CHAPTER 14 – PREPROCESSING

14.1 NEIGHBORHOODS

Extraction of feature points from a data set is based on examination of the local variation of the surface in the vicinity of the point. Each point in the data set is analyzed. Since the criteria are based on the behavior of the surface in the area near the given point, a neighborhood is established for each point.

A neighborhood of a point consists of data points from the original data set which are “near” the given point. Since the data points are stored in an array, an array of subscripts of neighbors is established for each point as a pre-processing step.

14.1.1 Establishing Neighborhoods

There are various definitions that could be used for the term “near” in the above definition of neighborhood. For example, all points within a given radius might be considered neighbors. Another approach would be to include the nearest n data points for some carefully chosen value of n.

An ideal definition of neighborhood of a point for the purpose of feature point extraction would be that a neighborhood includes only those data points which describe the surface at the given point but includes all points necessary for measurement of the variation of the surface around the given point.
By extension, the definition applied for the purpose of using these feature points would also include the notion that neighborhoods should be chosen so that feature points are not labeled if other points “near” a feature point describe that same geometric or topological structure so that the additional feature points are not necessary for surface reconstruction.

Neighborhoods should be as small as possible to minimize processing but should be large enough that features are not missed. Additionally, if neighborhoods are too large, small features are more easily missed in the processing phase.

The ideal “size” (measured either in distance or number of points) varies throughout the data set for at least three reasons: geometric and topological features vary in size, the density of the data sampled from the surface may vary, and the amount of noise in the data set may vary.

Stated in other words, ideal neighborhoods would include enough points to easily distinguish variations in surface structure but be no larger.

Actually, it is desirable to have two neighborhoods for each point; or the neighborhood could be easily partitioned into two sets. One set of neighbors would describe the surface at the given point including proper information to determine the amount of error present and properly smooth the data point. The other set would describe the changes in the surface occurring near the point. Even better would be the ability to partition the second set of neighbors into sets that would describe all independent surface features near the given point while retaining information necessary to measure interaction between these features.
If the neighborhoods allowed a given point to be labeled not only as a feature point but also as a feature point of a particular type associated with a set of other nearby feature points of the same type, the edges generated by connecting feature points could contain much more information.

In order to construct ideal neighborhoods of any type described above for each data point, the surface at that point would need to be known.

Since the purpose of the research is to reconstruct an unknown surface by approximation from data points, the construction of ideal neighborhoods is impossible. Therefore, the first approximation made in the process of surface reconstruction is the construction of an approximate neighborhood that contains only necessary points.

When this work was originally begun, neighbors for a given point were determined by choosing the closest N points. For some data sets, this approach did not work well since an N value that included enough data points for some parts of the surface would include many unrelated points for other areas of the surface. An additional potential problem arises if the point density near a point is non-uniform in such a way that there are N points clustered in one direction near the data point, but additional points farther away are still near enough to contribute to the understanding of the surface at the point.

The approach was changed to include all points within a given radius. For some data sets, in order to have enough neighbors for all points, some points have too many neighbors.

A combination approach was also used which, unfortunately, reduces efficiency. All points within a given radius are put into a neighborhood. There is also a maximum number of points allowed in a neighborhood. Therefore, if a neighborhood includes too many points, the radius is
reduced by a factor, typically 0.9, and points are removed. This process is repeated until the number of points in the neighborhood is less than or equal to the maximum allowed. This approach avoids the necessity of sorting the points by distance and usually removes enough points in 1 or 2 iterations.

An alternative possibility would be to choose the N closest points and remove those outside a given radius. The “removal” of points would be more efficient in that the distance is calculated in the process of finding the nearest points. However, the approach of finding the N closest points with no limit on radius is not able to use simple pre-processing techniques based on distance.

An additional complication is that some of the steps used later in surface reconstruction assume that if point A is in the neighborhood of point B, then point B is in the neighborhood of point A. Therefore, if point B should be removed from the neighborhood of point A because A has too many neighbors, A must also be removed from the neighborhood of point B. Alternatively, if A is needed in the neighborhood of B, then B is also included in the neighborhood of A.

Due to point removal, it is possible that some neighborhoods will include points that are farther away than some points that have been removed from the neighborhood.

This same inclusion symmetry property effects neighborhoods that are chosen based solely on number of neighbors. It is possible that point A is among the N closest points to B but B is not among the N closest points to A.

The data structure that is used for the neighborhood is an array indexed by point number. The information for the data point as well as the neighborhood, except for the actual point coordinates, is stored here. The data fields include a pointer to a variable size array of neighbors
(subscripts into the point array), the number of neighbors, an array for storing the normal for the data point, the feature point flag, a type field to indicate the type of feature point and an integer field that is used for various purposes.

14.1.2 Efficiency Considerations

Without pre-processing, finding the m closest neighbors to point in a set of N data points, is a $\Theta(N \log N)$ process. For all N points, the worse case time is $\Theta(N^2 \log N)$. For the alternate approach, finding all points within a given radius of a point from a set of N points without pre-processing requires $O(N)$ time. Therefore, finding all points within a given radius for each point requires $\Theta(N^2)$. For the large data sets often found in surface reconstruction problems, this approach is still too time consuming. Therefore, some preprocessing is used to restrict the search for neighbors to points that are known to be “near” the point being processed.

In order to group points, the minimum and maximum values of x and y in the data set are found as the data is input. A number of intervals in each dimension, k, is chosen and a k x k 2D grid is built which contains subscripts of points falling in the given interval. Each grid element also includes subscripts of points that are within the neighborhood radius of the grid element. As each data point is processed, its grid element is calculated and the points in that grid are tested. The preprocessing time has a complexity of $\Omega(N + k^2 N)$ where N operations are necessary to place the points in the proper grid and N operations are required for each of the $k^2$ grid elements to find points within the given radius of the grid boundary. In k is relatively small compared to the size of the data set, the pre-processing becomes $\Omega(N)$. 
In the implementation of the algorithm, the data was sorted before input by z value, then y value, and x value. This ordering resulted in fewer operations to place points within the radius of a grid in the grid. Another approach, that was not tested, would be to place all points into their proper grid element and then search only the adjacent grid elements for additional points within the radius of the grid boundary. This approach would produce correct results as long as the grids were larger than the radius.

The parallel implementation used a similar approach where the data was allocated to processors based on z value and grids were built on each processor for the local data. In this instance, points within the given radius in z were added to the grid.

A slightly different approach was also used where the grid was based on units which were twice the neighborhood radius in both x and y.

14.2 NORMAL VECTORS

Normal vectors are defined in a statistical sense for each data point. As mentioned in the overview section, many surface reconstruction methods rely heavily on the proper calculation of the surface normal at a point. In the feature point approach the normal vectors are not used directly for feature point detection but are used to establish a tangent plane to the neighborhood at the given point.

Since the neighborhood consists of all points within a given radius, R, of the point in question, the neighborhood can be viewed as a sphere of radius R centered at the point. Later in the processing, the points in the neighborhood will be projected to the tangent plane. When a plane,
such as the tangent plane, intersects a sphere and passes through the center of the sphere, the result is a circle with the same radius as the sphere. Thus, the projected neighborhood can be viewed as a 2D circle of radius \( R \) centered at the point and contained within the neighborhood sphere.

The method used to calculate the normal vector is similar to that used by other surface reconstruction approaches [55] where the orientation of the tangent plane is found from the normal vector and the center of the tangent plane is the centroid of the neighborhood. In these methods, surface reconstruction is based on a signed distance function found from the collection of tangent planes.

In the feature point approach, the data point is taken as the center of the neighborhood. The detection of feature points is dependent on the variance of data points in the neighborhood about the actual data point rather than the centroid of the neighborhood.
The normal is calculated using principle component analysis so that the tangent plane is the least squares best fit plane for the neighborhood. The covariance matrix for the neighborhood is the
expectation of the outer product of the difference vectors for each point about the center of the neighborhood (the data point under consideration.)

For data point $A_i$ with $k$ points $B_1, \ldots, B_k$ belonging to the neighborhood of $A_i$, the covariance matrix is the symmetric, positive, semi-definite 3x3 matrix given by

$$
\sum_{j=1}^{k} (B_j - A_i) \otimes (B_j - A_i)
$$

where $(B_j - A_i)$ is a 3D vector. Element $mn$ of the matrix is the sum over all points in the neighborhood of $(B_j - A_i)_m$ multiplied by $(B_j - A_i)_n$ where $m$ and $n$ take on the values 1, 2, and 3 representing $x$, $y$, and $z$ components of the difference vector.

This matrix has three eigenvalues $\lambda_1, \lambda_2, \lambda_3$ associated with eigenvectors $v_1, v_2, v_3$. For $\lambda_i$ the minimum eigenvalue, $v_i$ or $-v_i$ is the normal vector.

The choice of the calculated vector or its antiparallel is not important since the neighborhoods are treated independently in the feature point approach. Other techniques rely on a consistent orientation and must use an approximation [55] to make proper orientation choices since the general problem of enforcing a consistent orientation has been shown to be NP-complete via a reduction to MAXCUT [86].

The eigenvalues and eigenvectors have been calculated using two approaches with the same results. One approach is to treat the equation for the eigenvalues as a cubic equation and solve it using the standard approach to solving a cubic equation. The other approach is to use Jacobi [87] iterative factorization to find the eigenvalues and corresponding eigenvectors.