HIERARCHICAL HOUGH TRANSFORM BASED ON PYRAMIDAL ARCHITECTURE

Cecilia Espinosa, and Marek A. Perkowski,

Department of Electrical Engineering,
Portland Center for Advanced Technology,
Portland State University, P.O. Box 751, Portland, Oregon, 97207.

ABSTRACT

In this paper a hierarchical Hough Transform (HT) based on pyramidal architecture is described, being a main component of the low-to-medium spatial vision subsystem for a mobile robot. It was implemented on a 386-based personal computer with 640k memory and it proved to give results of high quality as compared with the standard Hough transform implementation. The scheme is very well suited for parallelization.

1. INTRODUCTION

There is recently a renowed interest in Hough Transform (HT) theory and implementations [1,2,6,7,16,17,18,20,22,23,25,26,28,29,32,34,35], as well as in pyramidal architectures for image procesing [5,13,27,30,31]. While in 1983 the annual survey by Azriel Rosenfeld in *Computer Vision, Graphics and Image Processing* journal included only 5 positions on HTs, there have been more than 80 papers since 1988.

The primary goal of the reported research was the development of a simple, inexpensive and automatic low-to-medium-level image processing subsystem that is directed towards the development of a high-level vision system for a mobile robot. The most critical aspect of the development was the determination of the line feature extraction and characterization since this stage determines how the scene is characterized for storage or for use by the recognition task. The result of this stage is practically the basis of the vision capability of the mobile robot [33,12,21]. Since, because of real-time requirements, the method is to be ulimately implemented in a multiprocessor [1,5,7,32], the Hierarchical Hough Transform was chosen. Processing in image pyramides is naturally *parallel* and *recursive*. This approach overcomes the complications and disadvantages that are inherent to the standard HT. It gives also rise to the development of some data abstraction to simulate a pyramid structure so as to make the line extraction and characterization scheme implementable in a standard PC-386SX. The data structure also provided for the output of the features in terms of the *hierarchical levels*, a form which was found useful for the scene recognition task. From the point of view of typology of parallel computers for HT from [1] our scheme can be characterized as *"distributed image memory and distributed transform memory"* and is similiar to those from [1,28,32,7].

The feature extraction method used for a recognition system highly depends on the features of interest and the subsequent characterization amenable to the required recognition task. Image features are closely associated with object boundaries identified in the image. Extraction of such features were addressed in different ways depending on some prior knowledge of the type of objects expected in the image and the intended treatments of the extracted features. Scenes of the inside of a building are described in terms of characterizations of line features, e.g., their relative orientations, relative lengths, and relative positions. This paper centers the issue of feature extraction and characterization on straight line features.

The sequence of processing in our system originally conceived to be essential to the extraction of line features in indoor scenes consists of: Histogram Equalization, Smoothing with the use of the median filter, Edge Detection using the Sobel edge detectors, Binarization to extract the edges detected, Labeling, reBinarization and Thinning to refine the edges to thin lines and Line Extraction using a hierarchical approach to the HT method. The Binarization step includes the automatic determination of the binarization threshold from the gray level histogram of the image using the Between-Class-Variance method (BCV). The reBinarization step converts the labeled image back to the black and white state using a threshold of one [11,8,9]. It was the task of this research to establish the importance of each step for the success of the hierachical HT.

2. REVIEW OF LINE FEATURE EXTRACTION TECHNIQUES

Collinearity and proximity of edge points is of course the primary concern in linear feature extraction. Each researcher has his own way of imposing such constraints. The following works are but a few of the numerous efforts done on the extraction of linear features.

Kahn, Kitchen and Riseman [19] extracted lines by using their so-called connected components algorithm to group pixels with similar intensity gradients into line support regions and fitting lines to these regions. Burns [3] also grouped pixels of similar gradient orientations into line-support regions and used the structure of the associated intensity surface to determine the location and properties of the line. Nevatia and Babu [24] linked edge elements based on proximity and orientation and approximated the linked elements by piecewise linear segments. Hung and Kasvand [15] used the chain code and the difference codes on quantized thin lines to identify the "critical pixels" (pixels marking significant bends in the line) and used these critical pixels' positions to determine linear approximations to the lines. The chain code is a sequence of numbers generated by labeling pixels with direction numbers corresponding to a fixed set of orientations on a fixed grid. The sequence of numbers generated by taking the differences of successive chain code elements is the difference code which indicates the relative direction of the chain code segments. Shneier [31] made use of the pyramid structure in his line extraction process. His method involves building a series of successively lower resolution images from the original image, applying line-detector masks to each level followed by a line enhancement step and grouping the line-response points into line segments by means of a stepwise clustering process. His stepwise clustering process first groups points with similar direction and then subdivides each group on the basis of the separation between the points. The number of points in each subgroup determines the existence of the line. The same treatments are used for each pyramid level, therefore, each line in the lower resolution level corresponds to elongated lines in the original image. The extraction and representation allows for finding relevant areas in the image for further examination or processing. Among many advantages of pyramids for robot navigation is also that they allow for relatively easy determination of symmetries in pictures [4].

HT is a means of feature extraction from raster images originally patented by Hough [14] for identification of straight lines, but next generalized for curves by Duda and Hart [10], circles, ellipses [22], parabolas, algorithms that employ gradient direction and magnitude, arbitrary templates, and motion detection. It was shown that HT is a particular case of Radon Transform, and several extensions of HT were created. There is also much research on parallel and VLSI realizations of HT. HT is also used for determinig vanishing point from perspective images [23] which is used in our robot navigation.

The Fast Hough Transform of Li at al [20] uses a recursively divided parameter space to reduce memory and improve speed. Illingworth and Kitler's Adaptive HT approach [16] is based on stepwise incrementation of the resolution in parameter space. Wallace [34] proposed a "divide and conquer" approach to apply HT independently to parts of image. Risse [29] uses a quad-tree subdivision. Similar approach is also presented in [25]. An interesting approach to hierarchical HT is the work by Princen, Illingworth and Kittler [26]. Line segments are identified in small subimages using the conventional HT parameterization and these short line segments are grouped into longer ones at each higher level of the hierarchy. They used the overlapped pyramid structure for the hierarchical grouping. Another approach to hierarchical HT is presented in [6]. Pyramidal architectures for HT were also proposed in [18] and [2]. The pyramid structure was also used by Shneier [30,31], Rosenfeld [27], Hong [13] and many other authors in edge feature extraction and other applications, but employing much different techniques for extracting the lines and relating the higher level lines with their lower level components.

Of the linear feature extraction techniques, the HT method seems to be the most tolerant to missing edge points and random extraneous data which are almost always inherent in digitized real images. The HT method is also applicable to non-linear shapes, both for shapes described in terms of some parametric curves, or non-analytic shapes. Illingworth and Kittler [17] made a comprehensive review of the HT method and the researches done in the area. The review cited several desirable features of the method that make it superior to other boundary-based feature extraction techniques for shape and even motion analysis in natural images. In most cases natural images contain noisy, missing and extraneous data. Among the advantages are: (1) the method treats each edge point independently, making it possible to be implemented in more than one processing unit; (2) it combines events based on the transform space rather than the input image thereby making it tolerant of partial or slightly deformed shapes in the image; (3) it is very robust to the addition of random data produced by poor image segmentation; (4) it can simultaneously accumulate evidence for several occurrences of a particular shape in the image.

However, the standard implementation of the HT method entails large storage and computational requirements. The review described also some work done to overcome this drawback like the use of small-sized accumulators and the use of extra data to restrict the range of parameters which need to be addressed in the case of non-linear shapes.

We use here the parameterization based on the normal form of the straight line equation:

$$\rho = x \cos\theta + y \sin\theta \qquad (2.1)$$

where ρ is the length of the normal vector and θ is the angle the normal vector makes with the x-axis.

With the same principle as the slope-intercept parameterization, each point (x_i, y_i) maps therefore to a sinusoidal curve in the $\rho-\theta$ parameter space. When θ is restricted to the interval $(0, \pi)$, the parameter space sinusoids that describe all the points in the image space line will intersect at a unique point (ρ, θ) in the parameter space. Thus, an intersection point in parameter space defines the unique (ρ, θ) parameters of the image space line formed by the points. In a study of the discretization errors in the HT, Van Veen and Groen [35] suggested that the sampling is optimal when:

$$\Delta\rho = l \sin\frac{\Delta\theta}{2} \qquad (2.2)$$

where l is the length of the segment in the image space, $\Delta\rho$ and $\Delta\theta$ are the quantization intervals for the parameter space.

3. APPROACHES TO THE USE OF THE HOUGH TRANSFORM

A Standard HT Method

The simplest implementation of the method is by the parameterization of all the edge points in the whole image into a single accumulator array and scanning the array for counts greater than a threshold length set for a valid line. However, this requires a huge accumulator array if an optimal quantization interval is to be observed. The simplest way to implement the accumulator is with the use of an array type data structure, setting aside as much computer memory data space as required by the parameterization space. Implementation on a PC-based machine poses problems because of the way memory space is segmented according to the operating system used. Assuming that memory data space is limitless, still, allocating such huge space is not a wise engineering practice since only a number of the array cells will indicate the valid lines, some cells may not even get voted into, meaning a big chunk of the space is not really useful in the final analysis. A hash table or a linked list implementation may be more economical but these implementations may also present drawbacks in the ease and speed by which the cells may be accessed.

The transform's independent treatment of edge points may also be viewed as a weakness in the standard HT because it could result into accidental associations of edge points which are by accident collinear but really belongs to some other true line. Accidental associations may occur for example when we have several true lines which may all be intersected by one line that does not really exist. All the intersection points are of course collinear and would vote for the same cell in the accumulator array and if the number of points satisfy the threshold set for valid lines then the false line will be declared to exist. Also, line segments that happen to be collinear will be declared as one long line regardless of their spatial separation.

Finally, localization of the lines in the image space must be considered. Since the parameter values describe an infinite length line, finding the specific location and extent of the line in the image space may require some back-transformation to find the endpoints of the line.

The Hierarchical Approach

Splitting the whole image into smaller subimages and implementing the transform on the smaller subimages has three advantages: parallelization of tasks, smaller accumulator array, and the localization of the lines detected to the subimage region. Localizing the parameter space to each subimage reduces the possibilities of accidental associations of points into a line, and eliminates the combination of widely separated collinear segments into single lines. The location of the subimage region also provides the exact location of the lines in the image. This application of the HT method results in a number of short lines, each line localized to its subimage region.

Since the ultimate goal is to come up with a global description of the image in terms of the lines found in the image, it is then a matter of grouping the line segments into longer lines according to some collinearity and proximity constraints. To maintain the proximity requirement, a neighborhood of subimages is treated as a single subimage, and line segments detected in these subimages that happen to be collinear are grouped into longer lines. This results in a set of longer lines, each one localized to its bigger subimage region.

The process of grouping the lines in the neighboring subimages to form longer lines in the combined neighborhood is continued until no more neighboring subimages may be combined into larger areas or no more lines can be grouped together. Each grouping of lines in a neighborhoods of subimages into longer lines in the bigger subimage constitutes one level of the hierarchy. Thus, the lowest level of the hierarchy consists of the image containing the edge pixels, the first level consists of the line segments in each subimage of the input image, the second level consists of grouped first level line segments in neighboring subimages, and so on.

4. THE LINE EXTRACTION SCHEME

The Hierarchical Structure

The hierarchical approach uses the concept of the overlapped pyramid structure. Each level of the pyramid consists of lines found in the subimage components of the level. This representation makes it different from the conventional pyramid structure which consists of successively lower resolution images [30,6]. The pyramid concept was used in the way the subimages are grouped into bigger subimages to reflect the extent of collinear line segments in successive levels of the hierarchy. At each level, a neighborhood of 2×2 subimages in the lower level is treated as one subimage as shown in Fig. 4.1. The lowest level of the hierarchy consists of short line segments determined from subimages which comprise the binary line-thinned image resulting from the edge extraction operations. The pyramid is built from the bottom to the top by grouping line segments from 2×2 adjacent subimages (to be called central subimages) to form longer line segments in the next higher level, provided there is sufficient support for the existence of the line from within the central subimage region or from the immediate neighbors of the central subimage region. The subimages in the 4×4 neighborhood composed of the central subimages and their immediate neighbors will be called sibling subimages and the region corresponding to the span of the central subimages in the higher level will be referred to as the parent subimage. Fig. 4.2 shows a neighborhood of sibling subimages used to form longer line segments for a parent subimage.

It must be noted that the parent subimages define the subimage components of the new level. The central 2×2 regions that make up the parent subimage are disjoint. However, they share rows and columns with adjacent central subimage regions to provide support information for the confirmation of the line existence in the parent subimage. Fig. 4.3 shows the overlapped regions for the adjacent subimages. This technique was first proposed by Shneier [30] for his edge pyramid and was later used by Princen, et.al. [26] for line segment grouping.

Formation of new levels continues as long as new parent subimages may be formed from the lower level sibling subimages. The formation of the parent subimage is actually implemented in terms of the grouping of collinear line segments in the lower level sibling subimages. This results into a longer line segment that is indicated to exist inside the parent subimage.

The Data Structure Representation of the Hierarchy

The hierarchical structure is implemented as a hierarchy of data structures as shown in Fig. 4.4. The pyramid itself may be viewed as a list of levels (Fig. 4.4a) with the head of the list corresponding to the highest level. Considering the manner by which the line segments are grouped, the most obvious representation of each level is by an array of cells, each cell containing the line segment groups that exist in each subimage component of the level. However, because only longer line segments propagate to higher levels, it is highly probable that not all the subimage components will contain line groups. It was then decided to store only those subimage components that contain line groups. To preserve the topology of the subimage partitioning for the level, the lines are listed according to the order by which their subimage locations occur in the image. This was effected by representing each level as a *list of rowlists* (Fig. 4.4b). Each rowlist corresponds to a row of subimages of the partitioning, it contains all the line segments along the row (Fig. 4.4c). Each line segment in turn is attributed with the pointer to the local center of its subimage location, its normal parameters, and a list of the lowest level subimage regions that contain the subsegments comprising the line.

The Lowest Level Line Segment Determination

At the lowest level, the binary line-thinned image is partitioned into subimages of size L ($L \times L$ pixel neighborhoods). The line segment determination in each subimage involves the ρ–θ parameterization of each edge point, incrementing the appropriate accumulator cell for each parameter point found and finally scanning the accumulator cells for counts that satisfy the threshold count for valid lines.

As shown in Fig. 4.5, the expanse of the L-sized central subimage implies that only lines intersected by the corresponding normal vector inside the circular region may exist inside the central subimage. This imposes the parameter space limits for the valid lines:

$$-\frac{L}{2} \le \rho \le \frac{L}{2} \quad and \quad 0 \le \theta < \pi \tag{4.1}$$

However, the shortest possible line that may pass the edge of the circular region will not be detectable from the accumulator counts, even if it is a part of a longer line that happens to pass through the subimage. In order to make these lines detectable, it is necessary that the immediate neighborhood of the subimage be considered to provide a support information for the line. Thus, an overlapped region of size $2L$ that contains the subimage as the central part and $L/2$ rows and columns of pixels that immediately surround the subimage was considered for the HT. The use of the overlap contributes also to a more uniform distribution of detectable counts. This is because for the scheme with overlaps the ratio of the longest and the shortest line length, *length(AB)/length(CD) = 1.55*, which is much lower than the ratio of a scheme with no overlaps: *length(EF)/length(GH) = 3.41*. This uniformity helps in setting a safe threshold value for the counts that will be taken as indicative of valid lines, i.e. for GH to be detectable the threshold must not be greater than the length of GH, which however would be too small for EF. Whereas if the overlaps were considered, the threshold may be a reasonable fraction of AB as the threshold will already be able to detect enough counts for CD. Thus, with the overlap, imposing that 50% of the line must be within the region to be detectable is sufficient to detect the properly supported short segment, whereas, a much lower requirement must be set for the non-overlapped case. Setting a low number of pixels as a threshold will result into indications of short segments that are really parts of longer segments, thereby introducing redundant results.

The size of the region and the parameter space limits dictate the quantization intervals (θ_Δ, ρ_Δ) which must be sufficiently small to distinguish all possible valid lines in the region. The number of possible line orientations in the overlapped region is taken as $4L$. This requires the parameter space quantization intervals to be:

$$\theta_\Delta = \frac{\pi}{4L}, \quad \rho_\Delta = L \sin\frac{\theta_\Delta}{2} \tag{4.2}$$

Thus the accumulator array needed to represent the parameter space will consist of $(\frac{L}{\rho_\Delta} + 1) \times 4L$ cells.

It must be recalled that the normal parameters (ρ, θ) are referenced to the center of the x–y coordinate system. In a similar sense, the (ρ, θ) parameters found for the collinear edge points in the subimage are referenced to the local center of the subimage. For the benefit of notational convenience, let the local parameters of the line segment be denoted as (ρ_o, θ_o), the local coordinates of the edge points as (x_l, y_l), and the local subimage center as (x_s, y_s). Thus, the detected line segments may be thought of as collinear edge points that satisfy (2.1), i.e.:

$$\rho_o = x_l \cos\theta_o + y_l \sin\theta_o$$

Higher Level Grouping

Line segments from neighborhoods of 4×4 subimages containing the 2×2 central subimages are grouped into longer line segments, provided that the line segments are collinear in a sense that will be explained shortly.

Each line segment detected in a lower level may be described as a feature point at the intersection of the line and its normal in the lower level subimage region, i.e.,

$$x_o = \rho_o \cos\theta_o, \quad y_o = \rho_o \sin\theta_o.$$

Fig. 5.6 shows a neighborhood of subimages containing a straight line represented as feature points marked at the foot of the normal (x_o, y_o), local to each subimage involved. Since each feature point is represented with respect to its local origin (x_s, y_s) and must be now considered with respect to the center of the parent subimage (x_p, y_p), the appropriate parameter space curve is determined by adjusting the x and y of (2.1) for the local centers of the subimages as:

$$\rho = (x_o - (x_p - x_s))\cos\theta + (y_o - (y_p - y_s))\sin\theta. \tag{4.3}$$

The conventional HT parameterization scheme is used to find the collinear line segments in the neighborhood of the feature points using (4.3) for the ρ-θ mapping of the feature points. The parameterization introduces, however, discretization errors, $\Delta\rho$ and $\Delta\theta$ (different from the sampling intervals ρ_Δ and θ_Δ). $\Delta\rho$ is taken to be $\sqrt{2}$, the minimum bar width that can include all edge points for a line of any angle if the line is drawn on a discrete grid with a point spacing of one pixel. $\Delta\theta$ is the θ sampling interval used in the lower level. The possible ρ, θ values for a feature point (x_o, y_o) in the new parameter space will be therefore given by:

$$\theta_o - \frac{\Delta\theta}{2} \le \theta \le \theta_o + \frac{\Delta\theta}{2}, \tag{4.4a}$$

$$\rho_o - \frac{\Delta\rho}{2} \le \rho \le \rho_o + \frac{\Delta\rho}{2}. \tag{4.4b}$$

This sets the collinearity constraints for the feature point (x_o, y_o), meaning that any other feature point that would have a (ρ, θ) value within this range is collinear with the point (x_o, y_o). The bounds given by (4.4a) and (4.4b) narrow down the range of the parameter space that need to be examined for collinear feature points.

Just like the lowest level scheme, the feature points in the sibling subimages are parameterized, this time according to (4.3) and only within the range of the collinearity constraints for the feature point. The appropriate accumulator cells increased by one for each parameter point found. Similarly, a longer line segment is detected, if there are collinear line segments in at least two neighboring subimages of the 4×4 neighborhood. Line segments identified to be collinear are grouped together and treated as a single longer line segment, this longer segment is attributed now with its local parent subimage center, represented as its (x_s, y_s), its local parameters (ρ_o, θ_o), and a list of member subsegments. Each member subsegment is represented as a pointer to the center of the lowest level subimage region that contains the subsegment. The segments that become members of the new groups at the higher level are deleted from the lower level, and their lists of member subsegments are now linked to the new group's subsegment list. The grouping scheme terminates shorter lines at lower levels in the hierarchy, and allows only longer lines to participate at higher level groupings, thereby reducing the amount of calculations needed at the higher levels.

The quantization intervals are set in the same manner as was used in the lowest level according to (4.2) where the L is now the size of the central region which is twice the size in the preceding level. The assumed error bound $\Delta\theta$ given by (4.4a) and the sample spacing θ_Δ implies that only five samples of θ are relevant for each feature point. The width $\Delta\rho$ from (4.4b) and the sample spacing ρ_Δ imply that the number of cells that need to be incremented along the ρ axis for each θ value is equal to the rounded off value of $\frac{\sqrt{2}}{\rho_\Delta} + 1$. (The sparsity of the feature points from the sibling subimages suggests also that the conventional accumulator array (which indeed grows larger as the level in the hierarchy increases) may be represented as a linked list with elements for only the relevant cells).

5. THE IMPLEMENTATION OF THE HIERARCHICAL HOUGH TRANSFORM SCHEME

The hierarchical scheme for the line feature extraction and characterization using the HT technique is divided into 2 routines:

(1) the lowest level line segment determination from the binary line-thinned image;

(2) the high level line segment grouping.

To simplify references to the routines, let *hhlow* refer to the lowest level line segment determination routine; and *hhigh* refer to the high level line segment grouping routine. The square of $2^n \times 2^n$ pixels will be called an *exp2 square*. The smallest exp2 square that may encompass the input image is first determined and the image centered in this square. This is necessary for the uniform partitioning of the image into subimage regions at each level of the hierarchy. The main algorithm may be summarized as follows:

(1) center input image in smallest exp2 square;

(2) call *hhlow* to generate the first level, given the exp2-square-centered input image.

(3) call *hhigh* to generate the second level given the first level;

(4) as long as a new level is generated, repeat calling *hhigh*, each time giving it the last level generated.

The program implementing this algorithm was written in C++ and runs on PC-386SX under Turbo C++ compiler [11]. It is available on request from Portland State University together with all other pre-procesing stages and test images.

The Lowest Level Line Segment Determination Implementation

The lowest level line segment determination scheme described is implemented according to the following pseudocode:

```
/* lowest level line segment determination */
/* begin hhlow */
{ Do initialization steps
  while (an image row is part of subimage row)
  { Set up the subimage row in buffer
    Set up first subimage
    while (an image column is a part of the central subimage)
    { List edge points found in overlapped subimage
      if ((edge point count >= threshold) && (central points count > 0))
      { Parameterize edge points
        FindPeaks in the accumulator array
        if (linesegments found)
          link up linesegments into linelist
      }
      Set up next subimage
    }
    if (linelist is not empty)
    { attach rowmarker to head of linelist
      if (is first rowlist)
        mark as head of rowlists
      else
        link up to rowmarker of last rowlist
    }
    Shift out upper L rows in buffer, shift up lower L rows
  }
  return (list of rowlists)
}
/* end of hhlow */
```

The initialization steps consist of setting up buffers to hold a row of overlapped subimages, setting up the space for the accumulator array, setting up the sine-cosine lookup table for use in the parameterization, setting up pointers to the first subimage row in the image row buffers and setting up the Hough Transform parameter space settings. Buffering the image rows that comprise the subimages avoids frequent access to the image memory during the processing of the subimages. It was considered reasonable to set up the sine-cosine look-up table for the parameterization so as to minimize calls to the sine and cosine functions in the compiler's math library. It may be noted that each edge point would have to be sampled for all possible θ in the parameterization range. Since the line segment determination is done for each subimage separately, the same accumulator array may be re-used for each subimage.

Subimage settings. The subimage size used for the lowest level line segment determination is $L = 4$, meaning, the exp2-square-centered binary line-thinned image is partitioned into 4×4 pixel wide subimages. This means that an overlapped region for a subimage is a 8×8 pixel neighborhood with the 4×4 pixel subimage centered in it.

Hough Transform parameter space settings. Based on the discussion given earlier, the parameter space sampling intervals are as follows:

$$\theta_\Delta = \frac{\pi}{4L} = \frac{\pi}{16}$$

$$\rho_\Delta = L \times \sin\frac{\theta_\Delta}{2} = 0.40$$

Therefore the quantized parameter space settings are:

	number of intervals	range
theta axis:	$4L = 16$	$[0, \pi]$
rho axis:	$\frac{L}{\rho_\Delta} + 1 = 11$	$[-2, +2]$

A 11×16 accumulator array was used to represent the quantized parameter space, each cell corresponding to a point in the quantized space. Since array cell indices are always positive integers, the ρ-axis need to be shifted so that all the ρ-ordinates may be addressed in terms of the row indices. This requires that each of the ρ values computed in terms of the quantized space (ρ_q) will have to be adjusted by:

$$\rho_q' = \rho_q + \rho_{zero}$$

where ρ_{zero} is the index for the middle row that corresponds to the center of the quantized ρ axis ($\rho_q = 0$).

The threshold for the counts that indicate a valid line is set at 4, assuming that at least 50% of the points on the line must exist in the overlapped image to indicate this line as valid.

Line segment determination in the subimage. Prior to the parameterization, the number of edge points in the overlapped subimage is first counted. If the number of edge points is at least equal to the threshold set, and there is at least one edge point in the central subimage, then the subimage is subjected to all the rigors of the HT method, otherwise the subimage is simply discarded. As was stressed in section 4, each edge point is sampled for each θ in the quantized space, the appropriate accumulator cell incremented for each (ρ, θ) determined, and the accumulator is finally scanned for counts that satisfy the threshold. Each identified line segment is represented as a list element containing its local (ρ_o, θ_o) and the local center of its subimage region (x_s, y_s), which is then linked to the list of segments found in its subimage row.

Level 1 of the hierarchy is therefore composed of at most 16 lists of line segments for the 256×240 image, each list corresponding to each row of subimages.

Higher Level Line Segment Grouping Scheme Implementation

The grouping of collinear line segments in a neighborhood of lower level subimages is very similar to the scheme used for the low level line segment determination. This time the elements of the subimage are no longer image pixels but feature points representing the lower level lines. A 4×4 array of pointers to line segment nodes in the lower level rowlists keeps track of the component feature points for each of the 4×4 sibling subimages that comprise the overlapped subimage. The inner 2×2 pointers keep track of the feature points in the central subimages. Given the list of rowlists that constitutes the last level, *hhigh* returns a list of rowlists for the new level formed. The pseudocode for the implementation of the high level grouping scheme is as follows:

```
/* higher level line segment grouping */
/* begin hhigh */
{ while (lowlevel rowlist is part of central subimage row)
  { Set up pointers to lowlevel rowlists comprising the subimage row
    while (not end of all the lowlevel rowlists of row)
    { Set up the siblings window
      if ((feature points >= threshold) && (central point > 0))
      { Label central points (line segment nodes in central siblings)
        for (each feature point is in a central siblings)
        { Get collinearity votes from immediate sibling point
          Form a group for the highest voted (ρ, θ) cell
          if (group found)
          { mark feature points as "grouped"
            link up new line segment into linelist
          }
          else
            mark the feature point as "not grouped"
        }
        Finalize Labels of central points
      }
      Shuffle pointers to the next siblings window
    }
    if (linelist not empty)
    { get rowmarker for linelist
```

```
        if (first rowlist)
          mark as head of rowlists
        else
          link up new rowlist to head of last rowlist
      }
      Delete grouped segments from the upper 2 lowlevel rowlists
      Shove out the upper 2 lowlevel rowlists, include the next 2 rowlists
    }
    Delete grouped segments in the last group of lowlevel rowlists
    if (list of rowlists not empty)
      Delete rowmarkers to empty lowlevel rowlists
    return (list of new rowlists)
  }
  /* end of hhigh */
```

The subimage size at each new level is the combined size of the 2×2 neighborhood of central sibling images. Thus, the subimage size L for a new level is simply twice the size in the last level.

The quantized parameter space settings are determined in the same manner as that in the lowest level. TABLE I presents a summary of the quantized parameter space settings for each of the levels for a 256×240 input image centered in a 256×256 square.

TABLE I

SETTINGS FOR THE HOUGH TRANSFORM QUANTIZED PARAMETER SPACE

level	subimage size L	sampling intervals θ_Δ	ρ_Δ	no. of intervals θ	ρ	ρ range $-L$ -> $+L$	ρ_{zero}
1	4	$\frac{\pi}{16}$	0.4	16	11	-2 -> +2	5
2	8	$\frac{\pi}{32}$	0.4	32	21	-4 -> +4	11
3	16	$\frac{\pi}{64}$	0.4	64	41	-8 -> +8	21
4	32	$\frac{\pi}{128}$	0.4	128	81	-16 --> +16	41
5	64	$\frac{\pi}{256}$	0.4	256	161	-32 -> +32	81
6	128	$\frac{\pi}{512}$	0.4	512	321	-64 -> +64	161
7	256	$\frac{\pi}{1024}$	0.4	1024	641	-128 -> +128	321

The tabulation shows that as the level increases the size of the accumulator array increases as well. However, as explained earlier, only the θ values within $\pm\frac{1}{2}\Delta\theta$ and the ρ values within $\pm\frac{1}{2}\Delta\rho$ need to be sampled for each feature point. Since $\Delta\theta$ is one half of the θ_Δ of the previous level, and $\Delta\rho$ is a constant of 0.707, then only 5 samples need to be taken for the θ dimension, and 3 samples for the ρ dimension. Thus, the accumulator array cells just required for the sampling were considered and represented as a list of 5 sets of ρ cells. Each set of ρ cells represents the 3 consecutive cells in the conventional array that lie within the range of the ρ values that need to be sampled. Each cell consists of a counter field for the accumulation, and a field where the identity of contributors to the count are noted. This provides a way of tracking the feature points that belong to a group, if ever one is indicated. The same space was used for the groupings in all the levels, each time initializing the space by assigning the appropriate accumulator array coordinates to the cells, according to the range established by the feature point's collinearity constraint. For simplicity, let us refer to this subspace of the parameter space as the voting array.

Similar to the low level scheme, there must be enough feature points in the sibling subimages, and there must be at least one in the central region for the line segments to be considered for grouping. Each feature point establishes the range of ρ and θ that need to be used for the sampling. For each central feature point, each point in the immediate neighboring sibling subimages is sampled. It is sampled under condition that the ranges of θ of both points overlap. The appropriate cell in the voting array is incremented by one for every ρ, θ that is within the range of the voting array. The "identity" of the voter is also noted in the cell if it is a central feature point. A group corresponding to a longer line segment is formed when a cell indicates more than one vote. Since there are 15 possible lines that the feature point can be assigned to, and each neighboring feature point may vote for these possibilities as its own collinearity constraints allow, it is then always possible that several cells may accumulate more than one vote. Thus, to get the best line description of a new group, the voting array is searched for the cell that obtains the highest vote. The parameter space point represented by the cell is then taken as the set of ρ, θ parameters for the new group. Since we are now dealing with short line segments that are being combined into longer line segments, a feature point is allowed to be a part of only one group. To ensure this, all the central feature points that get included in a group are appropriately marked as "grouped" feature points, so that they will no longer be considered in the formation of other groups within the same parent subimage.

Each feature point must be initially labeled to be able to keep track of the identity of the central feature points. This also differentiates the central feature points from those of the supporting sibling subimages. It must be recalled that the feature points in the sibling subimages outside of the central subimage region simply provide support information for the existence of a group, but are not included in the group formed for the parent subimage. The label assigned to a central feature point changes during the grouping process, depending on whether the feature point becomes a group member or not. The labels are finally changed after all the central feature points were considered to mark the feature point for the deletion from its lower level, if it became a group member in the higher level, or for the retention in its level if its presence was not indicated in the new level.

When a line segment becomes a member of a new group, its list of subsegments is detached from the node used to represent the line segment and linked up to the new group's subsegment list. After a new rowlist is completed, the last level's rowlists used for the grouping are cleared of the elements that were indicated as "grouped" and have no more subsegment lists anyway. These elements are detected by the finalized labels. Using the labels as the basis for deletion, rather than the subsegment lists, allows the use of the same cleaning up treatment for all levels, including the cleaning up of the first level lines which do not have the subsegment lists. Finally, when the whole level is completed, the last level is cleared of empty rowlists, thereby cleaning up the remnants of the segments that have propagated to longer lines, and leaving only those lines that really belong to the level.

The images in Fig. 5.1b to Fig. 5.1h show the lines formed at each level when the aforementioned scheme was applied to the binary line-thinned image shown in Fig. 5.1a.

The first level remap (Fig. 5.1b) shows that isolated streaks shorter than 4 pixels were not detected, which was of course expected. It is also noticeable that small discontinuities and/or irregularities in the lines were smoothed out, exhibiting the robustness of the HT method to missing data and to irregularities in the data.

The higher level remaps show that longer lines indeed propagate to higher levels. However, there are also short lines that propagate up regardless of their lengths. These are the lines that lie across the boundaries of adjacent subimages that do not get grouped into one parent subimage until at a much higher level. All throughout the lower level groupings, the part of the line that lies in one subimage consistently supports the other part located in other subimage. This results in the propagation of both parts to the level where their subimages become parts of a central subimage region. This may be attributed to the constraint that was set for the detectability of a line group. For this implementation, a line group is considered detectable or is indicated to exist if there are at least two feature points that are collinear, one feature point located in a central subimage and the other point lying in an immediate neighboring subimage which may be a central subimage or a sibling subimage outside of the central subimage region. Because of this constraint, the propensity of the HT method for combining feature points that are collinear in adjacent subimages into a single line, regardless of the their discontinuity in the lower level, is exhibited in the formation of the level 5 lines.

6. IMPACT OF THE INITIAL SUBIMAGE SIZE ON THE LINE FEATURE EXTRACTION

The higher level grouping simply takes the line segments available in the lower level for consideration. It is obvious that the amount of detail that may be extracted depends highly on the lowest level line segment extraction from the input image. The amount of detail is determined by the initial subimage size into which the input image is partitioned, noting that the initial proximity constraint imposed by the circular region in the central subimage and the threshold for the shortest detectable line are determined by the subimage size. A smaller subimage size means a smaller accumulator array, which is a good point, but a smaller subimage size would also mean more grouping levels and more line segment nodes to handle at each level. TABLE II presents the effects of the initial subimage size on the processing times for the lowest level line segment determination and the higher level groupings when the scheme was applied to the line-thinned images in Fig. 6.1a and Fig. 6.1b. Presented in TABLE III are the number of line segments identified at each level before and after the grouping.

TABLE II

PROCESSING TIMES FOR THE LINE FEATURE EXTRACTION WITH DIFFERENT INITIAL SUBIMAGE SIZES

	Image1			Image2		
	4	8	16	4	8	16
hhlow	31	51	101	35	64	130
hhigh	53	15	5	66	24	11
total	84	66	106	101	88	141

TABLE III

NUMBER OF LINES EXTRACTED BY THE HIERARCHICAL SCHEME AT DIFFERENT INITIAL SUBIMAGE SIZES

	Image1						Image2					
size	4		8		16		4		8		16	
level	i	f	i	f	i	f	i	f	i	f	i	f
1	1090	107	307	14	116	9	1379	57	470	21	211	9
2	562	34	174	12	63	2	729	30	254	7	112	10
3	328	37	99	14	35	3	413	24	141	13	63	3
4	177	31	51	9	19	3	222	25	80	9	37	7
5	88	17	27	5	8	8	114	15	47	14	15	15
6	44	13	11	11	-	-	60	14	16	16	-	-
7	14	14	-	-	-	-	20	20	-	-	-	-
total		253		65		25		185		80		44

The results show that a smaller subimage size requires lesser processing time for the lowest level line segment determination, but incurs more time in the higher level grouping. The timing results indicate that the optimal subimage size is 8. The lowest level remaps for the image in Fig. 6.1a shown in Fig. 6.2a, Fig. 6.2b and Fig. 6.2c for subimage sizes 4, 8, and 16, respectively, show that size 4 made the best extraction of the diagonal line defining the corridor outline, whereas a considerable part of this line was not detected with size 8 and the line was totally not visible with size 16. A clear outline of the corridor was propagated to level 6 (1 step from the top level) for the size 4 case, whereas a part of the corridor detected with size 8 was propagated to level 4 (2 steps from the top level) as shown in Fig. 6.3a and Fig. 6.3b, respectively.

Thus, size 4 was deemed to be a good initial subimage size despite the longer processing time involved considering the credibility of the features extracted and made visible at the higher levels.

7. EVALUATION OF THE RESULTS

Presented in TABLE IV are the processing times for each of the operations when applied to Image1 shown in Fig. 7.1 as implemented in a PC-386SX.

TABLE IV

EXECUTION TIMES FOR THE IMAGE PROCESSING OPERATIONS

operation	t(sec)	% of total time
Histogram Equalization	20	7.02
Smoothing	59	20.70
Edge Detection	50	17.54
Binarization	18	6.32
Labeling & reBinarization	23	7.72
Thinning	30	10.53
Line Extr. & Charac.	86	30.17
(total)	285	100

A total processing time of 285 seconds is indeed unattractive for the real-time application envisioned for the system. To minimize the processing time, the necessity or redundance of the operations is evaluated based on their impact on the scene description extracted and the processing times involved. Among the operations, Histogram Equalization, Edge Detection, Binarization, Thinning and Line Feature Extraction and Characterization are considered indispensable. However, Smoothing, Labeling and reBinarization are considered enhancement steps. Smoothing was used to reduce noise in the digitized image, hopefully to improve the detectability of the edges and to prevent the detection reduce the detection of false edges. Labeling and reBinarization are supposed to get rid of small spots or streaks in the binarized image, thereby reducing the number of pixels that will be processed by the Thinning and Line Extraction steps. Noting the processing times, Histogram Equalization, Smoothing, Edge Detection and Binarization have constant execution times since the operations process all the image pixels regardless of the image quality or the density of edge features in the image. On the other hand, the processing times for Labeling, Thinning and Line Extraction are affected by the amount of pixels comprising the extracted edges. The evaluation is therefore centered on the examination of the factors that affect the execution time of the Thinning and the Line Extraction steps and the impact of the Smoothing and Labeling steps on the execution times and the resulting quality of the scene features extracted [11].

The impact of the smoothing and labeling processes are determined by their effects on the Line Feature Extraction step. TABLE V shows their effects on the processing times and TABLE VII shows the number of lines identified at each level before and after the line segment grouping.

TABLE V

EFFECTS OF SMOOTHING AND LABELING ON THE PROCESSING TIMES OF THE LINE FEATURE EXTRACTION ROUTINES

	Smoothed Image		unSmoothed Image	
	with Labelling	no Labelling	with Labelling	no Labelling
hhlow	33	31	41	36
hhigh	59	53	95	84
hpost	2	2	2	2
total	94	86	138	122

TABLE VI

EFFECT OF SMOOTHING AND LABELING ON THE NUMBER OF LINES EXTRACTED

Level	Smoothed Image				unSmoothed Image			
	with Labelling		no Labelling		with Labelling		no Labelling	
1	1090		107		1194		127	
2	562	34	611	45	773	78	863	91
3	328	37	355	43	443	67	491	84
4	177	31	191	35	240	51	262	54
5	88	17	96	15	126	31	138	38
6	44	13	49	16	54	22	56	24
7	14	14	15	15	14	14	14	14
total	253		296		411		496	

The use of the Labelling step for the removal of small blobs or streaks that may thin out to very short lines which will not be detectable anyway, improves the total execution time for both the smoothed and unsmoothed image. The figures for the number of lines detected in the lowest level however show that some lines were made undetectable by the Labelling step. This phenomenon was cited in the subimage size determination analysis as caused by the tendency of ends of the edges to get eroded because of the Labelling step's negligence of vertex-connectivity in adjacent blobs. Also, weak edges that tend to appear as a series of aligned streaks are of course eliminated. So in a sense, the Labelling step causes some losses in the detectable lines especially if the extracted edges that will yield the line are rather weak.

Elimination of the Smoothing step increased the processing times even more, because of the increased number of white pixels that need to be parameterized. The number of lines formed at the lowest level is notably much higher than those formed with the smoothed image. Since the HT is not concerned about connectivity of pixels as long the right number of pixels vote for a parameter pair, then, the noise spots that cluster within a span of the initial subimage size are contributors to a line if the number is right. But as long as the lines formed by the noise spots are rather isolated from the lines indicated by the true edges, then there are no complications since these noise lines will never propagate into the higher levels. The problem arises when the noisy spots cluster around the edges and form lines with the real edge pixels. Remaps of the results show that the lines are indeed formed by the noise clusters and get linked with real lines during the grouping so that the now-adulterated line becomes longer and propagates to higher levels.

Because of the propensity of noise to gather around edges, and because of the high regard of the HT scheme for collinear points regardless of their connectivity, it is not really safe to disregard the initial noise removal from the processing sequence. However, the Labeling step is redundant since the small pixel blobs which the labeling step removes are discarded just the same by the lowest level line segment determination step. Though there is an increase in the line extraction step processing time, the increase is still less than time incurred for the Labelling and the reBinarization. The tendency of the Labelling routine to cut off edges is indeed detrimental to the extraction of the desired features especially when the edges are rather weak because of poor image quality or inadequate lighting.

8. CONCLUSIONS AND FUTURE WORK

In the reported research, for the first time the entire low-to-medium image processing for a mobile robot system was completely implemented in a standard 386 PC. The crux of this research is the development of a feature extraction technique that provides the scene recognition system a credible and easy to reconstruct description of the scene. The hierarchical approach to line extraction using the basic concepts of the HT method and pyramids offset the weaknesses of the HT and allow the exploitation of its strong points towards a robust line feature extraction scheme. The program developed was proven to efficiently extract line feature descriptions of real indoor scenes even with relatively poor image quality. Now it is about time to improve the real-timeliness of the system. We plan to develop an inexpensive parallel "on-board" system for the robot that will be entirely built from commercially available 486-based PC motherboards and DSP co-processors. The nature of the algorithms developed makes them very suitable for this system.

9. REFERENCES

[1] Ben-Tzvi, D., A. Naqvi, and M. Sandler, "Synchronous Multiprocesor Implementation of the Hough Transform", *Comp. Vision, Graphics, and Image Proc.*, Vol. 52, pp. 437-446, 1990.

[2] Bongiovanni, G., C. Guerra, and S. Levialdi, "Computing the Hough Transform ona Pyramid Architecture", *Machine Vision and Applications*, Vol. 3, pp. 117-123, 1990.

[3] Burns, J. B., A.R. Hanson, and E.M. Riseman, "Extracting Straight Lines", *IEEE Trans. on Patt. Anal. and Mach. Intell.*, Vol. PAMI-8, No. 4, July 1986, pp. 425-455.

[4] Burton, F.W., "An Implementation of the Exponential Pyramid Data Structure with Application to Determination of Symmetries in Pictures", *Comp. Vision, Graphics, and Image Proc.*, Vol. 25, pp. 218-225, 1984.

[5] Cantoni, V., and S. Levialdi (Eds.), "Pyramidal Systems for Computer Vision", *Springer-Verlag*, Berlin, 1986.

[6] Carrioli, L., M. Diani, and L. Lombardi, "Hierarchical Hough: Advantages and Automatic Construction of Models", *Intern. Conf. on Pattern Recogn.*, pp. 791-794, 1988.

[7] Chuang, H.Y.H., and C.C. Li, "A Systolic Array Processor for Straight Line Detection by Modified Hough Transform", in *IEEE Conf. on Comp. Arch. for Pattern Anal. and Image Database Manag."*, Miami Beach, Florida, Nov. 1985, pp. 300-304.

[8] Dawson, B.M., "Introduction to Image Processing Algorithms", *BYTE*, March 1987, pp. 169-186.

[9] Dawson, B.M., "Changing Perceptions of Reality", *BYTE*, Dec. 1989, pp. 293-304.

[10] Duda, R.O., and P. E. Hart, "Use of Hough Transformation to Detect Lines and Curves in Pictures" *Comm. ACM.*, Vol. 15, No. 1., pp. 11-15, Jan. 1972.

[11] *Espinosa*, C., and M.A. Perkowski, "Low to Medium Level Image Processing System Based on Hierarchical Hough Transform", *Report, Department of Electrical and Computer Engineering*, Portland State University, Portland, Oregon, 1991.

[12] Giralt, G., "Mobile Robots", *Robotics and Artificial Intelligence*, NATO ASI Series Vol. F11, Springer-Verlag Berlin Heidelberg, 1984, pp. 365-393.

[13] Hong, Ts-H, M. Shneier and A. Rosenfeld, "Border Extraction Using Linked Edge Pyramids", *IEEE Trans. on Systems, Man, and Cybern.*, Vol. SMC-12, No. 5, Sept/Oct 1982, pp. 660-668.

[14] Hough, P.V.C., "Method and Means for Recognizing Complex Patterns", *U.S. Patent 3,069,654*, Dec. 18, 1962.

[15] Hung, S.H.Y. and T. Kasvand, "Linear Approximation of Quantized Thin Lines", *Pictorial Data Analysis*, (ed: Robert M. Haralick), NATO ASI Series Vol. F4, Springer-Verlag, Berlin Heidelberg, 1983, pp. 15-28.

[16] Illingworth, J. and J. Kittler, "The Adaptive Hough Transform", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9, No. 5, 1987, pp. 691-698.

[17] Illingworth, J. and J. Kittler, "A Survey of the Hough Transform", *Computer Vision, Graphics, and Image Processing*, Vol. 44, No. 1, Oct. 1988, pp. 87-116.

[18] Jolion, J.M., and A. Rosenfeld, "An O(log n) pyramid Hough Transform", *Pattern Recognition Letters*, Vol. 9, pp. 343-349, 1989.

[19] Kahn, P., L. Kitchen and E.M. Riseman, "A Fast Line Finder for Vision-Guided Robot Navigation" *IEEE Trans. on Pattern Anal. and Machine Intell.*, Vol. 12, No. 11, Nov. 1990, pp. 1098-1102.

[20] Li, H., M.A. Lavin, and R.J. LeMaster, "Fast Hough Transform", *Res. Rep. RC 11080 (#49754), IBM T.J. Watson Res. Center*, March 1985.

[21] Madarasz, R.L. and L.C. Heiny, "Visual Navigation Techniques for Indoor Transport Systems", *Proc. 2nd World Congress on Robotics Research*, 1985.

[22] Milenkovic, V.J., "Multiple Resolution Search Techniques for the Hough Transform in High Dimensional Parameter Spaces", in *Techniques for 3D Machine Perception*, (A. Rosenfeld, ed.), pp. 231-254, Elsevier 1986.

[23] Magee,M.J., and J.K. Aggarwal, "Determining Vanishing Points from Perspective Images" *Comp. Vision, Graphics, and Image Proc.*, Vol. 26, 1984, pp. 256-267.

[24] Nevatia R., and K. R. Babu, "Line Feature Extraction and Description", *Comp. Graphics and Image Proc.*, Vol. 13, No. 3, July 1980, pp. 257-269.

[25] J. O'Rourke and K.R. Sloan, "Dynamic Quantization: Two Adaptive Data Structures for Multidimensional Spaces", *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-6, No. 3., pp. 266-279, 1984.

[26] Princen, J., Illingworth, J., and J. Kittler, "A Hierarchical Approach to Line Extraction Based on the Hough Transform", *Comp. Vision, Graphics, and Image Proc.*, Vol. 52, No. 1, October 1990, pp. 57-77.

[27] Rosenfeld, A., "Quadtrees and Pyramids: A hierarchical representation of images", *Pictorial Data Analysis*, (ed: Robert M. Haralick), NATO ASI Series Vol. F4, Springer-Verlag, Berlin Heidelberg, 1983, pp. 29-42.

[28] Rosenfeld, A., J. Ornelas, and Y. Hung, "Hough Transform Algorithm for Mesh-Connected SIMD Parallel Processor", *Computer Vision, Graphics, and Image Processing*, Vol. 41., pp. 293-305, 1988.

[29] Risse, T., "Yet Another Line Parametrization for Hough Transform", *Proc. of the 10th DAGM Symp. on Pattern Recogn.*, Zuerich, Sept. 1988, Informatick Fachberichte Vol. 180, pp. 142-150, Springer-Verlag, Berlin, 1988.

[30] Shneier, M., "Two Hierarchical Linear Feature Representations: Edge Pyramids and Edge Quadtrees", *Comp. Graphics and Image Proc.*, Vol 17, No. 3, Nov. 1981, pp. 211-224.

[31] Shneier, M.O., "Extracting Linear Features from Images Using Pyramids", *IEEE Trans. on Systems, Man, and Cybern.*, Vol. SMC-12, No. 4, July/August 1982, pp. 569-572.

[32] Silberberg, T.M., "The Hough Transform on the Geometric Arithmetic Parallel Processor", in *IEEE Conf. on Comp. Arch. for Pattern Anal. and Image Database Manag."*, Miami Beach, Florida, Nov. 1985, pp. 387-393.

[33] Stanton, K., et. al., "PSUBOT - A Voice-Controlled Wheelchair for Handicapped", *Proc. of the Intern. Midwest Symp. on Circuits and Systems*, Aug. 1990, pp. 669-672.

[34] Wallace, R.S., "A Modified Hough Transform for Lines", in *Proc. Comp. Vision and Pattern Recogn. Conf. 1985*, pp. 665-667, IEEE Comp. Soc. Press, New York, 1985.

[35] Van Veen, T. M. and F.C.A. Groen, "Discretization Errors in the Hough Transform", *Pattern Recognition*, Vol. 14, 1981, pp. 137-145.

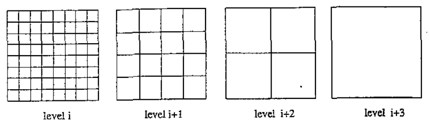

Fig. 4.1. Grouping of 2×2 subimage neighborhoods into bigger subimages.

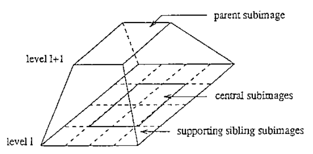

Fig. 4.2. Neighborhood of subimages used to form line segment groups for a parent subimage.

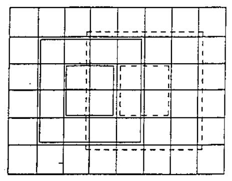

Fig. 4.3. Overlapping regions for adjacent central subimages.

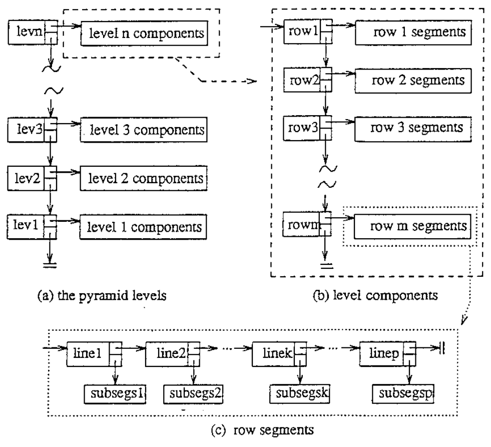

Fig. 4.4. Hierarchical representation of the hierarchical structure.

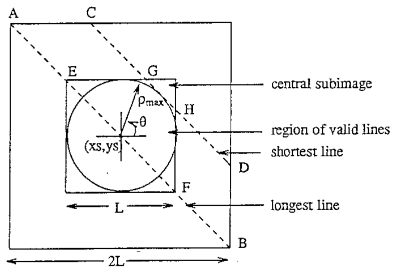

Fig. 4.5. An overlapped subimage used for the lowest level line segment determination.

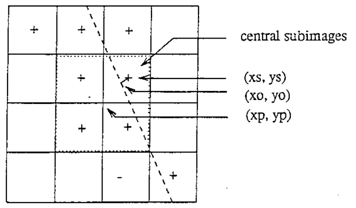

Fig. 4.6. A neighborhood of subimages containing collinear line segments.

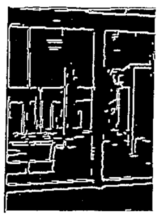

Fig. 5.1a. The binary line-thinned image.

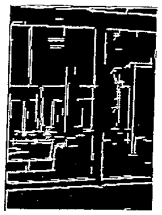

Fig. 5.1b. The level 1 lines.

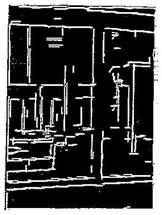

Fig. 5.1c. The level 2 lines.

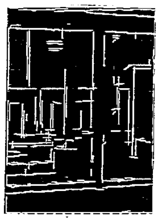

Fig. 5.1d. The level 3 lines.

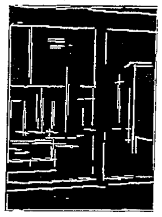

Fig. 5.1e. The level 4 lines.

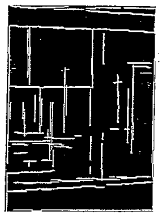

Fig. 5.1f. The level 5 lines.

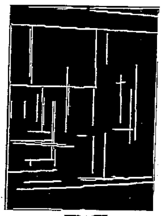

Fig. 5.1g. The level 6 lines.

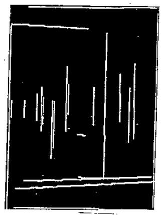

Fig. 5.1h. The level 7 lines.

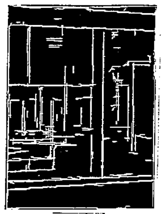

Fig. 5.1i. The complete set of lines from all the level.

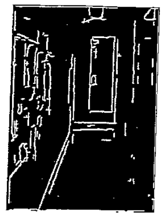

Fig. 6.1a. Line-thinned Image1.

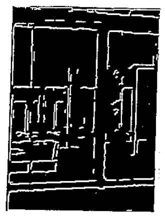

Fig. 6.1b. Line-thinned Image2.

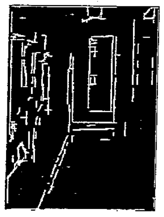

Fig. 6.2a. Level 1 lines of Image1 (init. subimage size = 4).

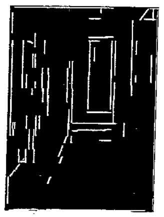

Fig. 6.2b. Level 1 lines of Image1 (init. subimage size = 8).

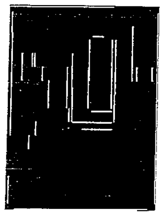

Fig. 6.2c. Level 1 lines of Image1 (init. subimage size = 16).

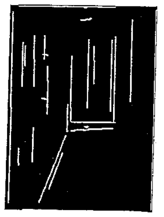

Fig. 6.3a. Level 6 lines of Image1 (init. subimage size = 4).

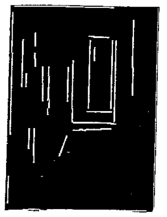

Fig. 6.3b. Level 4 lines of Image1 (init. subimage size = 8).

Fig. 7.1a. Input image for Table IV.

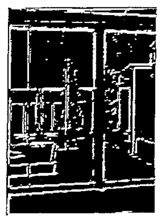

Fig. 7.1b. A binarized smoothed image.

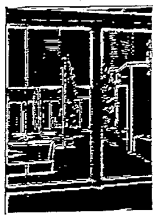

Fig. 7.1c. A binarized unsmoothed image.