1. Introduction

The purpose of perceptual grouping is to organize image data in a scene without the aid of semantic knowledge. The resulting groups identify significant structural relationships which are helpful in scene interpretation. The perceptual organization process starts with the primitives or tokens obtained from the images and groups "similar" tokens. It groups dots into blobs, blobs into larger blobs, edge segments into longer curves or complex structures, etc. The similarity among tokens is defined in terms of intrinsic properties of primitives and their spatial relationship in the image plane. Previous work closely related to grouping and extraction of perceptual structure in vision includes the clustering of d-dimensional dot patterns, orientation detection in glass patterns and dot patterns, the extraction of structure from a raw primal sketch, and clustering of one-dimensional line segments.

The principles of perceptual organization, which have been studied by Gestalt psychologists and others, include organization based on proximity, collinearity, curvilinearity, symmetry, parallelism, repetition, and closure. However, it is still very difficult to define quantitative similarity measures among tokens. Additional problems arise due to conflicts between the rules of perceptual organization applied to a given pattern.

The proximity principle involves relative distances between tokens in a local region. The gap between two tokens is compared with other gaps around it to quantify the relative distances between the tokens. The local region for this comparison needs to be defined. Zahn used the minimum spanning tree of a dot pattern for this purpose. Another way of describing the relevant local relationships between tokens is to use the Voronoi tessellation and its dual, the Delaunay graph, proposed by Ahuja and Tuceryan and Ahuja. In this description, two tokens are neighbors only when they are connected by a Delaunay edge. The Delaunay graph in Ahuja and Tuceryan and Ahuja was defined for dot patterns. The neighbor relationship in patterns containing arbitrary tokens, however, depends not only on the positions of tokens but also on their spatial extent. The original definition of the Voronoi tessellation, therefore, needs to be modified in order to accommodate properties of the arbitrary tokens which the dots do not possess.

The perceptual structures identified in the input patterns are based on inter-token distances, the intrinsic properties of tokens such as orientations, and the local interaction of neighboring tokens to enforce Gestalt constraints. Thus, the final groupings are the results of the integration of local geometric information through the use of global constraint propagation. Most of the time, this process will result in a single strong grouping. However, in some cases, ambiguities can arise during the grouping process.

This paper is concerned with the extraction of perceptual structure in line patterns. Line patterns consist of tokens which can be either dots or variable length straight line segments. Our goal is to design an algorithm to group the line patterns to form longer curves based on proximity, smoothness, collinearity, and curvilinearity. The algorithm uses the neighbor relationship defined by the Voronoi tessellation and obtains the grouping based on the distances between line segments, their orientations, and their local contexts. To test our algorithm, we have applied it to synthetic patterns which contain ambiguities and interactions between the position and the orientation properties. We have also tested our grouping algorithm on thresholded gradient images obtained by applying edge operators to intensity images.

* This research was supported in part by NSF Grant ECS-8603541.
Section 2 describes the Delaunay graph, probabilistic relaxation labeling scheme, and the grouping algorithm. Section 3 gives the experimental results of applying the algorithm to several line patterns. Finally, conclusions and future research topics are discussed in Section 4.

2. Perceptual Grouping for Line Patterns

Our goal is to identify perceptual structures present in a given line pattern. Our grouping process consists of two major steps.

(i) Defining a neighbor relationship among the tokens based on the Delaunay graph.

(ii) Deleting "inconsistent" edges of the Delaunay graph to form the groups.

Construction of a Delaunay graph for a dot pattern is fairly straightforward but requires some modification for line patterns as discussed below. Each edge in the Delaunay graph is assigned two labels, \textit{NTBRK} (not break) and \textit{BRK} (break) with associated weights (between 0 and 1). These labels and weights are updated iteratively based on the interaction of each line segment with its neighbors. The final labels determine which edges in the Delaunay graph are to be deleted or found inconsistent. The weights or measures of certainty associated with the edge labels enable us to represent ambiguous groupings of the line pattern.

2.1. Delaunay Graph

A Delaunay graph is best described in terms of Voronoi tessellation. The Voronoi tessellation of a set of dots in a plane is the partition of the plane \((\mathbb{R}^2)\) into regions such that the region assigned to a dot \(P\) consists of all the points in \(\mathbb{R}^2\) which are closer to \(P\) than to any other dot. This results in a polygonal region assigned to each dot, called a Voronoi polygon.\(^{14,11}\) The set of complete Voronoi polygons together with the incomplete polygons in the convex hull of the dots define a Voronoi tessellation of the entire plane. Two dots are said to be Voronoi neighbors if the Voronoi polygons enclosing them share an edge. The dual representation of the Voronoi tessellation is the Delaunay graph which is obtained by connecting all the pairs of dots which are Voronoi neighbors.

In constructing a Voronoi tessellation for the line patterns, we use the same definition as above, only the tokens may be lines in addition to dots. Algorithms have been developed for computing the Voronoi tessellation of a pattern consisting of dots and straight lines.\(^5\) In this research, however, we compute the Voronoi tessellation of a given set of tokens using a more general algorithm which works for arbitrary shaped tokens as well as for line segments. The Voronoi tessellation is computed by expanding the tokens in the image until the boundaries of the expanding regions meet. When the boundaries of two tokens meet, the two tokens become Voronoi neighbors, and the boundary at these points freezes. This expansion process is continued until all the pixels in the image are visited exactly once. The Delaunay edges between pairs of tokens are represented by connecting the closest points between the two tokens. Figure 1(b) shows the Delaunay graph for the example line pattern in Figure 1(a).

2.2. Probabilistic Relaxation Labeling Scheme

Probabilistic relaxation labeling is a technique of parallel constraint propagation for obtaining locally consistent labels of a set of objects.\(^9\) Following is a brief description of the probabilistic relaxation formulation of our grouping algorithm. The objects to be labeled are the Delaunay edges and the two possible labels on the objects are \textit{NTBRK} and \textit{BRK}. Let \(P_i(\lambda)\) be the probability that the label \(\lambda\) is "correct" for the \(i^{th}\) object, such that \(\sum P_i(\lambda) = 1\). These probabilities are initialized based on the local intrinsic properties of the line segments and are updated by using the compatibility coefficients between the neighboring line segments.

The probability \(P_i^{(k+1)}(\lambda)\) of label \(\lambda\) on edge \(i\) at iteration \((k+1)\) is obtained by using the following rule:

\[
P_i^{(k+1)}(\lambda) = \frac{P_i^{(k)}(\lambda)[1 + q_i^{(k)}(\lambda)]}{\sum_j P_j^{(k)}(\lambda)[1 + q_j^{(k)}(\lambda)]},
\]

where

\[
q_i^{(k)}(\lambda) = \sum_j w_{ij} \left[ \sum_{\lambda'} r_{ij}(\lambda,\lambda') P_j^{(k)}(\lambda') \right],
\]

\(r_{ij}(\lambda,\lambda')\) is the compatibility of the labels \(\lambda\) and \(\lambda'\) on edges \(i\) and \(j\) and \(w_{ij}\) is the coefficient which determines the contribution of the \(j^{th}\) edge towards the change in probability for the \(i^{th}\) edge. The quantity \(q_i^{(k)}(\lambda)\) is in the range \([-1,1]\), and it determines whether \(P_i(\lambda)\) is increased \((q_i^{(k)}(\lambda)>0)\), decreased \((q_i^{(k)}(\lambda)<0)\), or remains
unchanged \(q^{(B)}(\lambda) = 0\). The iteration continues until (i) all the probabilities converge, or (ii) the number of iterations exceeds a prespecified limit.

The selection of initial probabilities, \(P^0(\lambda)\), compatibility coefficients, \(r_{ij}(\lambda, \lambda')\), and the weights, \(w_{ij}\), is explained below.

2.2.1. Computation of Compatibility Coefficients

The compatibility coefficients embody the knowledge of local interaction of neighboring line segments and determine which combinations of labels on Delaunay edges are meaningful based on their relative distances and orientations. In this process, we only consider the compatibility relations between two neighboring Delaunay edges which share a common line segment. For example, if the compatibility coefficient, \(r_{ij}(\lambda, \lambda')\), for the label combination \((NTBRK, NTBRK)\) on two neighboring Delaunay edges \(i\) and \(j\) has a large value then the grouping of the three adjacent line segments which define edges \(i\) and \(j\) is supported. The label \(NTBRK\) on a Delaunay edge may arise from alignment or proximity of the corresponding two line segments.

The compatibility coefficients take into consideration the following three factors:

(i) Distance between two line segments is defined to be the minimum distance between them. In order to utilize intertoken distances for proximity based grouping, we compare them to the intertoken distances in their vicinity.

Suppose Delaunay edge \(i\) connects two line segments \(t_1\) and \(t_2\), and \(dis_i\) is the distance between them. Then the amount of distance contribution for edge \(i\), \(d_i\), is defined to be:

\[
    d_i = \frac{avg_i - dis_i}{\max(dis_i, avg_i)},
\]

where \(avg_i\) is the average length of the neighboring Delaunay edges. The value of \(d_i\) is between -1 and 1. When the distance between two line segments is relatively short, \(d_i\) is positive; otherwise, it is negative. A positive value of \(d_i\) contributes towards the label \(NTBRK\) for edge \(i\), whereas a negative value of \(d_i\) contributes toward the label \(BRK\) for edge \(i\).

(ii) Orientation agreement between two line segments is also important in defining the compatibility coefficients. When the line segments form a smooth curve, the total curvature of the resulting curve will be small. We compute a measure based on the angles between the line segments that reflects this fact.

Given two line segments, we connect their closest endpoints. This defines two angles \(\theta_1\) and \(\theta_2\) that each line segment makes with the connecting line. For a smooth curve the total angle change \(\theta_1 + \theta_2\) is expected to be small. We define our total angle change normalized within the range \([0,1]\) as \(x = (\theta_1 + \theta_2)/2\pi\). To compute the contributions for \(NTBRK\) and \(BRK\) we use the following formula:

Figure 1. (a) An example pattern, (b) its Delaunay graph, and (c) the result produced by the grouping algorithm.
When \( \theta_1 + \theta_2 \) is small the value of the \( a_i(NTBRK) \) is close to 1 and the value of \( a_i(BRK) \) is close to 0. As the total angle change \( \theta_1 + \theta_2 \) increases, \( a_i(NTBRK) \) decreases towards 0 and \( a_i(BRK) \) increases towards 1.

The orientation differences are not considered when the lengths of both the line segments are sufficiently short. In this case, we treat the line segments as dots and we set \( a_i(NTBRK) \) and \( a_i(BRK) \) to be 1 and 0, respectively.

The contributions \( L_{i,\alpha} \) of distance \( (d_i) \) and orientation agreement \( (a_i) \) between the tokens for Delaunay edge \( i \) and label \( \lambda \) can be combined as follows.

\[
\begin{align*}
L_{i,NTBRK} &= \min(d_i, d_i \times a_i(NTBRK)) \\
L_{i,BRK} &= \max(-d_i, d_i \times a_i(BRK))
\end{align*}
\]

(5)

If the tokens are far apart \( (d_i < 0) \) then the orientation information is not utilized, because \( a_i \) is in \([0,1]\). However, when \( d_i > 0 \), both the distance and orientation agreement information is used to label the Delaunay edges.

(iii) The angle between two neighboring Delaunay edges is used to enforce the smoothness of curves. We define this contribution, denoted by \( s_{ij} \), for edges \( i \) and \( j \) as:

\[
s_{ij} = \min(a_i(NTBRK), a_j(NTBRK))
\]

(6)

When the curve defined by the three line segments is smooth, \( s_{ij} \) is large. This factor only affects the compatibility coefficient for not breaking both the edges. That is, when two edges are not aligned, the compatibility coefficient of the label combination \( (NTBRK,NTBRK) \) should be reduced.

In our grouping algorithm, every Delaunay edge has two labels: \( NTBRK \) and \( BRK \). Thus, four compatibility coefficients, \( r_{ij} \), are calculated for two neighboring Delaunay edges \( i \) and \( j \). For each label combination \( (\lambda,\lambda') \), we sum the label contribution for the individual edges to obtain the compatibility coefficient as follows. Let

\[
r_{ij}(\lambda,\lambda') = \begin{cases} 
(L_{i,\lambda} + L_{j,\lambda'}) \times s_{ij} & \text{for } (\lambda,\lambda') = (NTBRK,NTBRK) \\
(L_{i,\lambda} + L_{j,\lambda'}) & \text{Otherwise.}
\end{cases}
\]

(7)

Notice that for the label combination \( (NTBRK,NTBRK) \), we also include the contribution from the difference between the angles of the neighboring Delaunay edges to ensure that the resulting grouping will be smooth. The compatibility coefficient is normalized to lie in the interval \([-1,1]\) by

\[
r_{ij}(\lambda,\lambda') = \frac{r_{ij}(\lambda,\lambda')}{\max_{\hat{\lambda},\hat{\lambda}'} r_{ij}(\hat{\lambda},\hat{\lambda}') - \min_{\hat{\lambda},\hat{\lambda}'} r_{ij}(\hat{\lambda},\hat{\lambda}')}.
\]

(8)

2.2.2. Assignment of Initial Probabilities

The initial probabilities \( P_i^0(\lambda) \) are assigned based on compatibilities. For example, if \( r_{ij}(\lambda,\lambda') \) is large then edge \( j \) provides a large contribution towards the initial probability \( P_i^0(\lambda) \). Since \( r_{ij} \) is between -1 and 1, \( P_i^0(\lambda) \), is computed as follows:

\[
P_i^0(\lambda) = \sum_j w_{ij} M_{AX} \left\{ \frac{r_{ij}(\lambda,\lambda') + 1}{2} \right\}
\]

(9)

Then, define

\[
P_i^0(\lambda) = \frac{\tilde{P}_i^0(\lambda)}{\sum_\lambda \tilde{P}_i^0(\lambda)}
\]

(10)

so that \( \sum_\lambda P_i^0(\lambda) = 1 \).

The coefficient \( w_{ij} \) in equations (2) and (9) represents the contribution of the edge \( j \) to its neighboring edge \( i \), computed as follows.
Let

\[ w_{ij} = \frac{\min_{k \in N(i)} dis_k}{\max_j dis_j}, \quad \text{and} \quad \bar{w}_j = |N(i)|^{-1}, \]  

(11)

where \( dis_j \) is the distance between the two line segments connected by Delaunay edge \( j \), \( N(i) \) is the set of neighbors for edge \( i \), and \( |N(i)| \) is the cardinality of \( N(i) \) (i.e. the number of neighbors of \( i \)). The coefficients are normalized by

\[ w_{ij} \leq \frac{\bar{w}_j}{\sum_j \bar{w}_j}, \]  

(12)

so that \( \sum_j w_{ij} = 1 \).

---

**Figure 2.** The flowchart for the grouping algorithm.

2.3. Algorithm

Our algorithm computes the final grouping by using the Delaunay graph for line patterns and the probabilistic relaxation labeling scheme described above. The flowchart of the algorithm is given in Figure 2. The final labeling is obtained by assigning to each Delaunay edge the label with the largest probability. If the final
label assigned to Delaunay edge $i$ is $NTBRK$ then edge $i$ is retained. On the other hand, if the final label assigned to edge $i$ is $BRK$ then edge $i$ is deleted. In the resulting graph, the connected components define our final grouping.

3. Experimental Results

Our perceptual grouping algorithm was tested both on synthetically generated patterns and on line patterns extracted from real images. The synthetic patterns were generated to test groupings based on proximity, smoothness, collinearity, and curvilinearity and possible conflicts among them. Figures 1 and 3 show two examples of such patterns and the resulting grouping. In Figure 1, the algorithm cannot group the complete circle because the line segments belonging to the circle which are separated by the side of the square are not Delaunay neighbors. In Figure 3, there is an ambiguity whether to group the straight lines with the sides of the square because of collinearity or whether to group them with the circular arcs because of proximity. Our algorithm groups the straight lines with the circular arcs, thus resolving the ambiguity in this example in favor of the proximity.

![Figure 3](image)

Figure 3. (a) An example pattern and (b) the result produced by the grouping algorithm.

Figure 4(b) shows a line pattern which was extracted from the gray level image of an outdoor view of a building shown in Figure 4(a). A gradient operator (Sobel) was run on the image and the result was thresholded. From the resulting edge image, straight line and dot tokens were extracted to obtain Figure 4(b). Finally, this resulting pattern was input to our grouping algorithm. Figure 4(c) shows the result produced by our algorithm.

In most cases, our grouping algorithm converges to the "correct" result in less than 10 iterations. But for some ambiguous cases, convergence is slower, taking about 100 iterations. Our algorithm for generating the Delaunay graph takes a constant amount of time for a fixed sized image regardless of the number of tokens. For an image of size 512x512 it takes about 20 seconds to construct the Delaunay graph on a VAX 8600 computer. The total execution times for grouping the various patterns ranged from 1.36 seconds (for the pattern in Figure 1(a) with 24 line segments) to 72.2 seconds (for the pattern in Figure 4(b) with 865 line segments).

4. Conclusions

In this paper, we have presented an algorithm to detect significant perceptual structures in line patterns. The relevant properties considered by the algorithm are the locations and the orientations of the line segments. The neighbor relationship between line segments and the geometric structure in a local neighborhood is represented by the Voronoi tessellation (and the associated Delaunay graph) defined for line patterns.

The grouping algorithm is formulated as a probabilistic relaxation labeling scheme. This formulation has three main advantages: (i) The grouping algorithm can work in parallel using only the local information around each object to be labeled but enforcing more global constraints through the neighbor interactions; (ii) By assigning weighted labels to each Delaunay edge, the algorithm is able to represent ambiguous cases; and (iii) The use of rigid thresholds is minimized.
Figure 4. (a) An picture of an outdoor view of a building. (b) The line pattern extracted from the picture shown in (a). (c) The result produced by our algorithm.

Our algorithm falls somewhere between the purely local algorithms and purely global algorithms and does not suffer from their limitations. It contrasts with purely local algorithms such as border tracking because of its ability to have interaction between tokens that may be many pixels apart. On the other hand, interactions between tokens are restricted to those between Delaunay neighbors. It is, therefore, also different from purely global algorithms such as the Hough transform. We have tried both Hough transform and border tracking on our data, and the results produced by our algorithm appear to be better.

Currently, the relative perceptual validity of the various groupings is not taken into account. A measure of the relative goodness of groupings needs to be incorporated into our implementation. Our current implementation assigns a unique label to each Delaunay edge by selecting the label with the largest probability. The capability of assigning more than one label to an edge needs to be incorporated into the algorithm in order to better handle ambiguous cases. This modification combined with the measure of goodness may give us a set of possible groupings each having a weight assigned to it.

Our algorithm is currently restricted to processing input patterns which consist of dots and straight lines. It is unable to group patterns that contain curves or tokens with complex shapes. Patterns obtained from real images often contain such complex tokens. Thus, in order to cope with real images successfully, the ability to process input patterns containing complex tokens must be incorporated into the grouping algorithm. We note that our current method of computing the Voronoi tessellation is sufficiently general to be able to handle the more complex shapes, but the rest of the grouping algorithm needs to be modified. Another area of
improvement is to extend our algorithm to detect any hierarchical structures present in the input pattern. Our grouping algorithm can be recursively applied to obtain a more complete description of the perceptual structure in the input pattern. We are currently working on these problems.

References


