TOKEN-BASED EXTRACTION OF STRAIGHT LINES

Michael Boldt and Richard Weiss

COINS Technical Report 87-104

October 1987

This work has been supported by the following grants: AFOSR-86-0021, DARPA N00014-82-K-0464, NSF DCR-8318776, and NSF DCR-8500332.

TOKEN-BASED EXTRACTION OF STRAIGHT LINES

Michael Boldt Richard Weiss

Dept of Computer and Information Science
University of Massachusetts, Amherst, MA 01003

*This work was funded by grants from DARPA (N00014-82-K-0464) and NSF (DCR-8318776).

†Address correspondence to: Richard Weiss, Dept of Computer and Information Science, University of Massachusetts, Amherst, MA 01003, or by computer, mail to weiss@umass.csnet

ABSTRACT

In this paper we outline an approach to intermediate-level processing which uses symbolic tokens and relational measures between them as the basis for grouping. The first part of the paper develops a computational framework for bottom-up image abstraction using zero-, one-, and two-dimensional tokens, i.e. points, lines, and areas; starting with the generation of tokens from the image, and followed by the creation of a hierarchy of levels of abstraction. Each abstraction step from one level to the next either groups several tokens into a single one (e.g. replacing a set of parallel lines by an area-token), reduces tokens (e.g. replacing a long, thin area by a line-token), or eliminates less significant tokens.

While grouping depends on the density of tokens and their features like contrast and color, often the most important property is their geometric configuration. Ways to reduce the computational complexity of the grouping processes are studied, and in so doing the advantages of a symbolic approach over other low-level vision methods are developed.

The second portion of this paper describes an implementation of collinear grouping for straight line segments, which is developed within the general abstraction framework of this paper. The initial line segments are generated from gradients at the position of Laplacian zero-crossing contours. The next step is repeated in a hierarchical fashion: a graph structure of linked lines is formed based on relational measurements between pairs of lines, this graph is searched to find sequences of lines which can be fit by a straight line, and the sequence of lines is replaced by a single line if the error in the fit is small. The relational measurements are endpoint proximity, orientation difference, lateral distance, overlap, and contrast difference. The results of the algorithm are shown for a variety of natural scene images.

1. Introduction.

The subject of this study is the construction of a symbolic, two-dimensional description of arbitrary images. The framework of grouping developed here fits within many different larger frameworks for image interpretation. For example, the VISIONS system at the University of Massachusetts is a general system for knowledge-based interpretation of natural scenes, such as house, road, and urban scenes [10,12,23]. The general strategy is to build an intermediate symbolic representation of the image data using grouping processes which do not make use of domain-specific knowledge. A partial interpretation is constructed from the intermediate-level data by associating an object label with selected groups of the intermediate "primitives" or "tokens". The object labels are used to activate portions of the system's knowledge network related to the hypothesized object. Once activated, the procedural components of the knowledge network direct further grouping, splitting and labelling processes at the intermediate level to construct aggregated and refined intermediate structures which are in closer agreement with the stored symbolic object descriptions. Within the VISIONS system, this forms the foundation for a three-level representation with the low-level for pixel-based images, an intermediate-level for tokens, and

a high-level representation for object hypotheses. The construction and grouping of tokens which is decribed here is part of the intermediate level of symbolic processing.

The grouping process also corresponds roughly to the full primal sketch proposed by Marr [19], but differs from it in two major aspects: it is not viewed as part of a model for the human visual system, and its descriptors are expected to be elaborate enough to be used *directly* by matching and other high-level interpretation processes. Marr, on the other hand, assumed that the primal sketch would be used only to construct the 2 1/2-D sketch and that the main purpose of a visual system was the recovery of the three-dimensional structure of a viewer's environment. He proposed four major representational stages within his model for the human visual system:

1. image;
2. primal sketch (organization of the image; two-dimensional);
   (a) raw primal sketch (token formation process);
   (b) full primal sketch (grouping processes);
3. 2 1/2-D sketch (description of visible surfaces; three-dimensional, viewer-centered);
4. 3-D model representation (description of shapes, organization of shapes; three-dimensional, object-centered).

The function of the primal sketch was to provide cues for the next representational level; i.e. how a surface behaves in three dimensions, such as the recovery of likely surface orientation and depth discontinuities. Vision was basically seen as the inverse of the image formation process. However, our philosophy is fundamentally different. The view taken in our paper and also expressed by Lowe [16] is that a complete three-dimensional reconstruction of the objects is not necessary for many of the tasks in computer vision, and in some cases is impossible. For example, since object recognition often can be achieved even from a line drawing or other symbolic representation, it does not seem to require a 2 1/2-D sketch or a three-dimensional model. On the other hand, having a detailed analysis of the two-dimensional structure of a single image or a sequence of images would be useful for any of the subsequent stages of processing. Even for model-based recognition, a representation of the two dimensional features of a three-dimensional model should lead to faster matching.

Two major tasks of computer vision are navigation and object recognition. To navigate, one needs to know *where* objects are in space, not necessarily *what* they are. For recognition, the case may be reversed; the viewer usually does not need to know how far away each part of an object is to recognize it. For example, human observers are able to easily recognize objects from a line drawing, and are very good at recognizing two-dimensional shapes like letters. Moreover, the human visual system has no difficulty in dealing with both tasks simultaneously. We realize that a movie screen is flat, and at the same time

are able to easily recognize objects on the screen, objects which we know to be three-dimensional. Whether we consider navigation based on stereo and motion processing or recognition based on two-dimensional features, it seems appropriate to have a much more powerful low-level system at the two-dimensional level than the one which Marr outlined. First, it should create symbols that are easy to match with those describing stored models. Second, it should also be able to support the quantitative processes of three-dimensional shape description, like shape from shading, contours, and texture in defining boundary conditions, or simply delivering contours and texture descriptors. An example of a system which recognizes an object from symbolic line data without reconstructing surfaces is SCERPO, implemented by David Lowe [16]. He uses grouping processes to find lines which are collinear or parallel and uses those lines in a model matching process.

Section 2 of this paper describes the requirements of a low-level system, and Section 3 elaborates on how a full symbolic intermediate representation (full primal sketch), which meets these requirements, might be computed. We present an argument for the early use of symbolic descriptors in low-level vision, and formulate an appropriate computational framework. We propose the creation of a hierarchy of geometric symbols, or tokens, by essentially replacing groups of tokens at level L in the hierarchy by a single token at level L + 1, where level in the hierarchy corresponds to an iteration of the grouping process. In the replacement step, a *perceptual radius* (or neighborhood) around each token is used to search for groups of tokens that belong together according to Gestalt laws. These laws rely heavily on geometric relationships among tokens and are based on the computation of relational measures such as orientation difference or percent overlap for any type of linear structure.

Repeated grouping leads to the description of image objects at multiple scales; however, the approach taken here is different from the current notion of scale space [27], where scale refers to the width parameter of the Gaussian function used to smooth an image and the results of processing are plotted with resect to this parameter. In our approach, scale refers to the range of sizes of tokens which are being processed.

Section 4 gives a demonstration of the practicality of the approach; the construction of straight lines from image data is described, showing that a straightforward algorithm which incorporates some nonlocal mechanism in a hierarchical fashion produces results which compare favorably with those produced by the best of existing algorithms.

2. Computational issues for low-level vision.

Consider the Gestalt psychologists' view: we see more often the whole rather than its parts. Unfortunately, introspection can only access the outcome of the abstraction processes, not their steps. In order to answer the question of how we get from local information to large scale geometric structures, we need to look at the type of processing that is required.

2.1 Why symbolic processing?

Low-level computer vision has often been viewed as image, or iconic processing (i.e. the generation of new images from old ones by convolutions, thresholding, or pixel labeling and so on), followed by some simple feature detection process. While iconic processing is thought to simplify subsequent computation, it does not reduce the huge representational gap between pixels at the low-level and semantic contexts at the high level. It seems natural to gradually change the representation as the scope increases from simple image events to complex groupings. This is fundamentally different from the approach which computes a succession of *intrinsic images*, e.g. surface orientation, reflectance, range, etc., as an intermediate representation [3].

Initial consideration of the following minimal requirements for a general low-level vision system will facilitate a more detailed discussion of the merits of the use of symbolic processing:

1. data reduction,
2. efficient performance,
3. integration of distributed image events,
4. multiscale geometric description of image events.

2.2 Data reduction

In low-level vision, we deal with enormous amounts of data. Obviously, the information has to be organized in such a way that the essence, or an overview, of the image structure is available as well as a quick access to finer levels of detail. There are essentially two ways of data reduction: filtering data and abstraction of data.

Let us consider two different types of filtering. The first is filtering via subsampling of the original intensity image. Image processing pyramids have been used, e.g. by Burt [7] to decompose an image into a set of bandpass images. An image at the level of a low-frequency band needs fewer sample points, and in this sense is a version with reduced data. While the result can be used in frame-to-frame matching, e.g. motion and stereo, it is not directly useful for description. There is also a loss of useful information, and, for example, one

cannot rely on only low-frequency data for detecting boundaries. Variations in lighting can create low-frequency changes which are not significant, while important fine lines show up only in the high-frequency spectrum. In addition, low-resolution images normally cannot help to find geometric relationships, since low-pass filtering destroys the necessary spatial accuracy which is especially needed to group small objects. Smoothing the image does not produce anything which describes the structure of the pattern. Simply removing the high-frequency information does not solve the problem of reducing the amount of data while maintaining the useful structures.

A second method of filtering is performed on a symbolic level to eliminate tokens which do not satisfy certain constraints. The criteria often used are size (for regions) and length, contrast, and orientation (for lines). This type of filtering can be used in conjunction with other processes in either a top-down or bottom-up approach. However, by itself, filtering via constraints on token attributes is not reliable for finding significant events for a wide range of domains [10]. Filtering only removes tokens, and it is often necessary to create new ones which are more complex and more likely to be significant.

The process of abstraction is another approach for reducing the amount of data. Unlike filtering it involves *replacing* a group of objects by a single object, or a complex object by a simpler one. For example, consider a checkerboard. What we would like as a description of the checkerboard is something like this: "the smallest meaningful units are squares; those squares lie on straight lines; the lines form a regular pattern covering a certain area." Note the increasing scale in size and abstraction for the descriptions, which are all valid simultaneously. They are symbolic, based on geometric relationships. We want to create a hierarchy of symbolic structures, where each level in the hierarchy has less data than the next lower level. The process of data reduction can be viewed as creating more abstract descriptions, rather than the process of reducing the spatial resolution of an image.

Searching for the appropriate abstractions which provide this capability for data reduction is still a difficult problem. Our approach has been to try to capture structural information, which is only part of the total information present in an image. This also differs from filtering techniques such as the laplacian pyramid from which it is possible to reconstruct the entire image. Leeuwenberg has formulated a measurement of conciseness for representations of structural information; this seems to be related to the ease with which humans perceive groups of lines as an object [15].

2.3 Efficient performance

When one considers the huge amounts of sensory data which must be processed and the large number of significant image events which need to be detected for effective navigation or object recognition, it becomes obvious that the low- and intermediate-level processes need to be computationally tractable. Since the computing resources are limited, this requires that in addition to reducing the amount of data so that there are fewer items to process, the abstraction processes themselves should be made efficient. The process of

abstraction involves searching for events which can be described more concisely as a unit. However, efficiency should not take place at the expense of generality.

Generalized Hough transform techniques are one way of performing rapid search and are capable of finding certain parameterized structures (e.g. rectangles or circles) [1,2]. However, there are several reasons why the Hough transform is not a solution to the general problem of finding efficient methods of processing for describing arbitrary images. first, the number of different structures is essentially infinite; although we restrict our interest to certain classes, like smoothly curved, compact, or repetitive structures, there are huge numbers of such patterns and each variation would need to be recognized separately. Second, Hough transforms ignore context (here we mean spatial or geometric context in which the primitive is embedded rather than semantic context) like the distance between tokens, or the location of other tokens that lie between the tokens to be grouped. For example, two line segments which lie on the same line are grouped regardless of the distance between their endpoints. To include all such parameters in a generalized Hough space would increase the dimensionality beyond a feasible level.

This leads us to the question, "What is the necessary scope in the image needed to detect 'the whole'?" Surely it is not necessary to examine the entire image. On the other hand, normally we can not just look at pairs of tokens (e.g. a token and a nearby neighbor) to see if they form something meaningful. Depending on the type of grouping, there is a "typical nonlocality" or context involved, leading to a minimal number of tokens, to form a new, more compact geometric structure.

What does it cost to find a new geometric structure involving a symbolic object A? The set of objects around A which are examined in this search is called a ***neighborhood.*** Let us assume we want to detect the best combinations of A with its neighbors which satisfy specific criteria. The amount of computation for blind search would increase very rapidly with the number of objects : if we searched in areas with n neighbors around every object, and looked for combinations of m objects, then the amount of computation would be of the order $C(n, m)$, which grows as n^m. How can the search space be reduced? One option is to reduce m, the number of objects which compose a new structure. This can be done by building structures hierarchically. Another option is to reduce n, the number of objects examined. This can be done by locally filtering the set of objects to be searched based on object features or searching only in appropriate directions or areas. The effect of these approaches is the reduction in the number of combinations in the neighborhood. In addition processing each neighborhood in parallel will reduce the time considerably.

2.4 Integration of distributed image events

In general, one of the most important issues in intermediate-level processing is the consistent integration of unreliable or ambiguous information across different and possibly large parts of an image. The image events which provide evidence for objects are often not present in contiguous pixels alone. While local processing in a small neighborhood is

necessary as the starting point, it is not sufficient. Figure 1a-d shows some examples of how distributed image events need to be integrated. In this figure the events are perceived as straight lines at some scales, but not at others. Figures 1a and 1b show lines which are fragmented, but which seem continuous when viewed from a distance. Figure 1c shows lines whose directions vary, but when viewed from a distance they are seen as straight. Figure 1d shows a sequence of dots are perceived as a straight line. In fact the lines on this page are composed of dots. In each of these cases the evidence for a straight line is not present if processing is restricted to a small window; there are gaps and the orientations are not equal. If we integrate the information present in a large number of these windows which are properly aligned with what we perceive as the line, then the evidence is very strong. The use of geometric context is this process of integrating distributed information.

2.5 Multiscale geometric description of image events

There are at least three possible meanings for the term *scale*. Here it refers to a range of sizes with respect to the entire image. This is useful for describing image events or processes. In our model of processing, computation for a particular token or image event is restricted to a neighborhood whose scale can depend on the goals of the process and the tokens already found. Processing is done at multiple scales because the same algorithm, when applied to objects of different sizes will produce different results, which then need to be integrated to obtain the most useful information. The integration of distributed information in the image is accomplished in our system by performing processing sequentially at multiple scales, restricting the processing at each scale to the output of processing at lower scales, although the processing at any scale can be spatially parallel. The reason for this ordering of scales is that our approach is both hierarchical and constructive. Fischler and Bolles [11] perform an analysis using multiple scales on curves which have already been produced, so each scale can be processed independently. Reynolds and Beveridge [22] process all scales simultaneously by defining the neighborhood of a line to be dependent on the length of the line and refer to this relational measure as spatial proximity. In any case, the results from different scales need to be combined. For example, the final description of our checkerboard example earlier encompasses information at many scales. Note that there is also a concept of abstraction scale, and the description of the checkerboard has multiple abstraction scales.

Lowe and Binford [17,18] have also tried to deal with the problem of how to determine which scales contain the information needed to solve a specific problem. For example, if one has a description of some objects which might be in a scene, and one does not know *a priori* how large the objects are in the image, how can one determine at which scale to look for evidence for those objects which are present? In their paper, they analyze the problem of segmenting a curve into straight and circular line segments. The solution they propose is a measure of meaningfulness based on probability. Events are considered meaningful if they have a low probability of occuring when the scene does not contain the desired

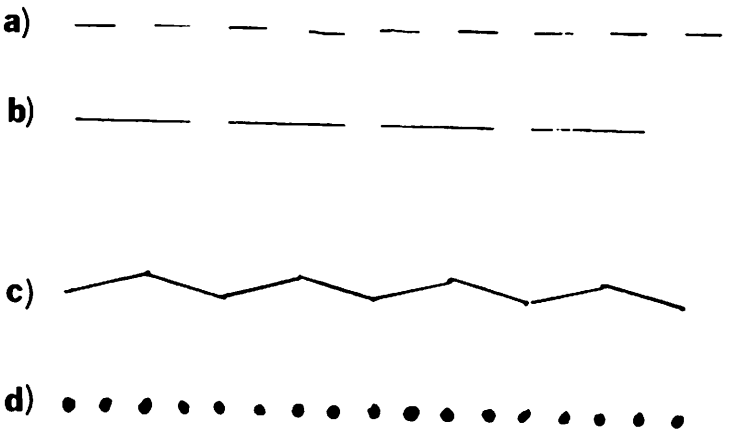


Figure 1: Perceptual Integration of Distributed Image Events. Each part is a group of events which can be perceived as a straight line at a sufficiently large scale.

objects and a high probability when they do. In the case of points lying on a curve, Lowe and Binford compute the first probability based on the assumption that edge points which do not lie on the same straight or curved line will be distributed uniformly in the image. This assumption will be violated in some cases, and it might be possible to find a weaker assumption which would still permit a mathematical formulation. In their paper, they do not explicitly compute the second probability that edge points which lie on the same line will be within a fixed distance from an approximating straight line. This probability will be large provided one does not impose too strict a test for straightness. A scale is determined to have significant information if the events when measured at that scale have a local maximum in the meaningfulness measure when compared with other scales. The curve segmentation problem is attacked by examining the curve at a point simultaneously at all scales and selecting those at which it is most likely a straight or circular line, provided that the meaningfulness is at least above a minimum value. This seems to work well for curve segmentation, but was not demonstrated for curve construction, where the complexity of the search is much greater. While a weakness in this approach is the assumption of a uniform distribution of edge directions for background areas, this type of analysis holds the most promise for deriving the parameters which govern the grouping in our system and which we have chosen empirically.

3. A computational framework for symbolic processing

The need for a multiscale description suggests a hierarchy, which when combined with a symbolic representation, facilitates the satisfactory meeting of the other processing requirements. For hierarchical, symbolic processing, neighborhoods at each level have a small number of image events, thus the search space is kept small. At the same time the neighborhoods at higher levels cover a large part of the image, so integration of distributed events is also achieved. Having explained why symbolic processing or abstraction would be useful for a low-level vision system, we now describe conceptually such a system. It consists of extracted image events called *tokens*, and operations on tokens called ***abstractions*** which produce new tokens.

3.1 Image tokens

A token is a description of a set of image events or other tokens. The goal is to create tokens that correspond to those events which the human observer perceives as a unit. There are two ways to produce tokens. They can be generated directly from an image, e.g. by edge detection, or they can be generated from other tokens, e.g. straight lines can be grouped into longer straight lines. Since tokens which have a large size may require several steps to construct, it will be necessary to have tokens of intermediate size as precursors.

Tokens fall into three major classes based on their dimensionality :

1. points (dimension 0): spots, corners, line endpoints, etc.;

2. lines (dimension 1): straight line segments, circular segments, quadratic curve segments, linked line segments, etc.;

3. areas (dimension 2): closed curves, parallel lines, rectangles, general polygons, texture patterns, etc.

One should consider the generality of the vocabulary of tokens. The complexity of a system can be reduced by including only those tokens which are specific to one domain. On the other hand, such a system might not easily describe events outside that domain. In general, the larger the vocabulary of tokens, the more concise the description can be, but this economy of description is achieved at the expense of increased computational complexity in the search for best description.

3.2 Token abstraction

There are two types of operations which we would need in order to carry out our programme of symbolic processing. The first type involves the grouping of tokens to form aggregates which are then represented by a single token; this is called "grouping". The second type occurs when we want to ignore some of the information present in a token; this is called "reduction". This not only reduces the amount of information which needs to be stored for a token, but also allows a simpler grouping process to be applied. For example, the latter type is useful in the case where we have a number of regions which are part of a larger geometric structure. The individual regions have a variety of attributes, such as average intensity, area, texture, and measurements associated with the boundary. In the case of geometric grouping, especially when the size of the whole structure is large compared with the size of the individual component regions, these attributes would not be used in the grouping process. Instead, we might consider each region as being represented by just its location (and perhaps its orientation). For example, an area which is not elongated can be reduced to a point, and several areas can be grouped by a point-grouping process. An area which does have a major axis can be reduced to a line which approximates that axis; e.g. smoothed local symmetry [4] or medial axis transform [5]. A line which has a short length can be reduced to its midpoint. In all of these cases, the criterion for the reduction process will depend on the relationship between the size of the token and the scale at which processing occurs. The scale will determine the size below which a token will be a candidate for reduction.

A general system which allows all possible reductions may require a sophisticated control structure to decide which reduction to apply and when. If one were simply to allow all reductions and groupings to be instantiated, the number of tokens would increase rapidly, resulting in a large increase in the processing time. While the application of some grouping

Token Abstractions		
gradients	group to	an area (spot) ('whole')
an area	reduces to	a point
points	group to	a curve segment
curve segments	group to	a circle (area) ('whole')
a circle	reduces to	a point
points	group to	a line ('whole').

Table 1: Example of Grouping and Reduction: abstractions for Figure 2.

processes can be applied in almost all domains, e.g. straight line grouping, it seems likely that top-down strategies would be useful as a way of limiting the types of grouping.

The grouping algorithm itself does not require that we know whether the tokens came from region segmentation, line extraction, or feature point detection. The reduction process makes this explicit by transforming the region tokens into point tokens, so that the same algorithm which was designed for points can be applied. Within the framework of hierarchical processing, the reduction transformations are also related to the concept of scale. If the scale at which processing is occurring is large with respect to the diameter of the regions involved, then their behavior should be point-like. As a more complex example of hierarchical grouping and reduction, consider Figure 2. Let us assume that the most primitive detector finds gradients in the image, and the token vocabulary consists of circular areas (spots), points, line segments (circular or straight), and closed curves (circles). Some tokens represent a complete unit or 'whole' in that they not part of a larger aggregate without first undergoing reduction. The abstraction process for this example is described in Table 1.

Now consider the general case of which transitions can occur for each of the three classes of tokens: points, lines, and areas. This is divided into two cases: grouping processes and reduction processes, which are shown in Table 2.

In this paper, we primarily discuss grouping tokens of the same type; however, work on grouping of tokens of different types and the grouping of tokens of the same type to produce a token of a different type has also been done. Riseman *et. al.* [24] have examined regions and lines together in order to combine the information available from different types of segmentation algorithms. This can be viewed in terms of the above formalism as creating an area token which includes in its description the attributes of the regions and lines which are grouped together and several relational measures based on their intersections. Reynolds and Beveridge [22] have grouped aggregates of parallel, collinear, and orthogonal lines into area tokens and have produced very useful results. They also discuss the problem of deciding which aggregates are most significant.

The abstraction process is central to our approach to symbolic processing and consists of multiple cycles of grouping and reduction, during which the scope and token size increase

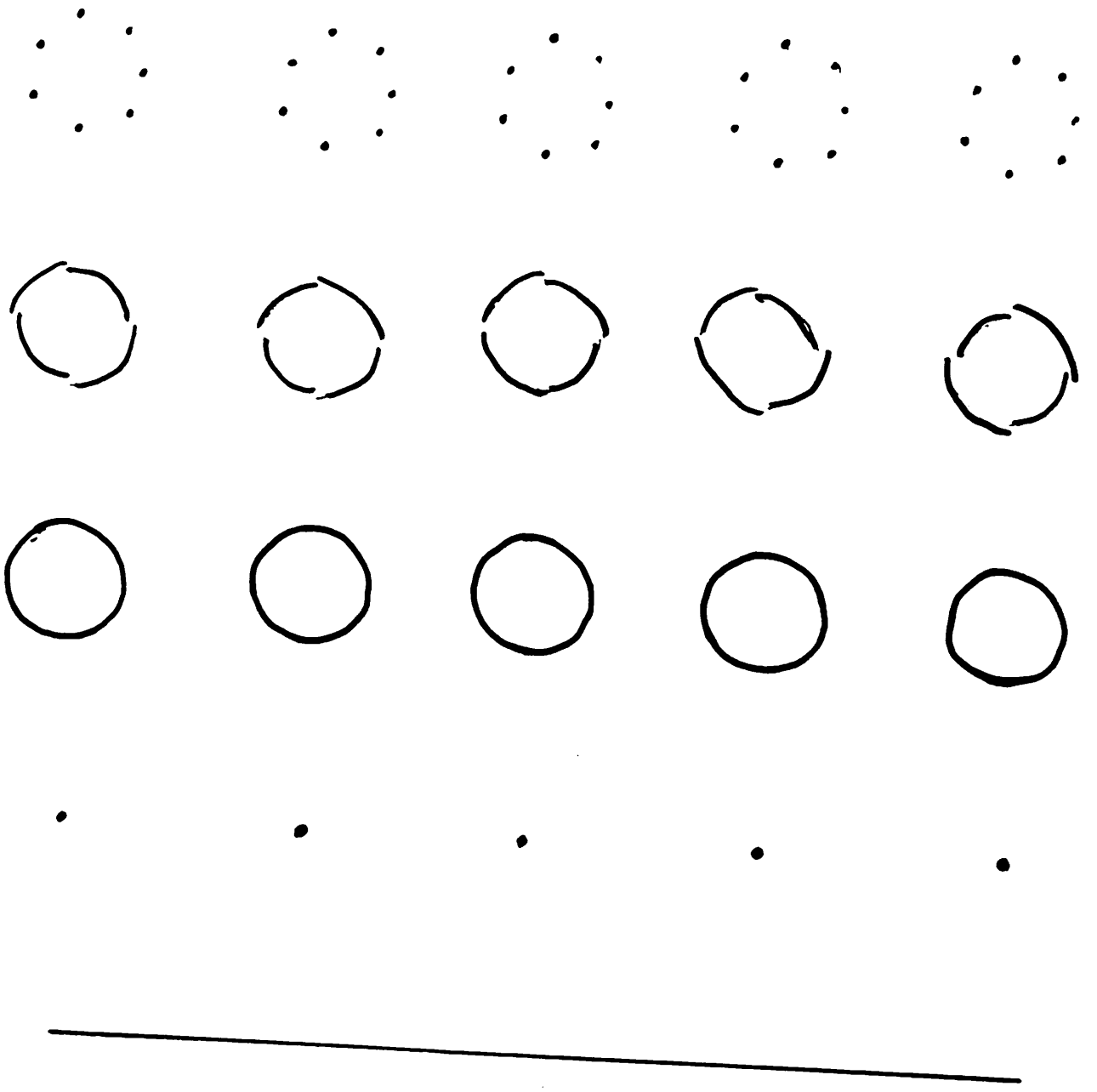


Figure 2: This shows an abstraction process: points are grouped into curved lines, curved lines are grouped into closed curves which define areas, areas are reduced to points, points are grouped into a straight line.

Grouping (takes multiple tokens to one token)		
dimension of input tokens	dimension of output tokens	description
0,1	1	points or lines group to a line
0	2	points group to an area
1	2	lines group to an area, e.g. texture, rectangles etc.
2	2	areas group to bigger areas

reduction(takes a single token to another one)		
dimension of input tokens	dimension of output tokens	description
1	0	small line is replaced by a point
1	1	rough line is smoothed
2	0	small region is replaced by a point
2	1	elongated region is replaced by a line

Table 2: Summary of token abstraction: grouping and reduction.

from fine to coarse scale. As discussed above, the reduction process transforms a complex token into a simpler one. In the grouping process, an aggregate of tokens is found which is transformed into a single token. This process is divided into two steps: linking and replacement.

3.3 Linking tokens

Linking is the first step in the grouping process and is the means by which we can reduce the size of the search space by filtering the number of tokens to be examined in the subsequent processes. For each token, all other tokens of a specific type in its neighborhood are examined, and links are formed only to those tokens whose attributes satisfy prescribed geometric and non-geometric criteria. These criteria are formulated in terms of *relational measures*, which are general functions of the attributes of a set of tokens. Since tokens will not necessarily link symmetrially, the links have a direction. The resulting graph, which has nodes which are tokens and arcs which are their links, is a directed graph called the "link graph". As illustrated by the example of the checkerboard, we would like to extract the structural information in the image, so we want the link graph to preserve this information. In the case of straight lines, this geometric information is captured by the geometric relational measures of proximity, collinear alignment, orientation, and overlap.

In general, for most tokens there will be a proximity measure and a relative orientation or alignment measure. The measurement of proximity and alignment is a function of the scale of the processing. For the linking process, the radius of the neighborhood is called

the *linking radius*; two tokens can be linked if the minimum distance between two tokens is less than this radius. In the case of lines it is from the endpoint of one line to the closest point on the line to which it is linked. The alignment measure for lines is the angle between them as well as the overlap of their lengths. For aggregates of regions with lines which bound them, alignment can be measured by the overlap between the boundary of the region and each of the lines. Relational measures for line grouping have also been explored by Reynolds and Beveridge [22]; for a general discussion of relational measures, see Riseman et. al. [23].

3.4 Replacing tokens

The second step in the grouping process, which is the replacement step, consists of searching the link graph for geometric structures, which may be implicitly or explicitly represented by models, verifying the match and replacing those structures by a token which describes them. In the case of straight lines or circles the model is given implicitly by the equation of the curve which best fits the data and the verification is done with a least squares measurement of endpoints on the line segments. For objects such as rectangles, the model may consist of pairs of parallel lines with an orthogonality relation between them. The verification might be accomplished by finding the smallest rectangle containing the line segements and comparing the boundary with the original segments. In either case, the relational measures which are used for the linking step (usually for pairs of tokens) are not necessarily used in the verification step where higher order relations are tested.

The size of the subgraph which should be examined during the search depends on the amount of context which must be examined in order to reliably determine if the new, abstracted token is present. At each step it is necessary to limit the computational complexity, so we restrict the search around each node to subgraphs contained in a neighborhood, whose size is given by the *perceptual radius*. The perceptual radius is possibly larger than the linking radius and allows the search process to look at more than two tokens at a time.

An additional aspect of the replacement process is the description of complex tokens which represent aggregates of simpler tokens. This must be done in such a way as to facilitate grouping of the new tokens. In the case of straight lines this is straightforward, but in many other cases such as curved lines a concise, symbolic representation must be developed.

3.5 Control of grouping

Each subgraph found in the grouping step is a hypothesis of a geometric structure to be verified before being replaced by a single new object and stored. A control issue arises since it is possible for a token to be part of multiple aggregations which are candidates for replacement. This could lead to multiple representations of the same image event. In some cases it may be desirable for the replacement of a token to block the replacement of

a different aggregation containing the same token. It is easy to see that the linking and replacement steps must be done sequentially, but within each step there is potential for parallel computation across the image. For the linking step this is not difficult to do, and if mutiple replacements of a token is allowed, this is also possible for the replacement step. If multiple replacements are disallowed, a decision must be made for any given aggregation by looking at all other aggregations which have a token in common with it. In general, this will not involve looking at a large number of aggregations and tokens; however, there is no guaranteed upper bound.

As a general approach to solving the control problems, we consider a hierarchical organization of tokens. This allows us to represent the description of a checkerboard at multiple scales and provides a framework for describing the processing of such an image which is done separately for each scale. The hierarchy must be capable of representing the information associated with reduction processes as well. The hierarchy which results has two dimensions; scale and abstraction. The simplest control structure for the processing of this token hierarchy is to start with tokens at the smallest scale (usually pixel size) and increase the scale monotonically with the number of cycles. A basic assumption is that the density of tokens at each scale is restricted and decreases as the scale increases. If there is a high density of tokens at one scale, and if they are not grouped, some tokens will be dropped. If the density is low, e.g. a few points against a homogeneous background, then the density constraint does not force their removal, so they can persist to higher scales. The abstraction level of a token is more difficult to describe, but it should increase with the number of reductions which have been applied to produce it.

4. Applying Token-Based Abstraction to Straight Line Grouping

In this section we apply the principles of low-level symbolic processing that we have developed to the problem of grouping lines into longer, straight lines. This is demonstrated on the images of natural scenes (see Figure 3) for which standard boundary detection techniques encounter significant problems due to noise or texture, or where edge events have low contrast [25,26]. Our goals are to show that geometric grouping via token aggregation and replacement is effective in finding the significant straight lines in an image when combined with standard local edge detection techniques, as well as to develop an effective straight-line grouping algorithm for image interpretation.

The grouping algorithm is based on relational measures which are used to derive binary relations between tokens. These binary relations can be summarized as collinearity, proximity, and similarity in contrast; they are used for filtering in the search process for replacing aggregations by new tokens. The relational measures are described in more detail in Section 4.2. A straightness test, which is a verification procedure, is applied line sequences of arbitrary length; it is not restricted to pairs of tokens. Thus, we take the following algorithmic definition of a straight line: it is composed of a sequence of line segments in which consecutive pairs satisfy the relations of collinearity, proximity, and similarity of contrast, such that the entire sequence passes a straightness test. All of these criteria depend on scale; long lines, for example, can be separated by a larger gap than small ones and still be close.

The two major components of the algorithm are edge detection and hierarchical grouping. Hierarchical grouping has two steps which are performed at each level: linking and replacement.

4.1 Edge Detection

The input to the grouping algorithm is a set of line tokens together with a measurement of the average intensity contrast from one side of the line to the other. Usually this is produced by an edge detection algorithm applied to the image. The properties which an edge detection process should satisfy are: 1) high positional accuracy of an edge, even with aliasing, 2) good sensitivity to high frequency data, and 3) reduction of the data (many pixels don't produce edge points). The last of these properties, reduction of data, can make a significant difference in the computation time. However, it is also important that the algorithm be sensitive to low-contrast edges. There are many possible edge detection algorithms which could be used. The two algorithms which we have used for selecting initial edge locations are zero crossings of the Laplacian operator and a directional edge operator based on the work of Haralick and Canny [13,8]. Although there may be algorithms which have better performance in some cases, these two are representative of a class of algorithms; most of them are expected to have the same types of problems which we encounter here [9]. The Laplacian operator was chosen because it seemed to be the better of the two.

Since most of the results have been obtained for the Laplacian operator, we describe the processing for that operator in more detail. First, the image is convolved with a 3×3 mask which approximates the Laplacian:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Next, zero crossings of the Laplacian image are detected and their positions determined by linear interpolation. A zero crossing is a zero value of the Laplacian output and consequently is the location of a change in sign; it typically occurs within a pixel and requires intrapixel interpolation. Figure 4a shows that the zero crossing contours do not consistently follow the boundaries of objects or other identifiable parts of the image (e.g. see the house roof). The reasons why the contours are not necessarily a reliable indicator of the direction of an edge can be understood by looking at some of the assumptions about edges and the Laplacian operator. In Marr's paper [19] on the use of the Laplacian, he makes the assumption that the intensity surface is planar in a neighborhood of the edge. Even for an ideal curved step edge this assumption will be violated, and the results in real images with aliasing from the digitization process have serious problems. On the other hand, the gradient tends to remain perpendicular to the edge. For this reason, we use the gradient direction at each edge point for the edge orientation; in general, this will differ from the direction of the zero-crossing contour. Figure 4b shows the gradient information for the edges in the house roof scene.

The location of a zero-crossing point is computed by interpolation of the values at four pixels which form the corners of a square. The gradient at that point is also computed by interpolation. An edge is positioned with its center at the zero-crossing point, and the orientation of the edge is perpendicular to the gradient. Each edge carries with it a contrast (the gradient magnitude) and direction. In addition, each edge (now considered as a line) has an *initial start point*, a *final end point*, and is defined to have a length of one pixel.

The grouping algorithm does not require that the input edges be completely reliable, and it has produced very good results despite the variety of problems encountered with the Laplacian operator. These problems include saddle points of the Laplacian and anti-edges at local minima. When a zero-crossing point is a saddle point of the Laplacian surface, then locally the contour will consist of a pair of hyperbolas or a pair of lines which cross. If these saddle points have low density, then they can be ignored. If they have high density, then a different type of grouping needs to be applied because the orientation at a saddle point is ambiguous (we do not consider that here.) Zero crossings of the Laplacian also occur where the gradient magnitude has a local minimum. The current implementation does not test for these *anti-edges*, and therefore they remain present in the initial edge data.

The process of acquiring an image itself can introduce artifacts which complicate the

line grouping process. Subsampling of an image, quantization effects, and the presence of noise can result in isolated pixels which have a high contrast with respect to their neighbors. Subsampling is especially noticeable when the pixel size of the sensor is large compared with the spatial period corresponding to the highest frequency of the intensity variations on the image plane. As a result of the local extrema produced by these phenomena, zero-crossing contours surround isolated pixels and produce a point-like structure. The line grouping algorithm presented here is not specifically designed to handle this situation, which would require the grouping of isolated points. In addition, noise can introduce numerous local maxima in the gradient of the intensity, producing multiple zero crossings and hence multiple edges parallel to the visually significant one. Many of these problems could have been solved by smoothing with different-width Gaussian masks, which has been explored by Witkin and others [27,28,19] using a scale-space approach. However, as mentioned earlier, smoothing removes details which are essential for some structures. For example, a very thin linear structure, even if it is long, will become undetectable if high-frequency data are filtered out. Our approach, on the other hand, is to use the geometric context to eliminate edges (due to subsampling, noise, anti-edges, etc) which do not form lines in the image.

4.2 The Hierarchical Grouping Process

In this application, the grouping process uses geometric and intrinsic properties at multiple scales to form long, straight lines from shorter segments. In general, a grouping cycle consists of two steps: linking and replacement. In the linking step pairs of line segments are connected based on binary relations. In the replacement step, sequences of linked line segments are tested for straightness and possibly replaced by a single line segment. The issue is how to avoid searching and testing all combinations of line segments within a large area. The first part of the solution is the construction of the link graph. If there are N lines in an area to be searched, then there are 2^N sets of lines which could combine to form a straight line. On the other hand, the length of a sequence of lines which would be linked is only about $L = \sqrt{N}$; and if the average degree of the link graph is G, then the complexity of the search is only G^L, where G is usually less than 2. The second way to reduce the computational complexity is to limit the number of tokens in the area being searched. This is done by processing the lines hierarchically: the repeated grouping of lines into longer lines, which implicitly defines a scale-space hierarchy, so that the search always remains local with respect to the current scale of processing.

The idea of grouping line segments is not new. Nevatia and Babu [7] have reported criteria for grouping edges; however, they only applied it locally and they did not take into account more global context. McKeown and Pane [5] have also devised rules which are similar to ours and have applied them to the problem of linking elongated regions. They use the criteria that regions a) must be close; b) must not overlap too much; c) must have similar orientations, and d) must have small lateral distance. Thus, they implement the

rules of collinearity and proximity. A major difference between their approach and ours, however, is that while they used the size of each region to determine its context, they do not use a hierarchy.

Linking

The linking step consists of a search for pairs of lines that satisfy the geometric and nongeometric relations which make them candidates for grouping. The criteria are based on relational measures, e.g. difference in orientation, difference in contrast, etc., and are used to filter the search space by producing the link graph. The relational measures provide a means of comparing pairs of tokens via a continuous variable. Thresholds make it possible to convert these measures into a binary relation, which are encoded as the link graph. In our work we chose these parameters empirically. One topic for future work will be an examination of how these parameters can be determined directly from the image or the models of objects to be recognized in a scene.

The geometric relations which we have used are collinearity and proximity, and the nongeometric relation is similarity of contrast. Consider a line $L_1 = (P_1, P_2)$. A Line $L_2 = (Q_1, Q_2)$ is linked to L_1 if it fulfills the following five relational constraints (the geometric relations are illustrated in Figure 5):

1. L_2 must intersect the circular area whose radius is the ***linking radius***, R_l, and whose center is P_2 on L_1 (See Figure 5a.). This radius should depend on the lengths of the lines and the density of line tokens. In our experiments it was set to be 0.6 pixels at the lowest scale and multiplied by a factor of 1.2 between scales. For computational efficiency, this test is a filter to select line candidates and is done in the process of retrieving lines from the data base. Thus, the other tests are only applied to a small fraction of the lines. A more restrictive test for proximity is performed in step 3.

2. L_2 and L_1 must be approximately collinear. This relation has two parts. a) The difference in the orientations of the two lines must be less than a threshold which was set to be 30 degrees based on several experiments (See Figure 5b.) In particular lines which which are collinear but have opposite contrast are 180 degrees apart and are not linked. This condition can be relaxed to allow such lines to be linked. b) The lateral distance between the two lines must be less than a threshold, which was set to be the linking radius. The lateral distance is measured as the perpendicular distance from the midpoint of the line L_2 to L_1 (See Figure 5c.)

3. Endpoints must be close. The projection of Q_1 onto L_1 must lie within a specified interval (which is proportional to the length of L_1). A typical threshold on this interval extends from the midpoint of L_1 to one-quarter beyond the endpoint P_2. (See Figure 5d.)

4. Lines must not overlap too much. Let P_1 and P_2 define an order on L_1, P_1 being *before* P_2. If Q_1 projects onto L_1 between P_1 and P_2, then the distance from the projection of Q_1 to P_2 must be less than the distance from P_2 to the projection of Q_2. (See Figure 5e.)

5. The contrast of L_2 must be similar to that of L_1. The ratio of the larger contrast to the smaller one must be less than a threshold, the default value of which is currently set at 2. Alternatively, the similarity of areas adjacent to the lines could also be used. (See Figure 5f.)

The same linking process is performed for all lines pairs retrieved by the first constraint above. It is done separately for each pair of endpoints and can obviously be done in parallel. The result is a global, directed graph in which the set of vertices is all the line segments in the image at a given scale and the links as its arcs. (See Figures 5g and 5h.) For the linking parameters which we have chosen at each scale, a line will be linked with between zero and three other lines at each of its endpoints. The replacement process examines paths in this graph and tests them for straightness. The linking process effectively reduces the number of combinations of lines which must be examined by the replacement process.

Replacement

The replacement process consists of searching and replacement. The amount of geometric context used is determined by the *perceptual radius*, which bounds the length of a sequence of lines which is tested for straightness. Each sequence of line tokens which includes a given line is approximated by a straight line, and if the approximation is sufficiently good, then the sequence is replaced by a new single straight line token. The algorithm proceeds by examining each line in the link graph and performing the following steps:

1. If the line has already been merged with others, skip it. (This is the easiest solution for a sequential implementation. For a parallel implementation this step has to be modified slightly.) For straight lines, the order of replacement is usually not significant, but for more complex tokens, it may be necessary to allow tokens to participate in multiple merges. In our research thus far, we have not yet investigated the question of choosing the best candidate for replacement. In order to simplify the computation, we choose the first candidate which passes the straightness test.

2. Generate all paths of lines from the initial point, P_1 and all paths from the final point P_2, within the *perceptual radius*, including the paths of length zero.

3. Generate all paths which are combinations of paths from step 2 as follows: a path to the initial point, the line itself, and a path from the final point. The longest paths

are generated first, so that a maximal path is tested for straightness before any of its subpaths.

4. Fit a straight line, using the least squares method, to the set of endpoints of the individual lines of a path from step 3, and calculate the following measure of straightness:

$$S = \frac{\sum_1^n d_i^2 \cdot l_i}{(n-2) \cdot s^2}$$

where n is the number of endpoints, d_i the distance from the i^{th} endpoint to the approximating line, l_i the length of the line segment corresponding to the i^{th} endpoint, and s the scale of the token, which is the distance from the initial point of the first line segment to the final point of the last. The factor of $n-2$ was used because the measure is not valid for a single line with two end points. If the straightness measure for a set of endpoints is less than a threshold whose default value is 0.05, the sequence of line segments is considered straight.

Although we limit our work to straight lines, we also want to introduce a method for approximating some smooth curves by straight lines; in particular, circular arcs with low curvature (we are currently working on a more general system for grouping and describing curves). We allow this to happen by applying a curvature measure to line sequences which fail the straightness test. The curvature of a set of points is defined to be the reciprocal of the radius of the circle which is the best fit to the set of endpoints. If the ratio of the scale to the radius of the best fit circle is less than a threshold, the sequence of line segments is considered straight. The default value of the threshold for the ratio was set to be 1.7, which was found empirically. Thus, piecewise straight lines are accepted as a reasonable approximation to a circular arc with low curvature if the sequence of lines fits it closely, even if the straightness measure is large.

5. Choose the first path which satisfies either of the two criteria in 4, and replace it by a straight line whose contrast is the average (weighted by length) of those of its parts. Other paths through the original line are also replaced, but we limit the number of these by requiring that the difference between the slopes of two such replacement lines be greater than a threshold, whose default value was set to be 40 degrees.

On each cycle, each edge or previously grouped line is considered for grouping and those which pass the straightness test are replaced. Since the grouping process is iterated many times, some lines will begin to organize into longer lines while some will remain unreplaced. Some of these latter lines are not part of a longer line and others are part of a longer line but failed to group. Sometimes this happens because each line token can participate in at most one replacement in each cycle, so there are line tokens which could be grouped with another line token but that token has already been replaced. In both

cases, unreplaced lines are copied to the next level of the hierarchy. However, there is a price for doing this; since it is important to eliminate lines and to reduce the size of the search space during later cycles, under our current grouping scheme an unreplaced line is copied to the next level up to four times (after that it will be deleted for future cycles of grouping.) As a result, lines at a single level will not all have the same length; their lengths will lie within a range, and the ranges for different levels will overlap.

The hierarchical representation

The hierarchy used is very simple; lines are classified by size in the image. There are two important reasons for using a hierarchy. First, the description of straightness is dependent on scale. As was shown in Figure 1, a sequence of lines which is not straight at one scale can be part of a longer sequence that passes the straightness test at a larger scale. Second, there are computational advantages. In our implementation, the hierarchy contains multiple levels, each representing a different scale. A level is a set of directed line segments together with their contrasts. The scale of a level is the range of the lengths of lines which can be stored in that level. In the current implementation these ranges overlap, and the linking and replacement process is repeated at each level.

The linking and perceptual radii are multiplied by constant factors from one level to the next, typically 1.2 and 2.0 repectively. The linking radius controls the retrieval of candidate lines from the data base. The perceptual radius controls the length of sequences in the link graph. In order to reduce the search computation during the linking step, each level of the hierarchy is divided up into a spatially uniform grid corresponding to the density of lines. Ideally, the density will decrease by a constant factor, since a certain fraction of aggregates of line tokens will be replaced by single line tokens. If each line token were merged with another single line token during each cycle, then the density of tokens would be halved; if at the same time the linking radius were multiplied by the square root of two with each cycle, the area would double and the number of tokens to be searched would remain constant. In practice many of the initial lines are never grouped, so a strategy has been adopted that they will be copied a fixed number of times and then dropped. With the current parameters, aggregates of more than two tokens are rarely replaced, so the most common grouping is a pair of lines. Thus, the density of tokens will decrease slowly at first and then rapidly after four cycles.

Although all of the levels in the hierarchical grouping process share the same coordinate system, there has been no need of explicit pointers between the individual lines and the pixels of the original image in the algorithm presented here.

5. Experimental Results for Hierarchical Line Grouping

The algorithm has been applied successfully to many natural scene and aerial images. Four of these are shown in Figure 3. We illustrate the algorithm on a subimage of the one

shown in Figure 3a. Figure 4a shows the contours of the Laplacian zero crossings. There are several places where the contour does not follow the boundary of the roof, but wanders off into the texture of the roof or the texture of the trees. Figure 4b shows the initial edge segments which are input to the grouping algorithm. Figures 6a – d show the stages of the geometric grouping algorithm after two, four, and six cycles, as well as the final output by integrating the results at all of the scales by only taking lines from each scale that were not grouped at a higher scale. After the fourth cycle, the outline of the chimney is present as are large pieces of the boundary of the roof. After the sixth cycle, many of the shorter lines have been dropped.

Restricting the algorithm to just the subimage containing the chimney, we illustrate linking and replacement in more detail. Figures 7a and 7b show the lines after one and two cycles, respectively, with links shown by circular arcs. In general, only pairs of lines are merged. There are also many cases where linked lines do not pass the straightness test, e.g. the trihedral vertex at the top. There are also cases where merged lines are able to link with lines which were not linked at a smaller scale.

Filtering by contrast or length is likely to retain the most perceptually significant lines, for example the boundaries of objects in the image. Figure 8 shows the resulting lines whose contrast is greater than 15 gray levels (out of 255); none of the lines in the sky region are left. As a basis for comparison, Figure 9 shows the filtered output of the Burns straight line algorithm [6]. That algorithm is very successful in locating many of the straight lines, but many of the long ones are fragmented. This is due to the fact that the Burns algorithm uses a connected components algorithm to group pixels with similar gradient orientation, and pixel data supporting an edge can be interrupted by aliasing, texture on either side, or even the smallest occlusion. The Burns algorithm is conceptually somewhere between an edge extraction algorithm and a line grouping algorithm, and applying the geometric grouping algorithm to the output of the Burns algorithm pieces together the lines which were fragmented. The geometric grouping algorithm performs best on the long lines because it integrates evidence from many parts of the image. One reason for not using the Burns algorithm to produce the input lines for the grouping algorithm is that the former sometimes distorts the orientation of lines somewhat. This is due to the use of all pixels within a connected component of pixels with similar gradient orientation. For example, if there is surface shading perpendicular to an edge, the gradient orientation could remain constant in the shaded and unshaded areas, and if the shading is stronger at one end, the line will be skewed toward the shaded area. This situation does not create a problem for the geometric grouping algorithm because parallel edges do not influence each other and the collinearity constraint would limit the orientation error.

Figure 10 shows the grouping algorithm applied to the entire image of the house scene with lines of length greater than one pixel and no filtering on contrast. Removing lines of length greater than one makes it easier to examine the results. Lines corresponding to large parts of the entire roof boundary, telephone wires, and rain gutters are connected. The only major problem which has been encountered is overmerging of lines, particularly

in textured areas where accidently collinear line segments occur (such as in the lines in the roof of Figure 3). If texture is present, then the size of the gaps and their total length compared to the length of the replacement line become important. This measurement is called the *coverage* [14] of a line and indicates the extent to which a line is supported by actual intensity variations extracted from the image. Thus, lines with gaps (such as a dashed line) which are grouped into a new token will have a lower value of coverage as the size of the gaps relative to the length of the lines increases. Figure 11 shows the effect of filtering by coverage; there is some improvement in the textured areas, but the problem is not eliminated. More significantly, this is a two-dimensional problem, not a one-dimensional one. Lines which are part of a texture pattern will have a high density of similar lines surrounding them, but lines corresponding to a contour boundary often will not. This is analogous to the example in Lowe and Binford [18]: groups of collinear points are difficult to detect when they are embedded in a random distribution of points with the same interpoint distance. Thus, the algorithm could be improved by using the density of lines to inhibit linking when the density is high. In our implementation, the linking radius depends on the scale, but it should also depend on the density of lines. If there is a high density of lines with different orientations, we would only perceive a straight line if the gaps between the fragments were very small.

We have performed a few tests to examine the effects of filtering to remove very low contrast lines which are usually texture. Figures 12 and 13 show the resulting lines after successive filtering. Figure 14 shows the results with filtering for the other images in Figure 3.

6. Conclusion

We have outlined an approach to intermediate processing which uses symbolic tokens and relational measures between those tokens as the basis for grouping them. The relational measures are based on principles of perceptual organization, e.g. proximity, collinearity, and similarity of contrast. The valuable aspects of this approach are that symbolic tokens are closer to a representation needed for higher level interpretation processes and that it provides a framework for the selective integration of information which is distributed in an image.

The grouping algorithm has been tested on a variety of images, and the results as shown in this paper indicate that the use of geometric grouping is effective for extracting long, straight lines. The hierarchical linking and replacement algorithm was shown to be a computationally feasible approach to the extraction of straight line segments, and more sophisticated control of the replacement promises to reduce the computation and improve the results further. The hierarchy can be used to represent objects at different scales, and each level of the hierarchy has its own appropriate geometric context. Since token-based grouping has been successful in this particular application to straight lines,

it is a good candidate for generalization to more complex geometric structures; Reynolds and Beveridge [22] have done this for rectilinear line groups.

Acknowledgments

We wish to thank J. Brian Burns, Allan Hanson, Ed Riseman, and George Reynolds for many helpful discussions and suggestions.

REFERENCES

[1] Ballard, D. and C. Brown, *Computer Vision*, Prentice-Hall, New Jersey, 1982.

[2] Ballard, D. and D. Sabbah, "Viewer Independent Shape Recognition", *PAMI*, Vol 3, No. 6, pp. 653-660, Nov 1983.

[3] Barrow, H. and J. Tenenbaum, "Recovering Intrinsic Scene Characteristics From Images", *Computer Vision Systems*, A. Hanson and E. Riseman, eds., Academic Press, New York, pp. 3-35, 1978.

[4] Brady, M. and H. Asada, "Smoothed Local Symmetries and Their Implementation", *The First International Symposium on Robotics Research*, M. Brady and R. Paul, eds., MIT Press, Cambridge, 1984.

[5] Blum, H., "Biological Shape and Visual Science", *J. of Theoretical Biology*,Vol. 38, pp. 205-287, 1973.

[6] Burns, J.B., A. Hanson, and E. Riseman, "Extracting Straight Lines", *PAMI*, Vol 8, No. 4, pp. 425-455, 1986.

[7] Burt, P.J., C. Yen, and X. Xu, "Multi-resolution flow-through motion analysis", *IEEE CVPR Conference Proceedings*, pp.246-252, June 1983.

[8] Canny, J., "A Computational Approach to Edge Detection", *PAMI*, Vol 8, No 6, pp. 679-698, 1986.

[9] Davis, L., "A Survey of Edge Detection Techniques", *Computer Graphics and Image Processing*, Vol 4, pp. 248-270, 1975.

[10] Draper, B., R. Collins, J. Brolio, J. Griffith, A. Hanson, and E. Riseman, "Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes",*DARPA Image Understanding Workshop*, Los Angeles, pp. 178-193, 1987.

[11] Fischler, M. and R. Bolles, "Perceptual Organization and the Curve Partitioning Problem", *Proceedings IJCAI 1983*, pp.1014–1018.

[12] Hanson, A. and E. Riseman, "The VISIONS Image Understanding System – 1986", to appear in *Advances in Computer Vision*, ed. C. Brown, Erlbaum, 1987 (also appeared as UMASS COINS Tech. Report 86-27, Dec. 1986.)

[13] Haralick, R.M., "Digital Step Edges from Zero Crossings of Second Directional Derivatives", *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol 6, pp. 58-68, 1984.

[14] Herman, M. and T. Kanade, "The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images", *DARPA Image Understanding Workshop*, pp. 137-148, 1984.

[15] Leeuwenberg, E., "Quantification of Certain Visual Pattern Properties: Salience, Transparency, Similarity" *Formal Theories of Visual Perception*, E. Leeuwenberg and H. Buffart, eds., John Wiley and Sons, pp. 277-298, 1978.

[16] Lowe, D.G., *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985.

[17] Lowe, D.G. and T.O. Binford, "Segmentation and Aggregation: An Approach to Figure Ground Phenomena" *DARPA Image Understanding Workshop*, 1982, pp. 168-178.

[18] Lowe, D.G. and T.O. Binford, "The Perceptual Organization as a Basis for Visual Recognition", *Proceedings of AAAI-83*, Washington, DC, pp. 255-260, 1983.

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

[20] McKeown, D.M., and J.F. Pane, "Alignment and Connection of Fragmented Linear Features in Aerial Imagery", CMU Tech. Report CMU-CS-85-122, 1985.

[21] Nevatia, R., and K.R. Babu, "Linear Feature Extraction and Description", *CGIP*, v.13, pp. 257-269, 1980.

[22] Reynolds, G., and J.R. Beveridge, "Searching For Geometric Structure in Images of Natural Scenes", UMASS COINS Tech. Report 87-03, 1987.

[23] Riseman, E. and A. Hanson, "A Methodology for the Development of General Knowledge-Based Vision Systems", UMASS, COINS Tech. Report 86-27, July 1986.

[24] Riseman, E., A. Hanson, and R. Belknap, "The Information Fusion Problem: Forming Token Aggregations Across Multiple Representations", UMASS COINS Tech. Report 87-44, 1987.

[25] Weiss, R. and M. Boldt,"Geometric Grouping Applied to Straight Lines", *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, Miami Beach, pp. 489-495, June 1986.

[26] Weiss, R., E. Riseman, and A. Hanson,"Geometric Grouping of Straight Lines", *Proc. DARPA Image Understanding Workshop*, Miami Beach, FL, pp. 443-447, 1985.

[27] Witkin, A. "Scale-Space Filtering",*Proceedings of IJCAI*, pp. 1019-1021, Karlsruhe, 1983.

[28] Yuille, A.L., and T. Poggio, "Scaling Theorems for Zero-crossings", MIT A.I. Memo 730, 1983.

Figure 3: Digitized images of a) house scene, b)house scene, c) aerial image of runway, d) road scene.

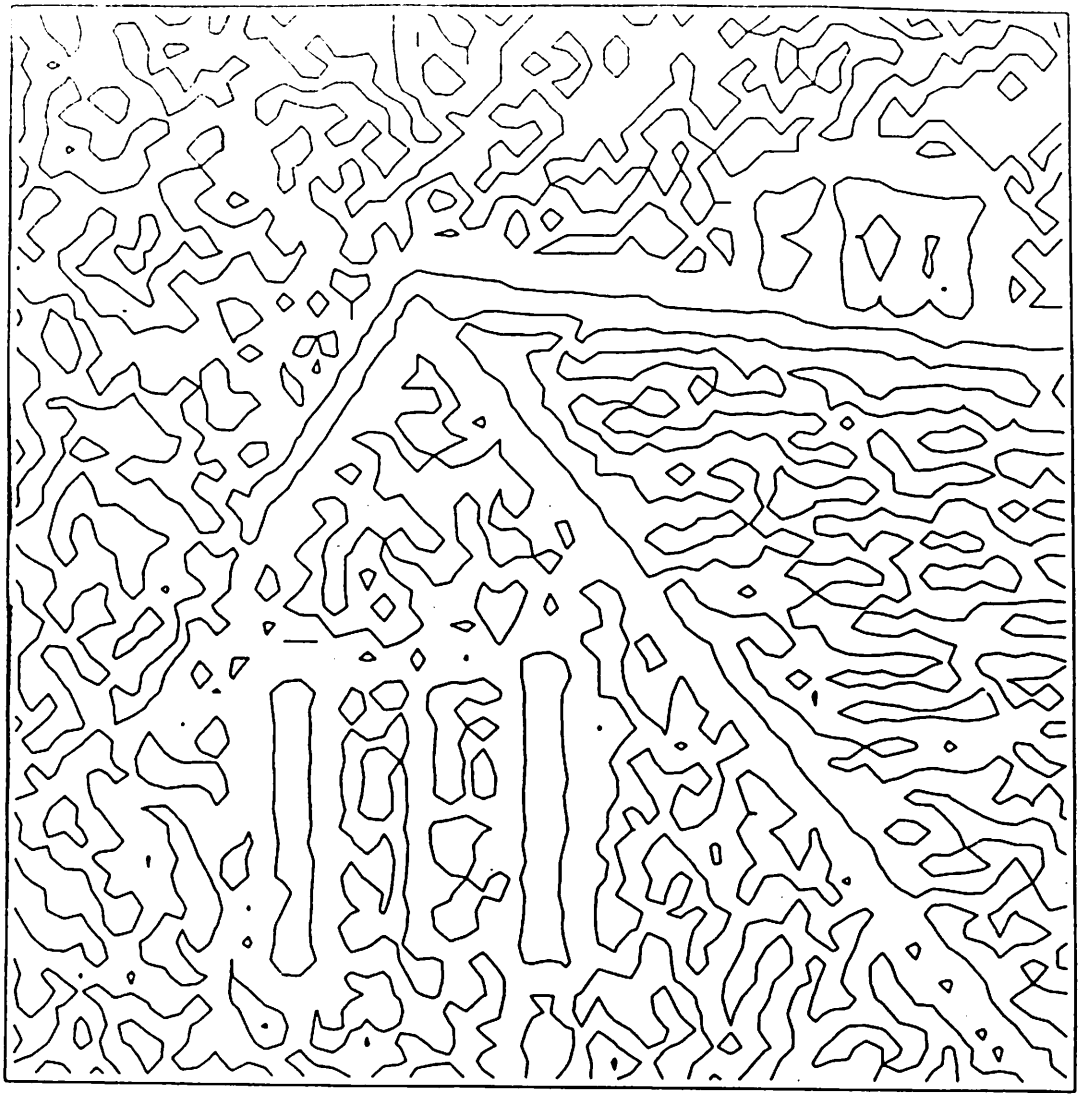


Figure 4a: Zero crossing contours of the Laplacian for part of a house scene. Many of the boundaries are fragmented.

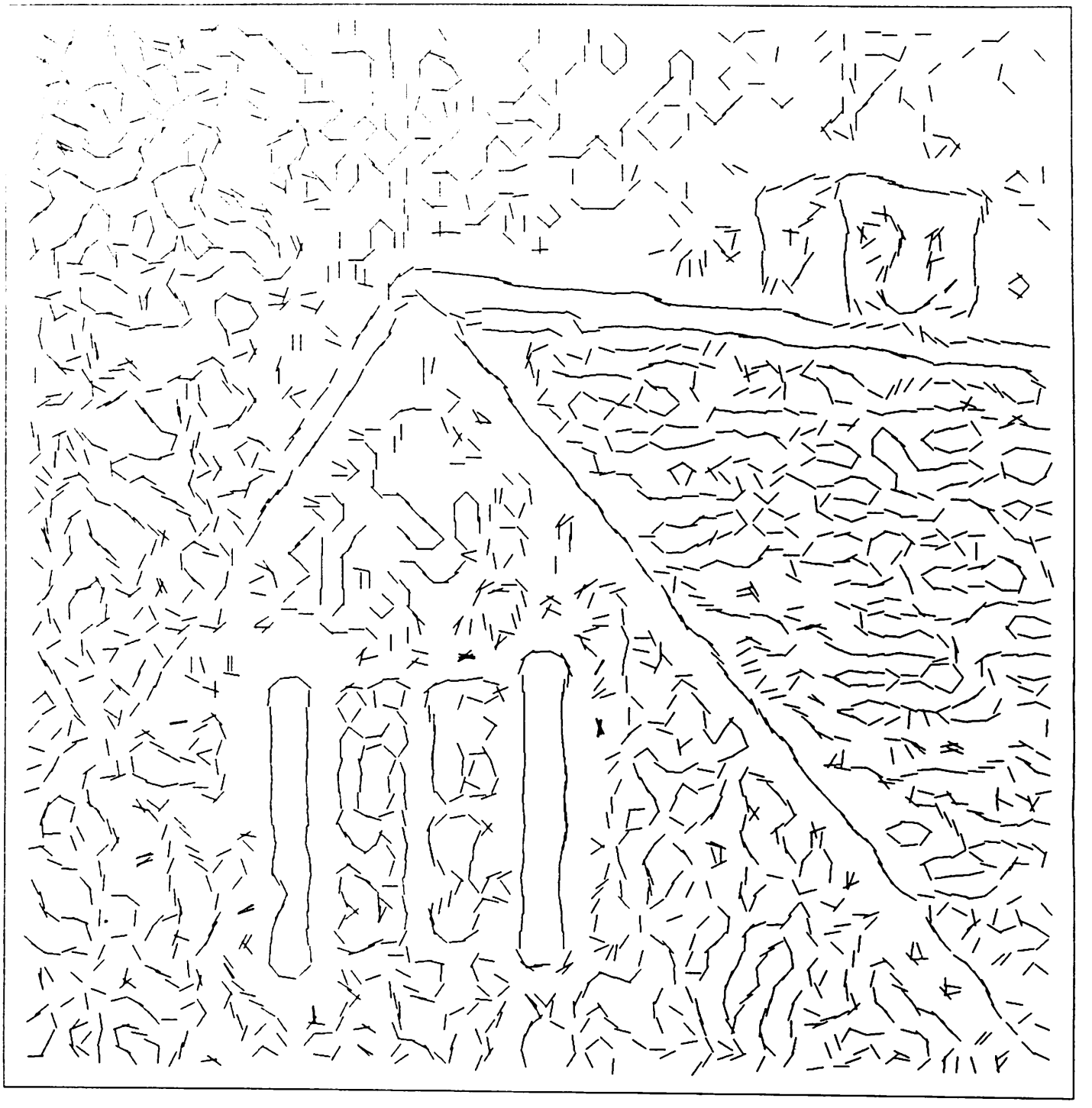


Figure 4b: **Edges from the zero crossing points of the Lapalacian; edge directions are determined by the gradient.**

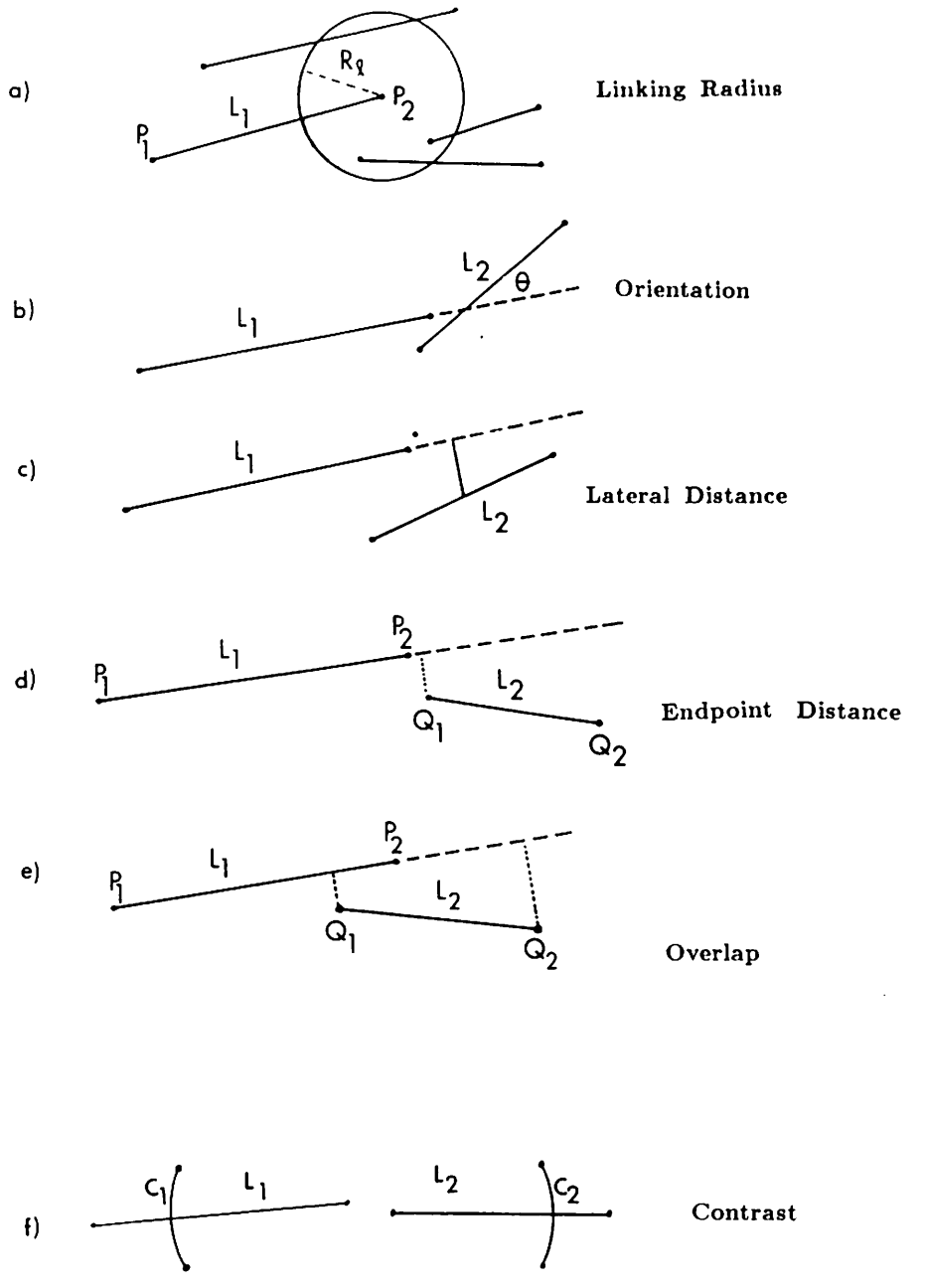
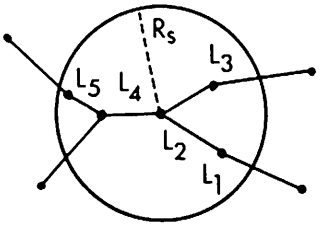
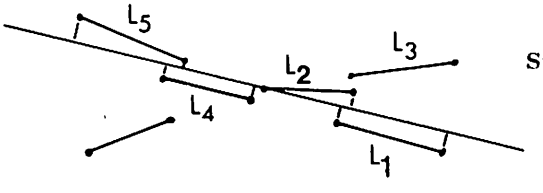


Figure 5: Criteria for linking (a)–(f) and grouping (g)–(h): (a) Proximity: lines are linked with L_1 if they pass within the linking radius R_l from endpoint P_2, where R_l is related to the length of L_1; (b) Orientation: lines must have similar orientation; (c) Lateral distance: lines must be close in the lateral direction as measured by the distance of the midpoint of L_2 perpendicular to the extension of L_1; (d) Endpoint distance: the endpoint P_2 must be close to the projection of the endpoint Q_1; (e) Overlap: the lines must not overlap too much; the distance from the projection of Q_1 to P_2 must be small compared with the distance from P_2 to the projection of Q_2; (f) Contrast: the ratio of the contrast of L_1 to the contrast



Link Graph

(g)



Straightness

(h)

of L_2 must be close to 1; (g) Link Graph: for each line there is a candidate group of lines which are possible extensions from each endpoint; (h) Straightness test: each sequence of lines is tested and the straightest is selected if it passes a threshold test.

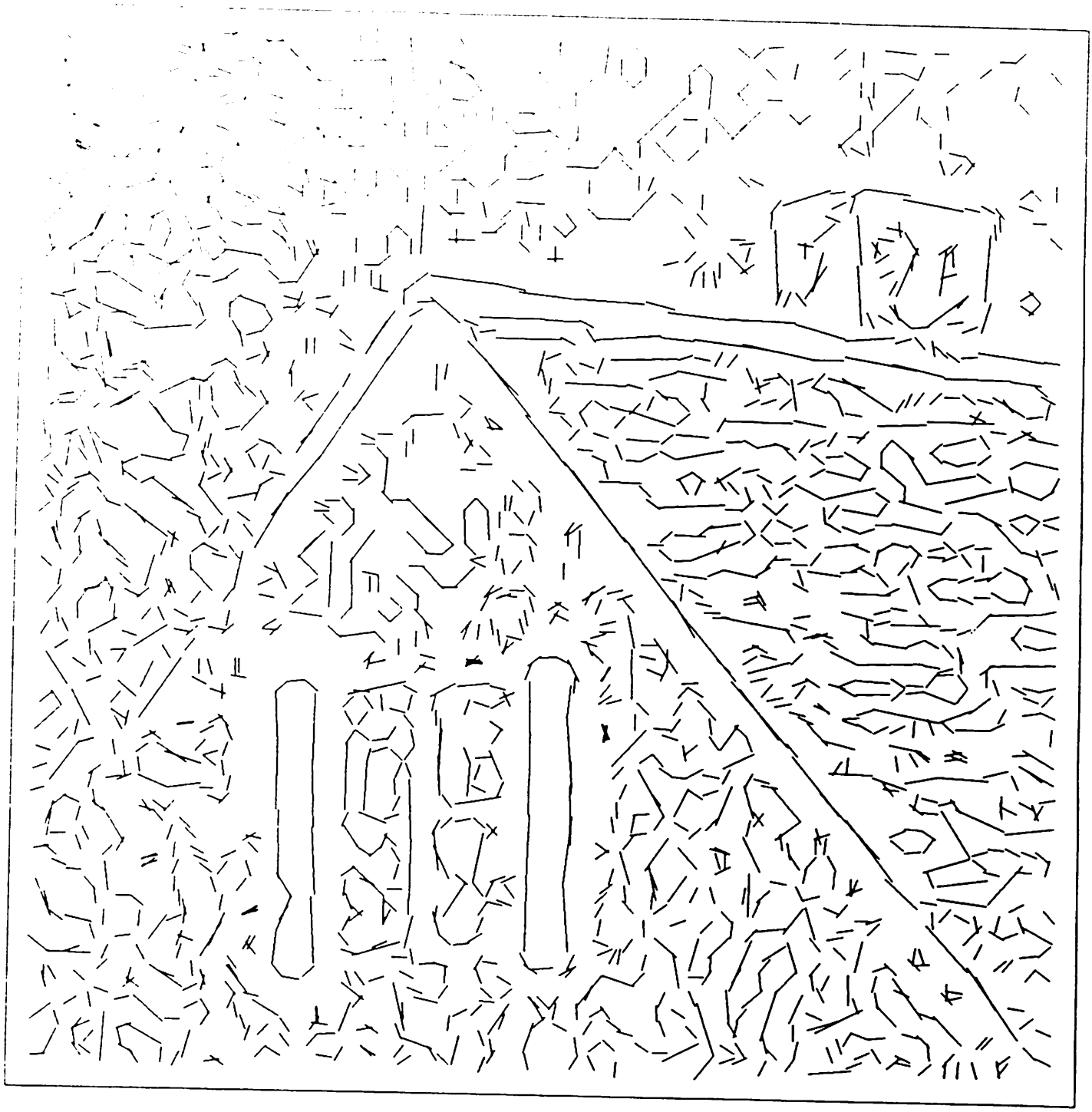


Figure 6a: Grouping algorithm after two cycles.

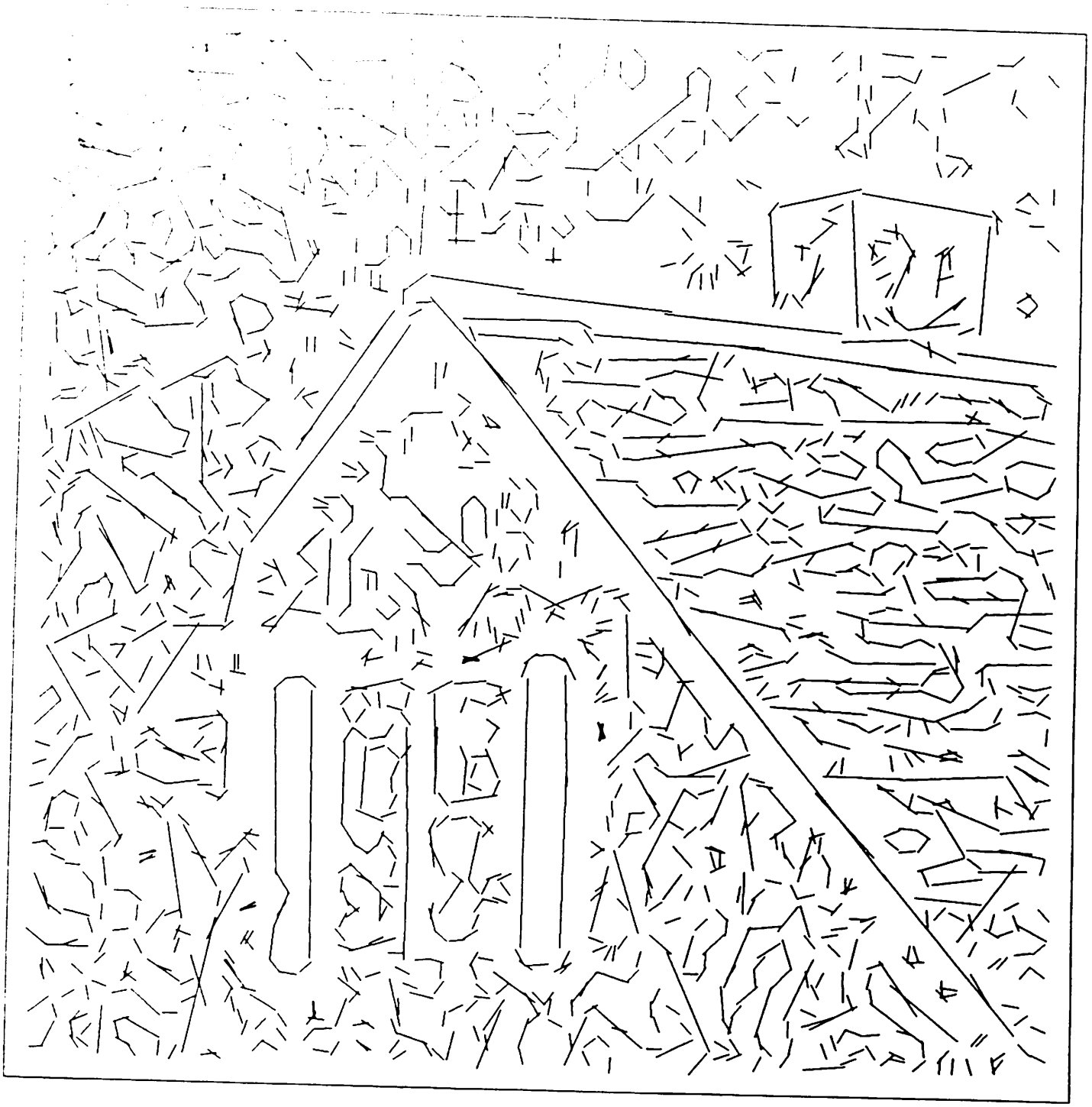


Figure 6b: Grouping algorithm after four cycles.

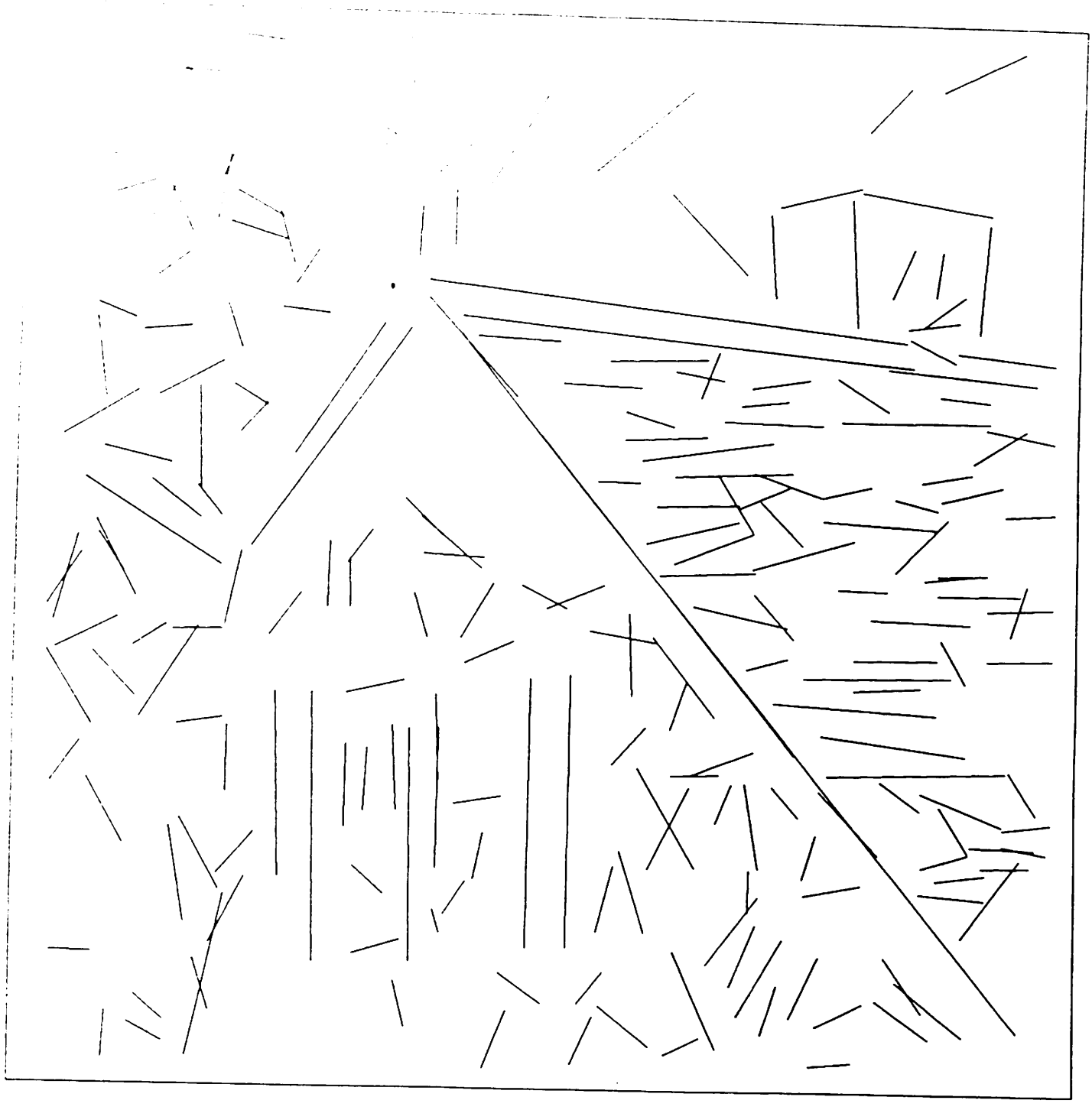


Figure 6c Grouping algorithm after six cycles. The right diagonal of the roof is spanned by a single line token; however, there are other smaller line tokens which overlap it.

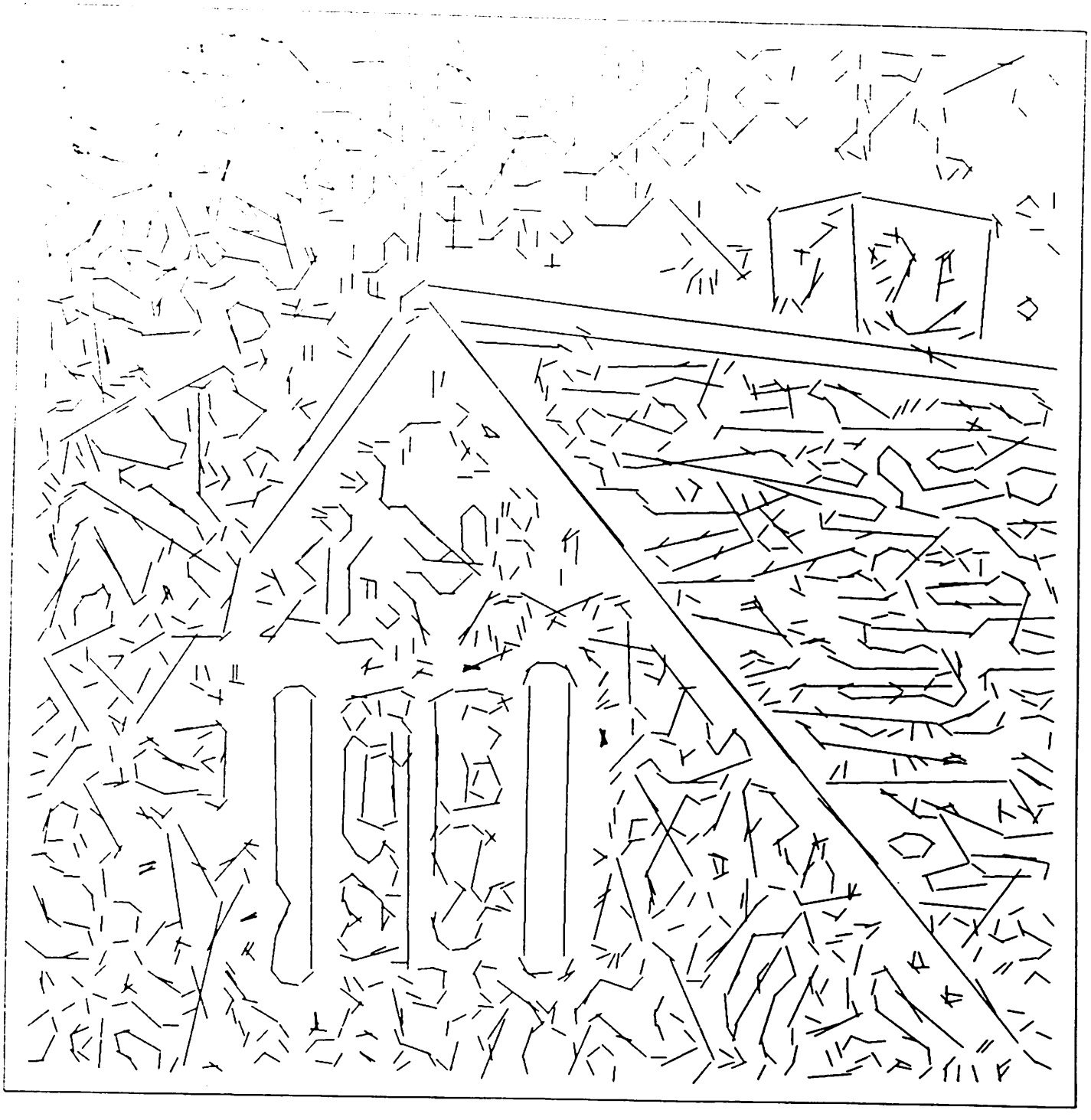


Figure 6d : The output of the grouping algorithm showing lines at multiple scales. Short, unmerged lines from a small scale which are "covered" approximately by longer lines could also be removed.

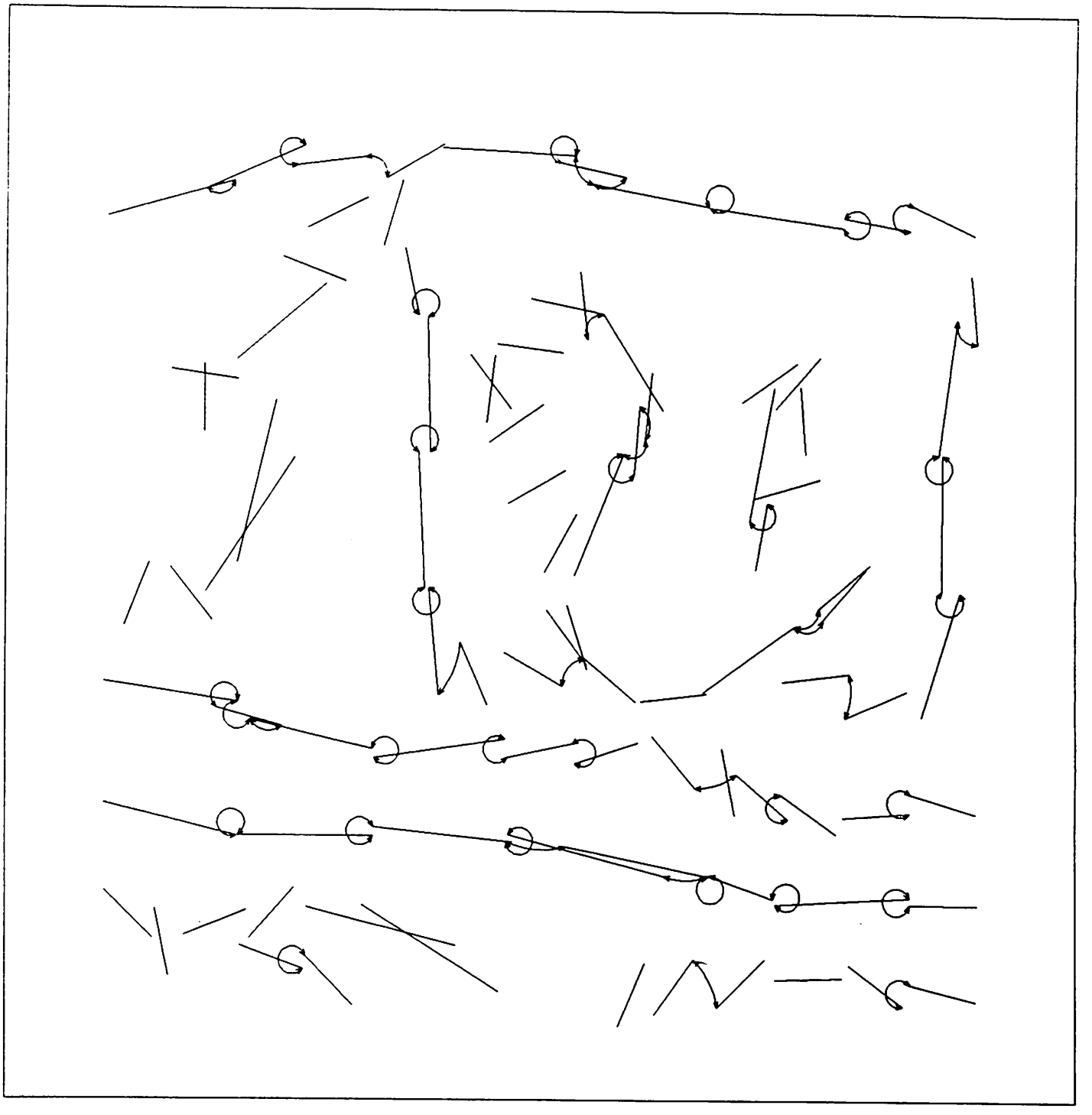


Figure 7a: Chimney subimage: the linking process is applied to the output of the first cycle. Links are shown as circular arcs.

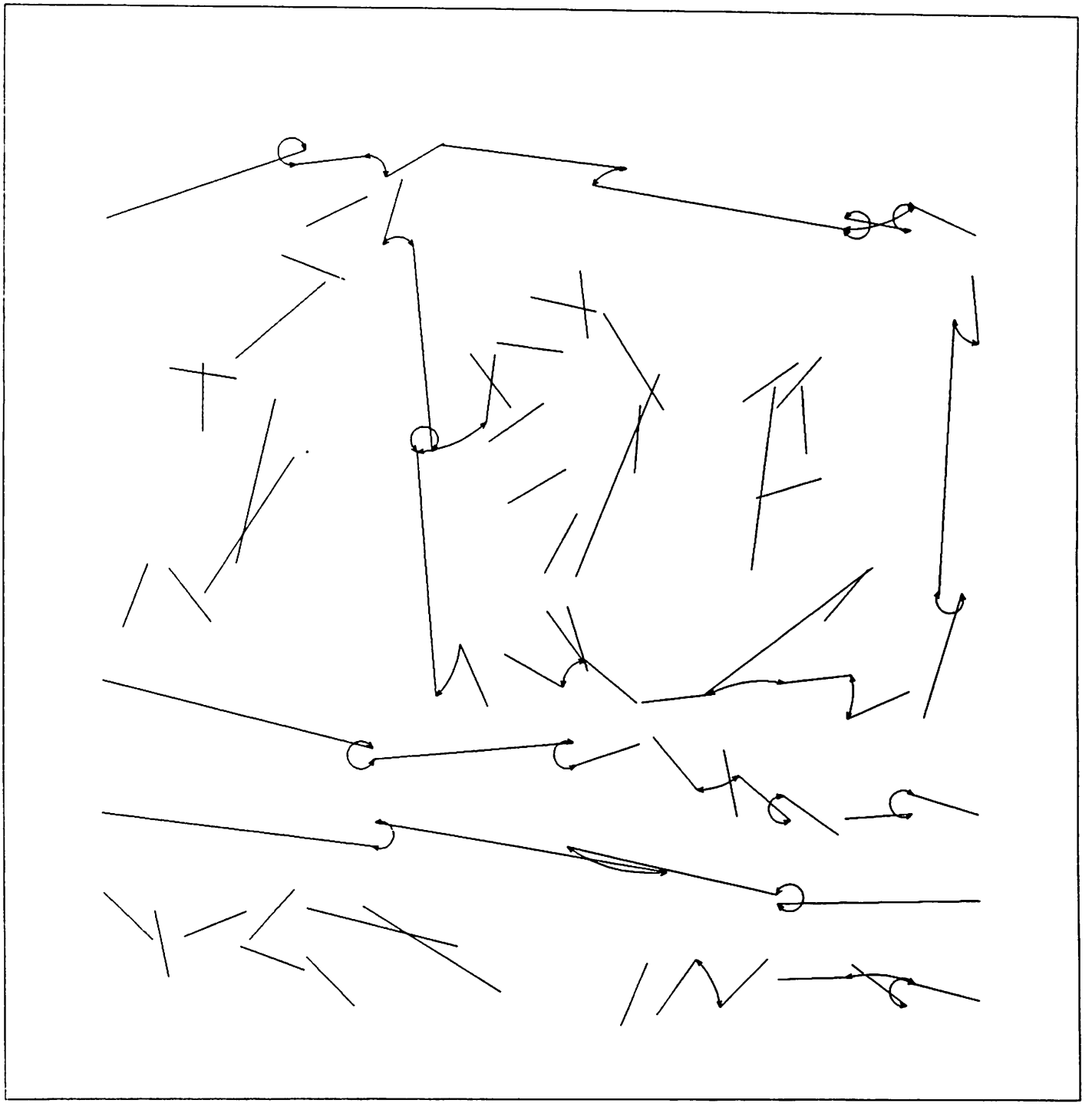


Figure 7b: **Chimney subimage: the linking process is applied to the output of the second cycle. Links are shown as circular arcs.**

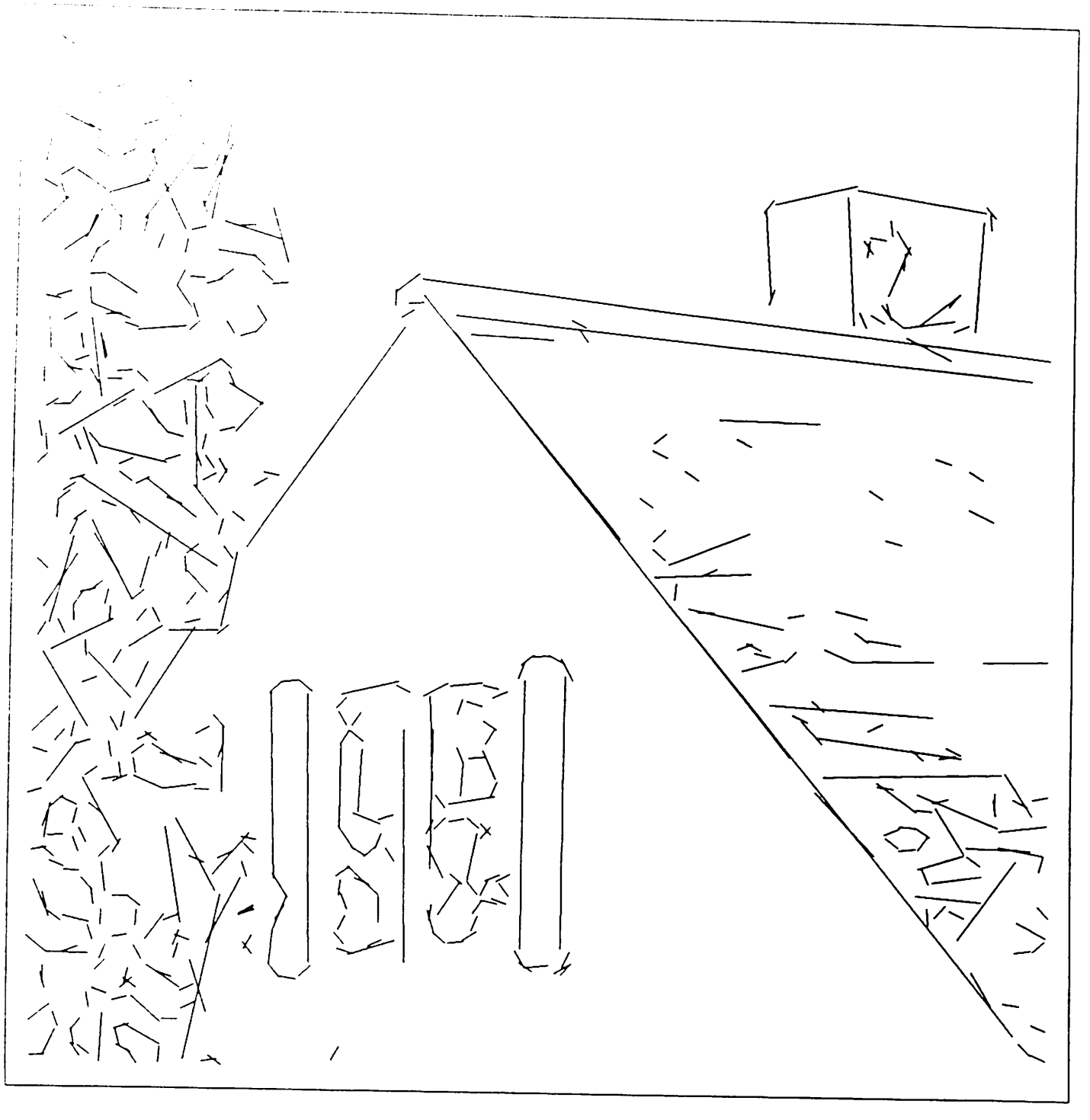


Figure 8: Filtering on gradient magnitude removes low contrast lines. The threshold was set at 15 (for a range of 255).

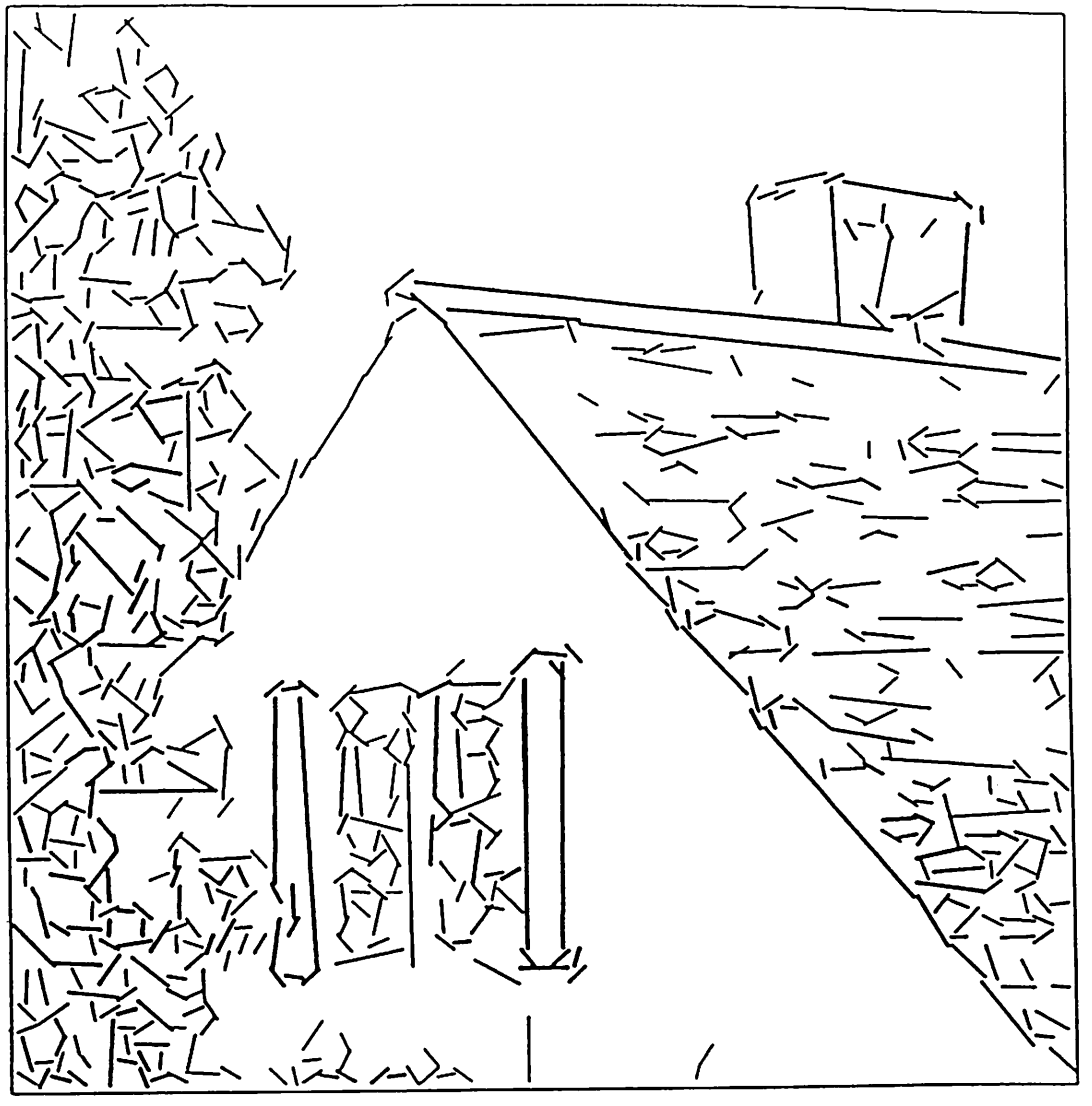


Figure 9: Output from Burns straight line algorithm, filtered on contrast. The locations of lines is similar to the grouping algorithm, but there is more fragmentation.

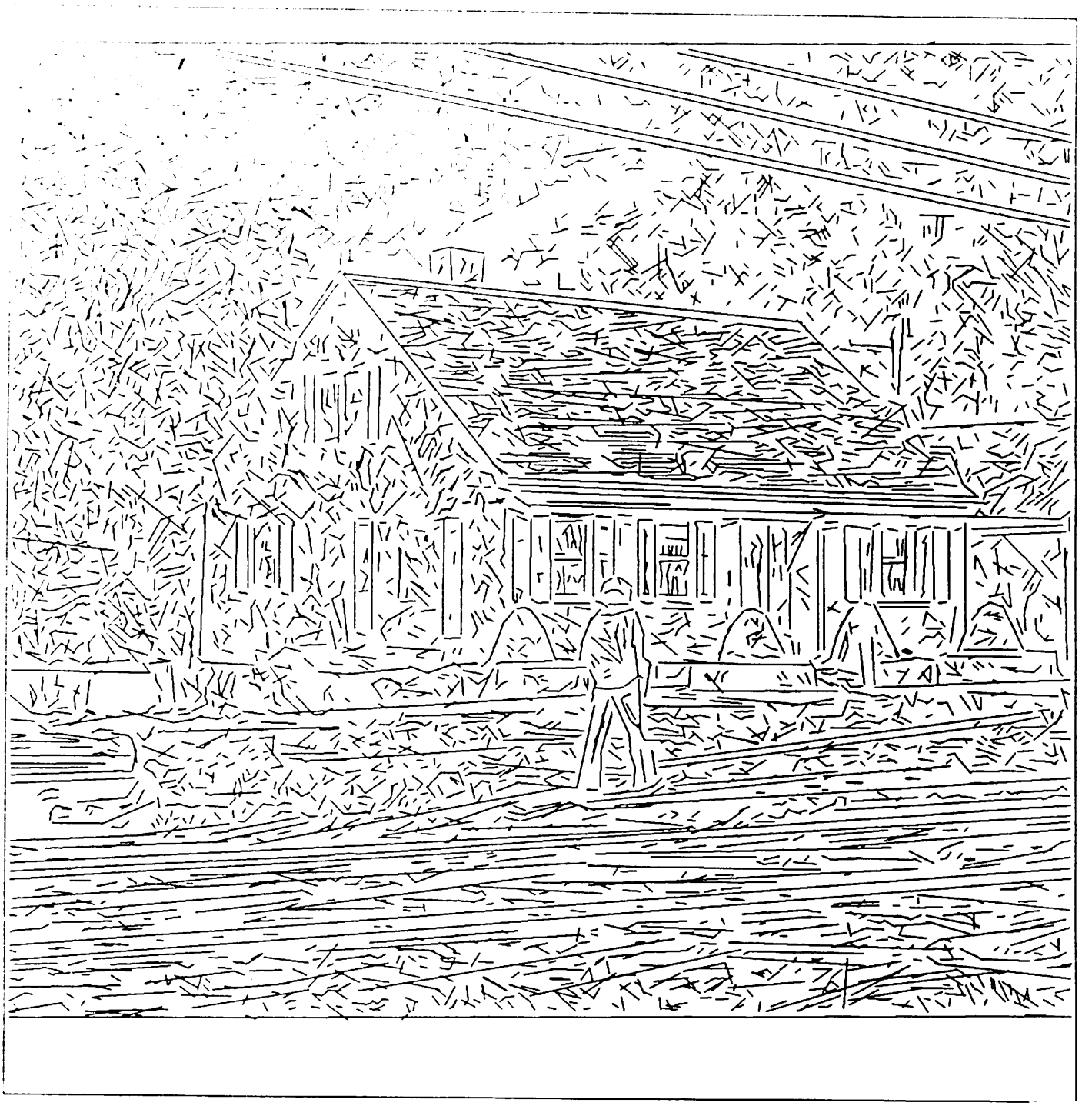


Figure 10: Final output of grouping algorithm on entire image. Lines of length one have been removed. Lines at all scales are shown.

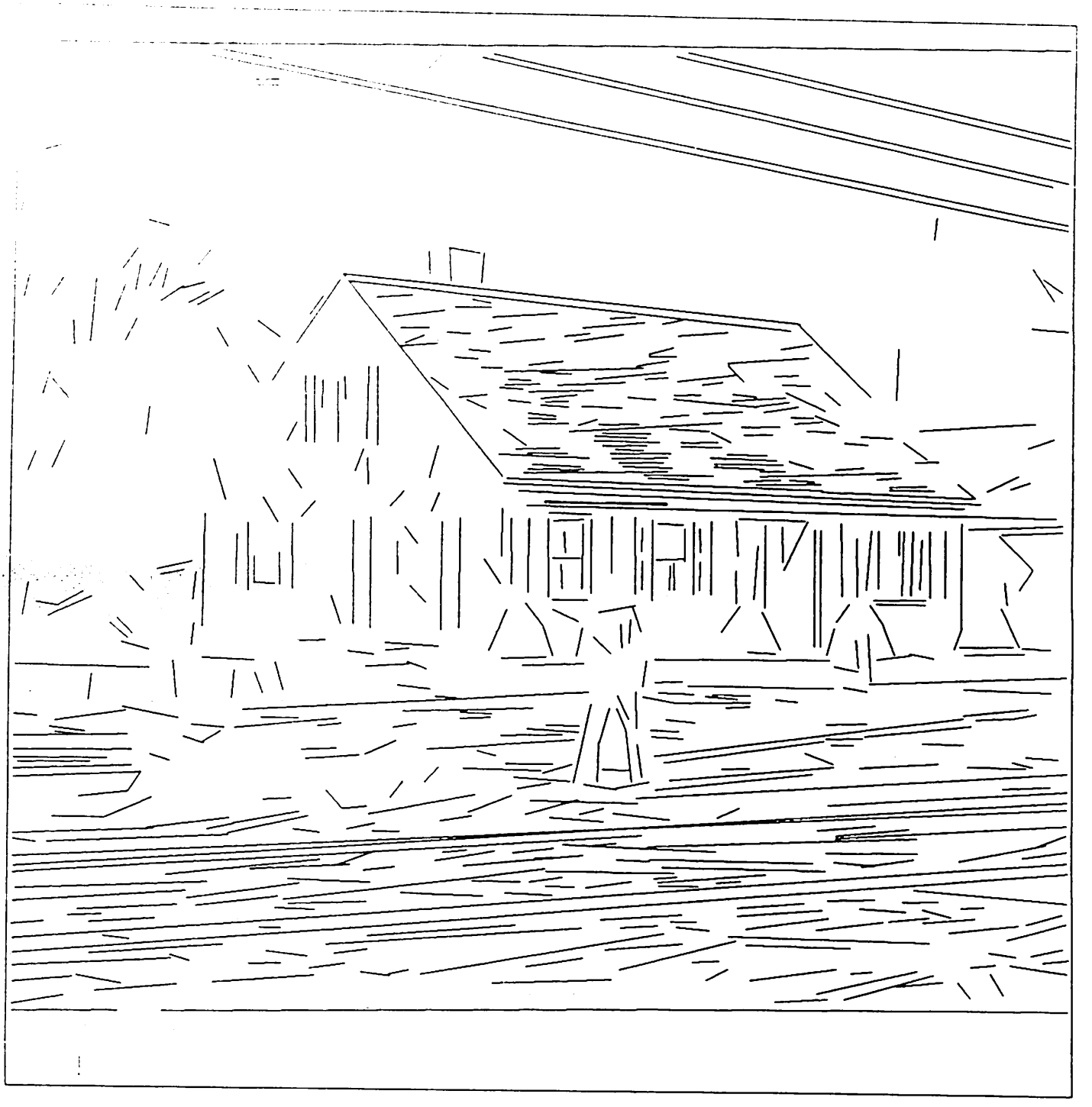


Figure 11: Lines from the house scene which have been filtered for a minimum length 5, a minimum gradient magnitude 5, and a minimum coverage of 90 percent.

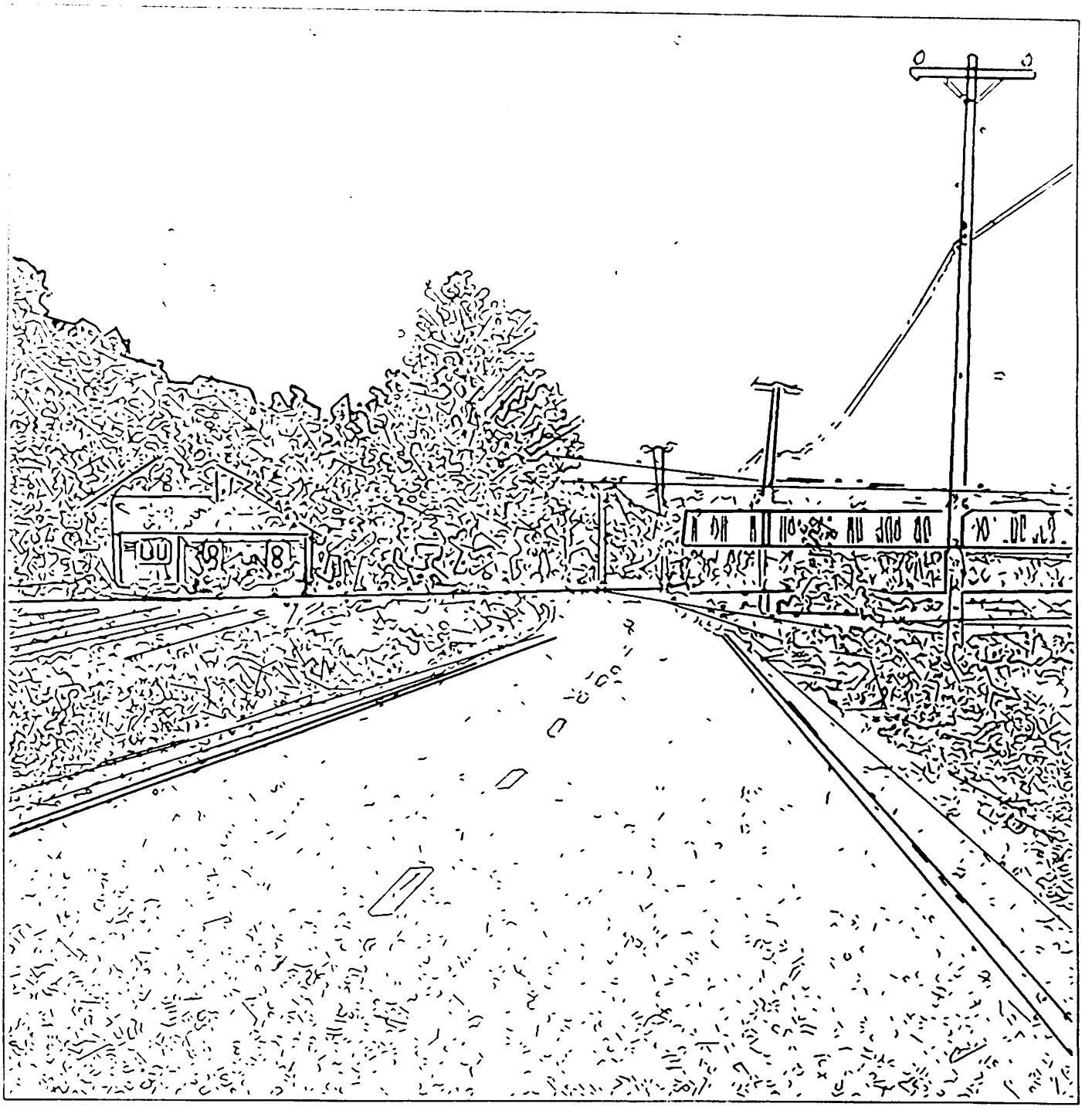


Figure 12: Lines in the road scene, filtered to remove all lines with constrast less than 5 (out of 255). Contrast is measured by average gradient magnitude. Lines with contrast greater than 30 are shown thicker.

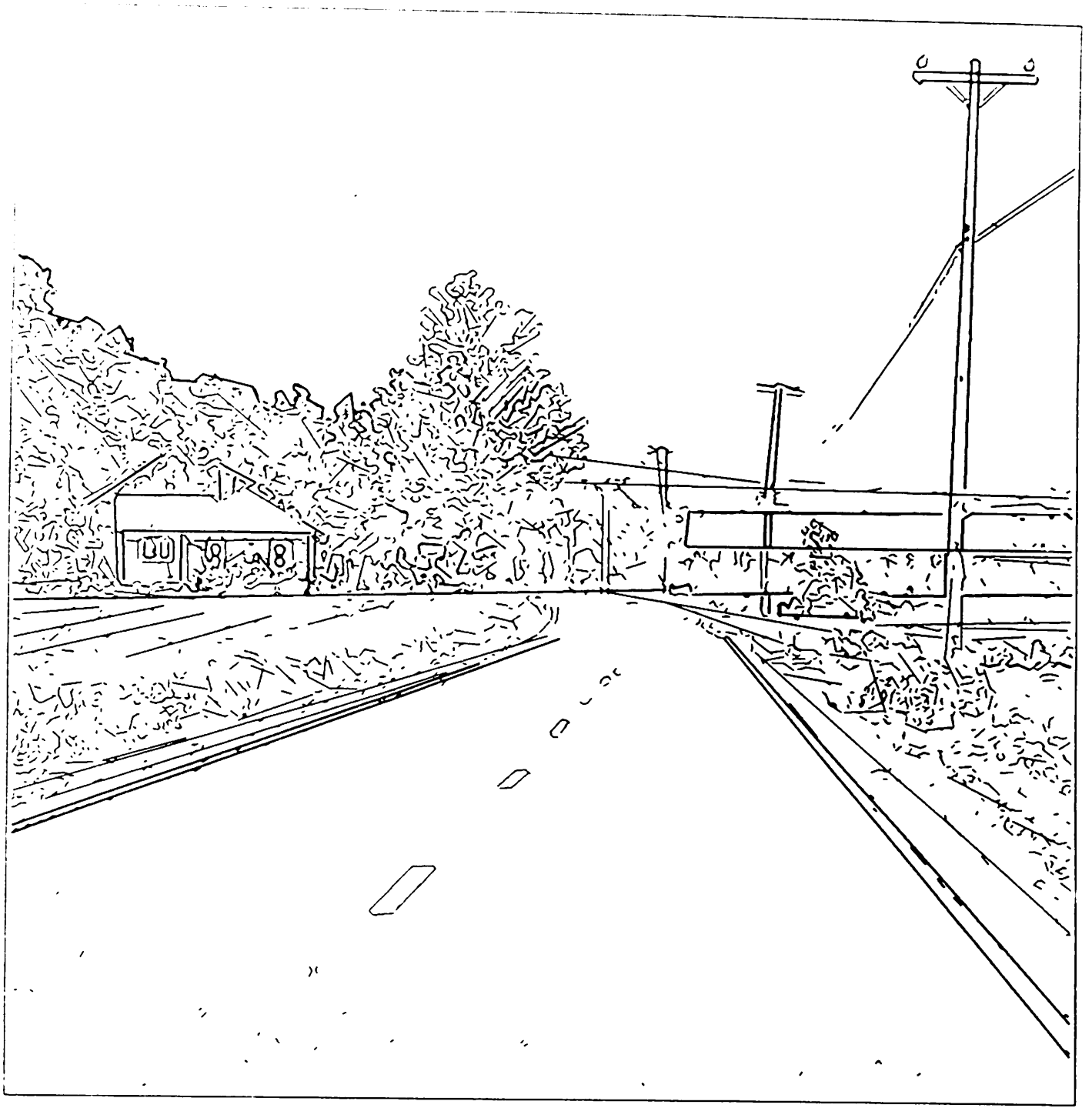


Figure 13: Lines in the road scene, filtered to remove all lines with contrast less than 10. Lines with contrast greater than 30 are shown thicker.

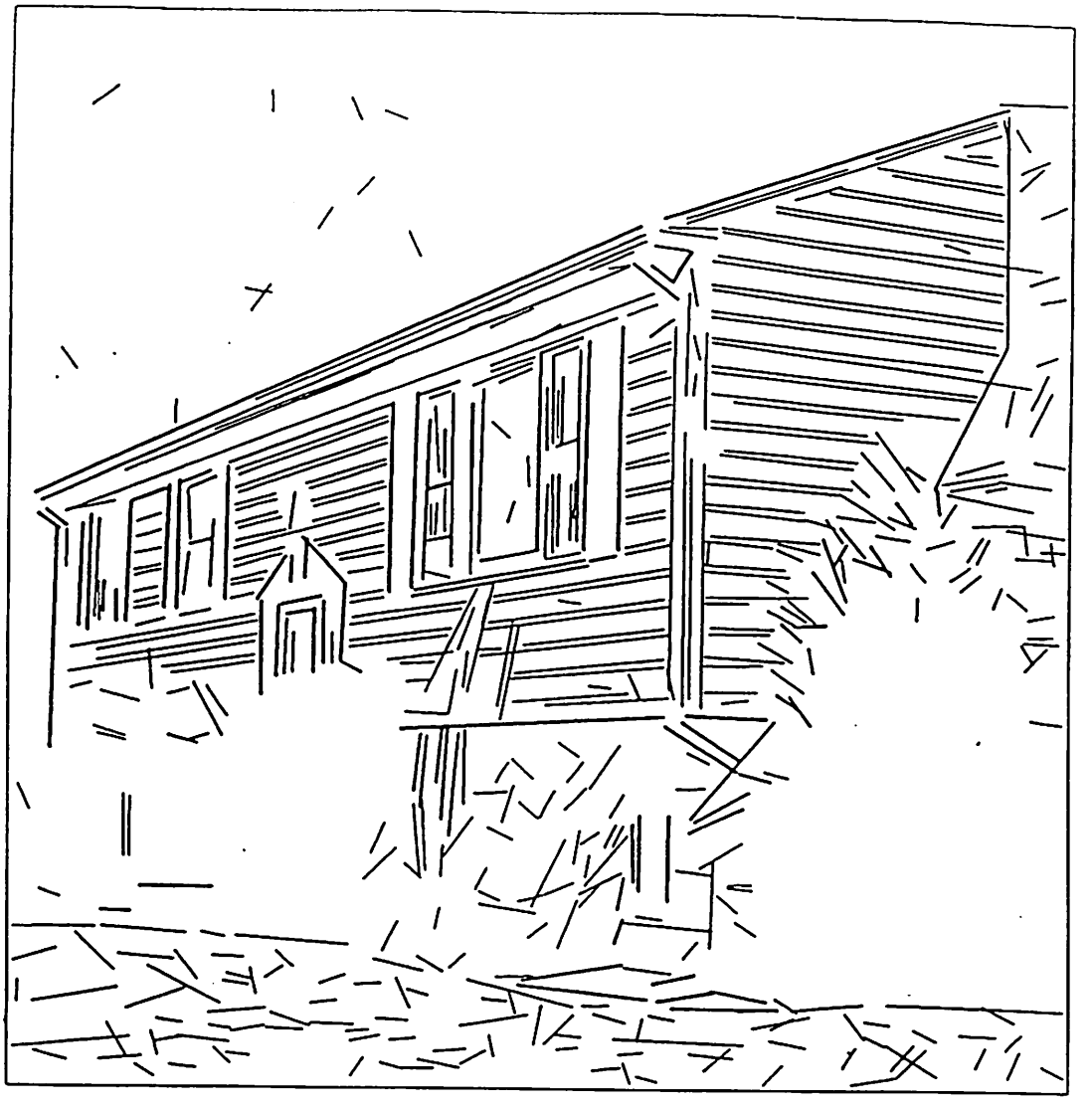


Figure 14: Output of the geometric grouping algorithm for house image in Figure 3, filtered on length.

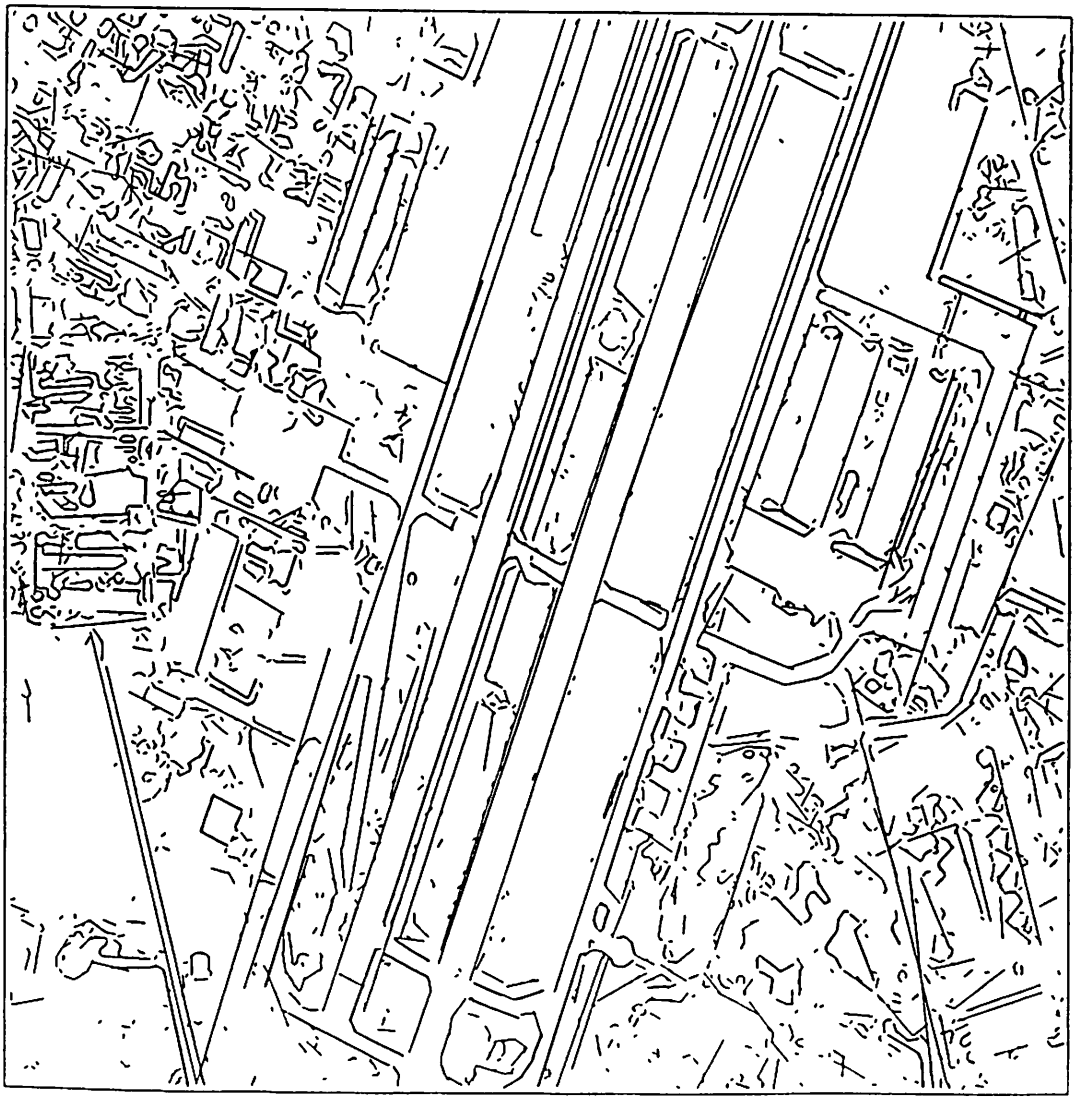


Figure 15: Output of the geometric grouping algorithm for aerial image in Figure 3, filtered on contrast.