'Focus Areas' in 'House35'

Statistics on 'focus areas' enclosed by pairs of long edges 4-0:

area	μ	σ^2	top	bottom	left	right	
24	1.2	0.8	44	47	103	108	
32	0.2	0.4	40	55	49	50	--> upper shutter
44	0.0	0.0	40	55	76	78	--> upper shutter
48	1.5	1.4	52	55	99	110	
48	0.1	0.2	68	83	55	57	--> lower shutter
96	0.3	0.4	68	83	71	76	--> lower shutter
28	0.1	0.1	72	83	90	92	--> lower shutter
64	0.1	0.1	88	95	52	59	

Statistics on 'focus areas' enclosed by pairs of long edges 0-4:

area	μ	σ^2	top	bottom	left	right	
44	0.2	0.3	72	75	25	35	
140	1.0	1.0	68	83	60	69	--> lower window
124	1.1	1.2	72	83	79	88	--> lower window
92	0.1	0.2	72	83	94	101	
48	0.2	0.3	84	91	24	29	
164	0.3	0.6	84	95	33	46	--> door way

** σ^2 — variance.

*** The symbolic names pointed to by arrows are not program output at this stage of the recognition.

Fig. 7 (b) The Statistical Data for 'Focus Areas' in 'House.sri'

Statistics on 'focus areas' enclosed by pairs of long edges 4-0:

area	μ	σ^2	top	bottom	left	right	
56	0.2	0.4	52	55	18	31	
20	0.0	0.0	72	79	64	70	
24	0.0	0.0	72	83	29	30	--> lower shutter
24	0.0	0.0	72	83	45	46	--> lower shutter
32	0.0	0.0	72	83	88	90	--> lower shutter
120	0.0	0.0	72	87	63	70	--> door
32	0.0	0.0	72	87	105	106	--> lower shutter
32	0.5	0.8	88	91	63	70	

Statistics on 'focus areas' enclosed by pairs of long edges 0-4:

area	μ	σ^2	top	bottom	left	right	
12	0.0	0.0	44	51	103	104	--> dormer side
28	0.4	0.9	76	79	109	115	
52	0.1	0.2	72	87	32	42	
48	0.0	0.0	72	87	59	61	--> door frame
40	0.0	0.0	72	87	72	74	--> door frame

*** The symbolic names pointed to by arrows are not program output at this stage of the recognition.

(a) Non-Tiny, Low-Edgeness Areas

area	μ	σ^2	top	bottom	left	right	
24	0.0	0.0	72	83	29	30	>> lower shutter
60	2.4	0.9	72	83	36	40	>> lower window
24	0.0	0.0	72	83	45	46	>> lower shutter
32	0.0	0.0	72	83	88	90	>> lower shutter
76	1.8	1.4	72	83	95	100	>> lower window
32	0.0	0.0	72	87	105	106	>> lower shutter
48	0.0	0.0	72	87	59	61	>> door frame
120	0.0	0.0	72	87	63	70	>> door
40	0.0	0.0	72	87	72	74	>> door frame

(b) Matched Window-Assemblies and Door-Assembly

Fig. 8 The Lateral Search results in 'House35'

Statistics on Edge Separation Measure:

μ	σ^2	top	bottom	left	right
2.0	0.0	54	63	88	110
2.0	0.0	68	79	72	83

Statistics on Edgeness Measure:

μ	σ^2	top	bottom	left	right
1.8	0.4	54	63	88	110
1.9	1.1	68	79	72	83

(a) house34

Statistics on Edge Separation Measure:

μ	σ^2	top	bottom	left	right
undef	undef	64	68	88	106
undef	undef	72	83	78	84

Statistics on Edgeness Measure:

μ	σ^2	top	bottom	left	right
0.2	0.4	64	68	88	106
0.2	0.3	72	83	78	84

(b) house35

Fig. 9 Edge Separation and Edgeness Measure on Wall Areas