JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 34, No. 4, July–August 2011

Automating the Process of Airport Surface Node-Link Model Generation

Hak-Tae Lee*
University of California, Santa Cruz, Moffett Field, California 94035
and
Thomas F. Romer†
NASA Ames Research Center, Moffett Field, California 94035

DOI: 10.2514/1.49184

Currently, few detailed airport surface models for air transportation system simulation and analysis exist, because the models are complex to develop and maintain. An improved method of generating detailed models is needed to expand the set of detailed airport surface and terminal airspace models. In this paper, an efficient means to automate the procedures to develop airport surface node-link graph models is proposed. Node-link graphs were created by connecting the centroids of the polygons from geographic information system data. A principal axis-based algorithm was formulated to identify and divide complex polygons. The graph was further improved by adjusting node positions, using the known centerlines from rectangular polygons and removing redundant nodes and links. The procedures were tested on 79 United States airports and proved to be robust, accurate, and efficient.

Nomenclature

A	=	area of polygon, m^2
AR	=	effective aspect ratio
$A_{\min}$	=	area of smallest polygon, m^2
$\mathbf{C}$	=	connectivity matrix for runway
c_{ij}	=	element of $\mathbf{C}$
h	=	height of rectangle, m
$\mathbf{I}$	=	moment-of-inertia tensor, m^4
I_{xx}, I_{yy}, I_{xy}	=	two-dimensional area-based moments of inertia, m^4
I_{11}, I_{22}	=	principal moments of inertia, m^4
k	=	longitude scale factor for Lambert conformal projection
Ni	=	node with index i
Pi	=	polygon with index i
R	=	latitude scale factor for Lambert conformal projection, m
$\mathbf{r}$	=	position vector, m
R_0	=	Earth radius, m
S_{ij}	=	two-dimensional cross product of $\mathbf{r}_i$ and $\mathbf{r}_j$, m^2
s_{ij}	=	element of $\mathbf{C}^2$
v_x, v_y	=	x and y components of eigenvector
w	=	width of rectangle, m
X, Y	=	centroid-centered coordinates, m
x, y	=	planar coordinates, m
x_{cen}, y_{cen}	=	coordinate of centroid, m
x_0, y_0	=	reference position for Lambert conformal projection, m
ϵ_A	=	threshold for minimum area, m^2
ϵ_{AR}	=	threshold for maximum effective aspect ratio
ϵ_b	=	threshold for point and line segment relation, m
ϵ_{b0}	=	reference threshold, m
ϵ_c	=	threshold for runway entry/exit consolidation, m
ϵ_g	=	threshold for graph simplification, m
ϵ_L	=	threshold for minimum shared border length, m
ϵ_r	=	threshold for rectangle identification, m
ϵ_s	=	threshold for polygon simplification, m
λ	=	latitude, rad
λ_0, λ_1, λ_2	=	reference latitudes for Lambert conformal projection, rad
τ	=	longitude, rad
τ_0	=	reference longitude for Lambert conformal projection, rad
ψ	=	complementary latitude, rad

Presented as Paper 2008-7101 at the AIAA Modeling and Simulation Technologies Conference and Exhibit, Honolulu, HI, 18–21 August 2008; received 2 February 2010; revision received 10 December 2010; accepted for publication 31 January 2011.

*Associate Scientist, Mail Stop 210-8. Member AIAA.
†Aerospace Engineer, Wind Tunnel Operations Branch, Mail Stop 210-15. Member AIAA.

I. Introduction

SIMULATIONS of the air transportation system with detailed models of terminal areas and surfaces are often necessary to assess future concepts for managing air traffic. These simulations will include studies of operations at individual airports [1], as well as wider regions covering many airports [2]. All of them will require detailed surface and terminal airspace models capable of representing both current and future operating configurations. Because no two airports are identical, proper analysis of a concept's capabilities and benefits will require modeling operations at individual airports.

To simulate the movements of aircraft on the airport surface, it is convenient to use node-link graph models, since the movements are constrained by taxiways and runways. Modeling aircraft movements on a node-link model makes it straightforward to determine separation between aircraft. In addition, various depths of details can be modeled depending on the nature of the simulation [2], and the models can be adapted for existing simulation tools, such as the Airport and Airspace Delay Simulation Model [3] used by the Federal Aviation Administration (FAA) or the Total Airport and Airspace Model [4]. Another distinctive benefit of using the node-link graph structure is that the exact locations of nodes and shapes of links are not critical, as long as the connectivity is correct and the lengths of the links are close to the actual runway and taxiway lengths.

Developing, maintaining, and modifying even a small number of detailed airport surface node-link models are significant tasks. Most existing detailed surface models are created manually by extracting the coordinates of nodes from data sources, such as aerial photography, and connecting adjacent nodes with links. Reference [5] describes in detail such a manual model generation procedure for a surface simulation and a decision support tool [1].

One of the few efforts to automate the surface node-link model generation process is found in [6]. In this work, a feature from geographic information system data [5,7] that provides the actual taxiway centerlines is used to generate node-link models. Although this approach can result in accurate node-link graphs, the taxiway centerline feature is not available for many airports, thus limiting the number of airports that can be quickly modeled.

A more general problem of extracting centerlines from digitized vector map data or aerial photography has been actively researched in the cartography and remote sensing fields. Voronoi-diagram-based methods are widely used, because the edges of a Voronoi diagram represent local centerlines. These methods are particularly effective for extracting centerlines from river networks [8,9], because the shape of riverbank boundaries is generally irregular. Reference [8] used a Voronoi diagram to find a medial axis and compared several approximation models for the medial axis. Reference [9] directly applied Voronoi diagram to a river network. The authors proposed merging and densifying the river contours to create a large number of short segments that represented the centerlines and to automatically identify junctions. Reference [10] proposed a centerline extraction method based on triangulating the road contour, using rightangle triangles instead of Delaunay triangles used by the Voronoi-diagram-based methods. This method is effective for curved roads. Reference [10] also proposed a separate algorithm for identifying intersections. Some research focused on extracting the road centerlines from pixilated imagery. Reference [11] used a template window that translates and rotates along the road to find the best matching target window by least-squares correlation matching. Reference [12] also employed a moving window but used a sequential similarity detection algorithm to find the optimum target window. These methods are effective when the vector map is not available. However, these methods cannot automatically identify road intersections.

In the present study, a comprehensive set of procedures that automatically generates node-link graphs for airport surfaces from geographic information system vector data is developed. The basic approach is to use the centroid lines from [8] as approximations for centerlines. However, instead of using Delaunay triangulation, the polygons in the data set are divided only when necessary. The division criteria and processes are based on moments of inertia and a principal axis. It will be shown in the following sections that the principal axis-based division is more efficient and robust than the Voronoi-diagram-based methods [8,9] in creating airport surface node links. A separate procedure for treating the runway data is presented, which identifies the runway centerlines and places nodes on runway junctions as well as on entry and exit points. The node-link graph model is further improved by adjusting node locations using the known centerlines from rectangular polygons and then by removing redundant nodes and links using a line simplification algorithm.

The resulting node-link graph models efficiently capture the layout and connectivity of runways and taxiways. They are tested on all 79 airports from the data set. Node-links graphs for 76 airports are automatically created without manual intervention.

The structure of this paper is as follows. In Sec. II, source data are described. Section III explains the step-by-step procedures taken to automatically generate the surface node-link graph. Section IV shows the automatically generated node-link graphs of selected airports to demonstrate the effectiveness of the procedures developed in this study and discusses issues and concerns. Section V concludes the paper.

II. Airport Surface Information

Safe Flight 21 (SF21) is a program sponsored by the FAA to explore and demonstrate the use of Automatic Dependent Surveillance-Broadcast and other related technologies for use in the free flight concept of operations [7].[‡] As a part of the SF21 project, the FAA created digitized airport maps by processing aerial photography for about 80 airports to facilitate airport surface automation.

Among several available digitized map data, the SF21 data are chosen for the present study. mainly because many of the existing surface simulation and decision support tools, including the surface management system [1], are based on this data, and the entire data set is readily available for the study.

SF21 data are a collection of polygons or lines categorized by different fields, such as runway, taxiway segment, taxiway guidance line, and other fields. Each polygon is represented by a list of latitude and longitude pairs for the vertices that define the polygon. Figure 1 shows the runway and taxiway polygon representation of the San Francisco International Airport (SFO).

As can be seen from Fig. 1, taxiways are divided into a number of small polygons in the SF21 data. Two notable characteristics can be found among these polygons. One characteristic is that all the polygons are simply connected, and the other is that many of the polygons are rectangles. Table 1 lists the polygonal statistics of taxiway polygons for four airports: Dallas–Fort Worth International Airport (DFW), Chicago O'Hare International Airport (ORD), SFO, and Hartsfield–Jackson Atlanta International Airport (ATL). Table 1 shows that the average area is on the order of 4000 m^2, and the number of rectangles can vary. In the following sections, the graph generation processes that exploit these characteristics will be presented.

III. Procedures for Automated Node-Link Model Generation

Procedures for automated node-link model generation are described throughout the following subsections. SF21 data are projected to a planar coordinate system using Lambert conformal projection. The formulas and reference values used for the projection are listed in the Appendix A. First, these polygons are simplified to improve runtime performance. Then, the connectivity between the taxiway polygons is searched, and runways are identified. Next, runway entry and exit nodes and links are created. Finally, the graph is adjusted to improve node-link placements and simplified to remove redundant nodes and links.

The core of the automated node-link generation process is finding connectivity between polygons. Two polygons are considered to be connected if the polygons share a series of line segments for which the total length is longer than a minimum length ϵ_L. These shared lines will be referred to as shared borders in the following sections. If the two polygons are connected, a node is created at the centroid of each polygon, and the two nodes are connected by a link, as depicted

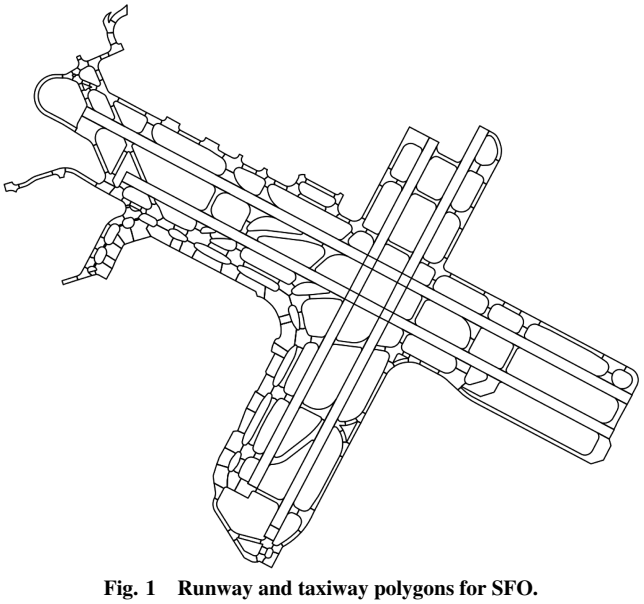


Fig. 1 Runway and taxiway polygons for SFO.

[‡]Data available at http://hf.tc.faa.gov/projects/safeflight21.htm [retrieved 22 Jan. 2008].

Table 1 Statistics for selected airports

Properties	Airports			
	DFW	ORD	SFO	ATL
Number of polygons	733	347	198	257
Number of rectangles	297	17	18	0
Average area, m^2	3836	4453	4642	4860

in Fig. 2. Since ϵ_L can be considered a minimum taxiway width, 5 m (a width that a ground vehicle can pass through) is assigned for ϵ_L.

Shared borders, which determine the polygonal connectivity, are identified by testing all of the edges of one polygon against the edges of another polygon. If any two edges share a line segment, it becomes a part of the shared border. When testing a pair of edges, the endpoints of one edge are checked to determine if they lie on the other edge, and vice versa. A bounding box is constructed around the given segment, as shown in Fig. 3. The box extends the segment by ϵ_b in the direction perpendicular and parallel to the edge. If a point is inside this box, it is considered to be on this segment. In Fig. 3, one point, **r**, is considered to be on this segment, while the other point, **r**′, is not.

Since the data are not perfect, it is important to find a reasonable value of ϵ_b. If ϵ_b is too small, some of the connected polygons will not be detected. On the contrary, if ϵ_b is too large, links will be created between some of the closely spaced polygons, even though they are not actually connected. So, ϵ_b should be small enough that the bounding box should not fully cover the smallest polygon. The initial

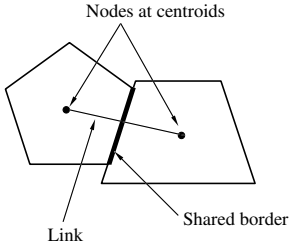


Fig. 2 Diagram showing two connected polygons with a shared border.

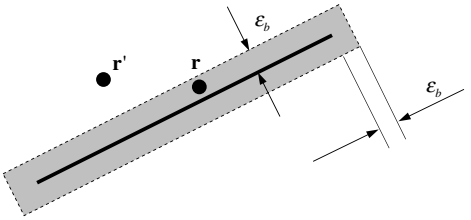
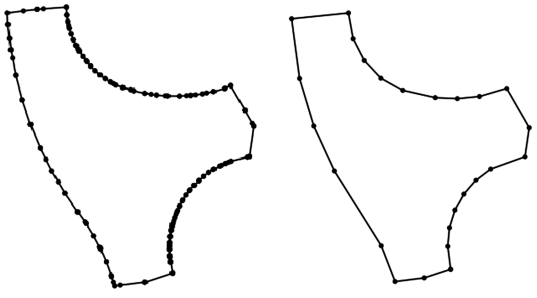


Fig. 3 Criteria for determining whether a point lies on a segment.



a) Original polygon **b) Simplified**

Fig. 4 Example of polygon simplification through the Ramer–Douglas–Peucker algorithm.

Table 2 Result of polygon simplification with varying threshold

Threshold, ϵ_s, m	Average number of vertices per polygon			
	DFW	ORD	SFO	ATL
0 (Original)	14.9	24.8	40.6	128.0
0.5	11.0	15.1	14.7	15.3
1	8.7	12.0	11.5	11.9
2	7.6	9.6	9.2	9.7

value for this threshold ϵ_{b0} is set as one-10th of the square root of the minimum area $A_{\min}$, as shown in Eq. (1). This value is one order of magnitude smaller than the characteristic edge length of the smallest polygon in an airport:

$$\epsilon_{b0} = \frac{\sqrt{A_{\min}}}{10} \tag{1}$$

A. Polygon Simplification

Some airports, such as ATL, have polygons with unnecessarily large numbers of vertices, which can significantly slow down the computation. The polygons are simplified using a Ramer–Douglas–Peucker algorithm [13,14]. Figure 4a shows a polygon from ATL that has 245 vertices. Figure 4b shows the result of the Ramer–Douglas–Peucker simplification with a threshold ϵ_s of 1 m. The number of vertices is reduced to 25.

To determine the acceptable level of ϵ_s, average vertex counts per polygon for four different airports with varying ϵ_s are shown in Table 2. The threshold values, ranging from 0.5 to 2 m, result in about 10 vertices per polygon on average. Based on these results, 1 m is used.

B. Creating Node Links for Taxiways

The basic node-link structure of an airport is constructed by connecting the centroids of adjacent taxiway polygons. As mentioned in Sec. II, since the data consist of relatively small simply connected polygons, original polygons are used instead of the Delaunay triangulation proposed in [8]. Figure 5 shows two examples of taxiway node links. Connectivity is correctly captured, and the links are reasonable approximations to the centerlines. Moreover, the links are much longer and straighter than those created by the Voronoi-diagram-based methods, avoiding the jaggedness shown in [8].

Some of the polygons are long, narrow, and often concave. In such cases, the centroid lines significantly deviate from the centerlines, as shown in Fig. 6. It is necessary to identify and divide those polygons. A concept of effective aspect ratio and principal axis division process is proposed, which is simpler and more efficient than the Delaunay triangulation in [8,9].

Before introducing the polygon division algorithm, several terms are defined. An effective aspect ratio is defined as the aspect ratio of

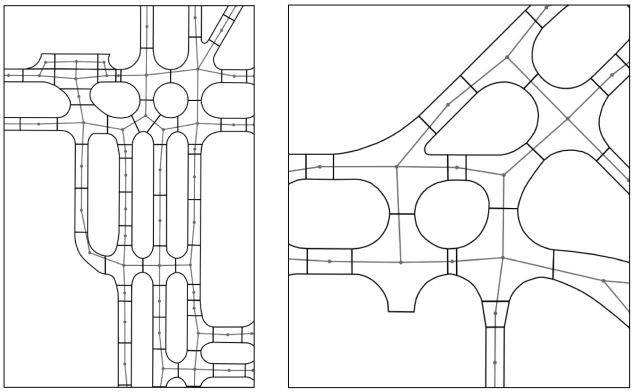


Fig. 5 Examples of taxiway node links created by connecting centroids.

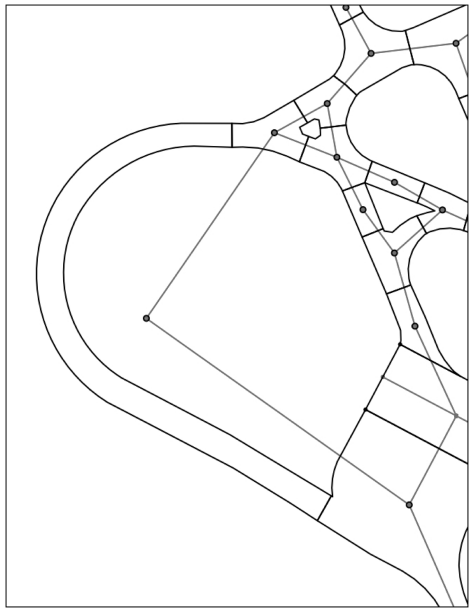


Fig. 6 Examples of a polygon that requires division and a node-link graph that significantly deviates from the centerlines.

an effective rectangle. An effective rectangle of a polygon is a rectangle that shares a centroid, principal axes, and moments of inertia with the polygon. A principal axis division is defined as dividing a polygon along the principal axis around which the moment of inertia is the largest. This division generally results in polygons with smaller effective aspect ratios compared with the original polygon. The interior condition is defined such that it is satisfied when the centroid is inside the given polygon. Detailed mathematical procedures of computing moments of inertia and principal axes are in Appendix B.

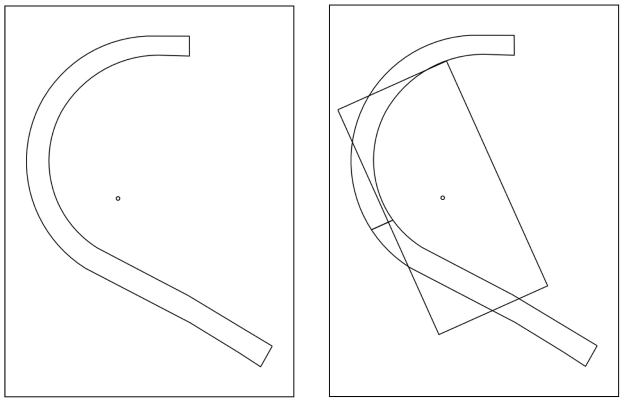
The polygon division algorithm recursively applies principal axis division. There are three stopping criteria, with the first criterion being a rectangle. Since a rectangle has a well-defined centerline, it is not necessary to divide a rectangle. The second criteria are the effective aspect ratio and interior condition. If the effective aspect ratio is smaller than the maximum aspect ratio and the interior condition is satisfied for a polygon, this polygon is not divided further. The third criterion is minimum area. If the area of a polygon is smaller than the minimum area, the division process stops for this polygon.

Figure 7 describes the division process with an example. Figure 7a shows a J-shaped polygon that does not meet the interior condition. Figure 7b shows it divided into two parts, along with its effective rectangle. Figure 7c shows the upper portion being divided because it violates the interior condition, and it shows the lower part being divided because its effective aspect ratio is too large. Figure 7d shows the original polygon divided into four polygons. This process can continue according to the stopping criteria and thresholds.

Three threshold values are involved in the stopping criteria. The rectangle is identified if all the vertices of a polygon are within a bounding region of its effective rectangle, as shown in Fig. 8. The bounding region is the banded area of width $2\epsilon_r$ around the effective rectangle. The value ϵ_{b0} from Eq. (1) is used for ϵ_r.

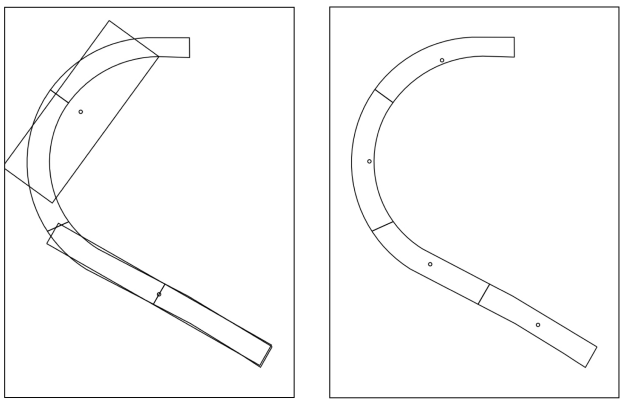
To prevent the division algorithm from generating extremely small polygons, a fifth percentile of the taxiway polygon area is used for the minimum area ϵ_A. This approach of using a percentile value ensures that the minimum area is small enough but not exceptionally small.

The maximum aspect ratio is determined based on a simple arc length analysis. Figure 9 shows the error if the arc length of a half-circle is approximated by a number of chord lines of equal lengths. If the half-circle is approximated by four chords, the error is about 2.5%. A typical radius of a section of a taxiway that forms a half-circle is around 100 m. If 20 m is assumed for the taxiway width,



a) J-shaped polygon and its centroid

b) Divided into two polygons effective rectangle overlaid



c) Divided again

d) Divided into four polygons

Fig. 7 Polygon division process.

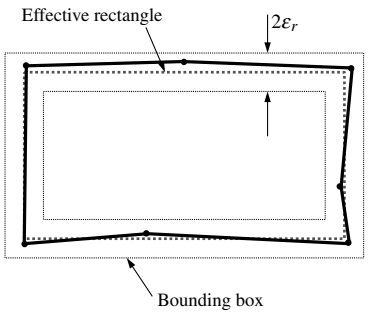


Fig. 8 How to determine whether a polygon is a rectangle.

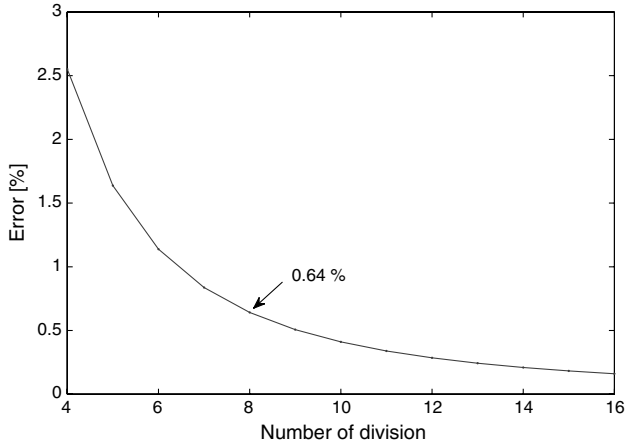
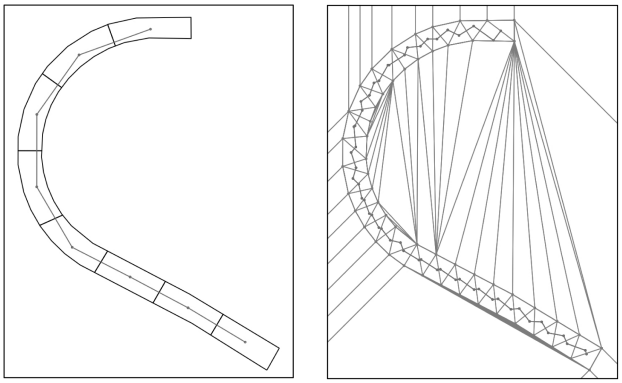


Fig. 9 Error in arc length with respect to the number of chords used for approximation.



a) Principal axis division b) Voronoi diagram

Fig. 10 Principal axis-based division compared with Delaunay triangulation.

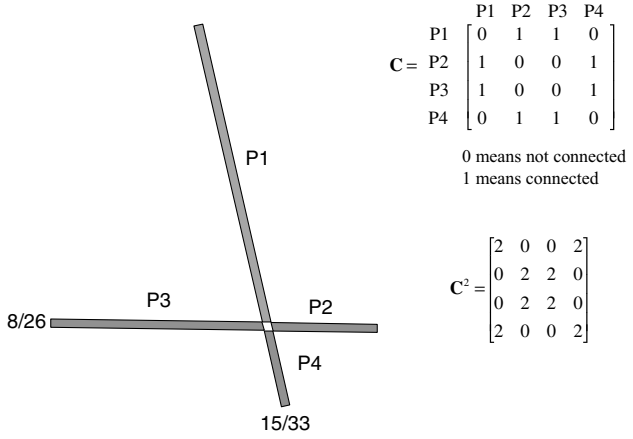


Fig. 11 Example of crossing runways at BUR.

one-quarter of this half-circle will have an arc length of 78 m. The aspect ratio is approximately four, which is assigned as the maximum aspect ratio ϵ_{AR}. Nonrectangular polygons with effective aspect ratios greater than four will be divided.

Figure 10 compares the centerlines created for a J-shaped polygon. The centerline in Fig. 10a is created by the principal axis-based division algorithm. Figure 10b shows the centerline from the Voronoi diagram along with the Delaunay triangulation. As recommended in [9], the polygon is densified with extra vertices spaced about 20 m apart. As can be seen in Fig. 10b, the Voronoi diagram centerline is notably jagged. This jaggedness depends on the alignment of the densified vertices, and it is not straightforward how to control this alignment.

With the polygon division algorithm in place with the previously mentioned thresholds ϵ_A and ϵ_{AR}, along with ϵ_L, the effect of the threshold for determining the shared border ϵ_b is examined. Table 3 shows the number of taxiway links when ϵ_b is varied for SFO. The reference value of Eq. (1) indeed captures the correct number of taxiway links, which is 276. Table 3 shows that Eq. (1) is adequate and that the process is not overly sensitive to the value of ϵ_b.

C. Identifying and Grouping Runways

The source data represent runways using single or multiple polygons. If a runway does not have any crossing runways, it is represented by a single rectangular polygon. However, if there are crossing runways, two or more polygons are used to represent a single runway, and these polygons are not connected. A special treatment is required to automatically identify how these polygons are connected and grouped.

Figure 11 shows the runway portion of the input data for Bob Hope Airport (BUR) in Burbank, California. Polygons P1 and P4 represent runway 15/33, and polygons P2 and P3 represent runway 8/26. The nonshaded parallelogram at the junction of the two runways is not represented as a polygon in the original data. To identify each runway, P1 and P4 must be grouped to one runway, and P2 and P3 must be grouped to another runway.

A runway-grouping algorithm is proposed, using the connectivity matrix between the runway polygons. The connectivity matrix $\mathbf{C}$ is constructed by the rule described in Eq. (2). Unlike the connectivity between taxiway polygons that requires a shared border, runway polygons are considered to be connected if they share a vertex. If two vertices from two different polygons are within the threshold distance ϵ_b, the two runway polygons are considered connected. For example, from the runway configuration shown in Fig. 11, polygon P1 is connected to polygons P2 and P3. The completed connectivity matrix for the runways in BUR is shown in Fig. 11:

$$c_{ij} = \begin{cases} 0 & \text{if } i = j \text{ or polygon P}i \text{ and P}j \text{ are not connected} \\ 1 & \text{if polygon P}i \text{ and P}j \text{ are connected} \end{cases} \quad (2)$$

Each element s_{ij} of the square of the connectivity matrix $\mathbf{C}^2$ indicates the number of polygons that are connected to both polygon Pi and polygon Pj. An element along the diagonal of $\mathbf{C}^2$, s_{ii}, represents the total number of polygons connected to polygon Pi. For example, if s_{ii} is zero, polygon Pi is a standalone runway. In the example runways shown in Fig. 11, s_{14} is equal to two, which means that two polygons are shared between polygons P1 and P4, and that polygons P1 and P4 form a runway. Similarly, polygons P2 and P3 form another runway.

For general runway configurations, such as in Fig. 12a, additional processing is required on top of calculating the $\mathbf{C}^2$ matrix. A temporary node-link graph is created, as shown in Fig. 12b. A node Ni is placed at the centroid of the runway polygon, Pi. Two nodes, Ni and Nj, are connected by a link only if s_{ij} is equal to two, which means there are two polygons that are connected to both polygon Pi and polygon Pj. This graph is used only for runway identification; thus, it is not related to the airport surface node-link graph.

Once the graph is constructed, the number of links connected to each node is checked. If a node has a rank higher than two, all the link pairs are examined to find the pair that forms a straight line. In the example shown in Fig. 12c, N4 has a rank of three. Among the three possible link pairs, one pair forms a straight line: namely, the link connecting N3 and N4 and the link connecting N4 and N5. After this pair is found, other links are removed. In the example, the link connecting N4 and N9 is removed. This procedure is repeated until none of the nodes has a rank greater than two, as shown in Fig. 12e.

The process separates the graph into a number of disjointed polylines. Finally, each polyline is replaced by a link that directly connects the two endpoints of the polyline, as shown in Fig. 12f. In the example, the initial graph (Fig. 12b) is separated into four links that are not connected to each other, as shown in Fig. 12f. Two polygons corresponding to the end nodes of each link are identified. A convex hull is created around the two polygons, as shown in Fig. 12g. Even though this convex hull is very close to a rectangle, due to projection errors and other errors in the original data, it is not exactly a rectangle in most cases. The effective rectangle of this convex hull is assigned as the rectangle that represents the runway. Using the runway rectangles, centerlines are identified and become

Table 3 Number of taxiway links for SFO with respect to ϵ_b

ϵ_b	Number of taxiway links
$\epsilon_{b0}/8$	275
$\epsilon_{b0}/4$	276
ϵ_{b0}	276
$2\epsilon_{b0}$	276
$4\epsilon_{b0}$	277
$8\epsilon_{b0}$	279

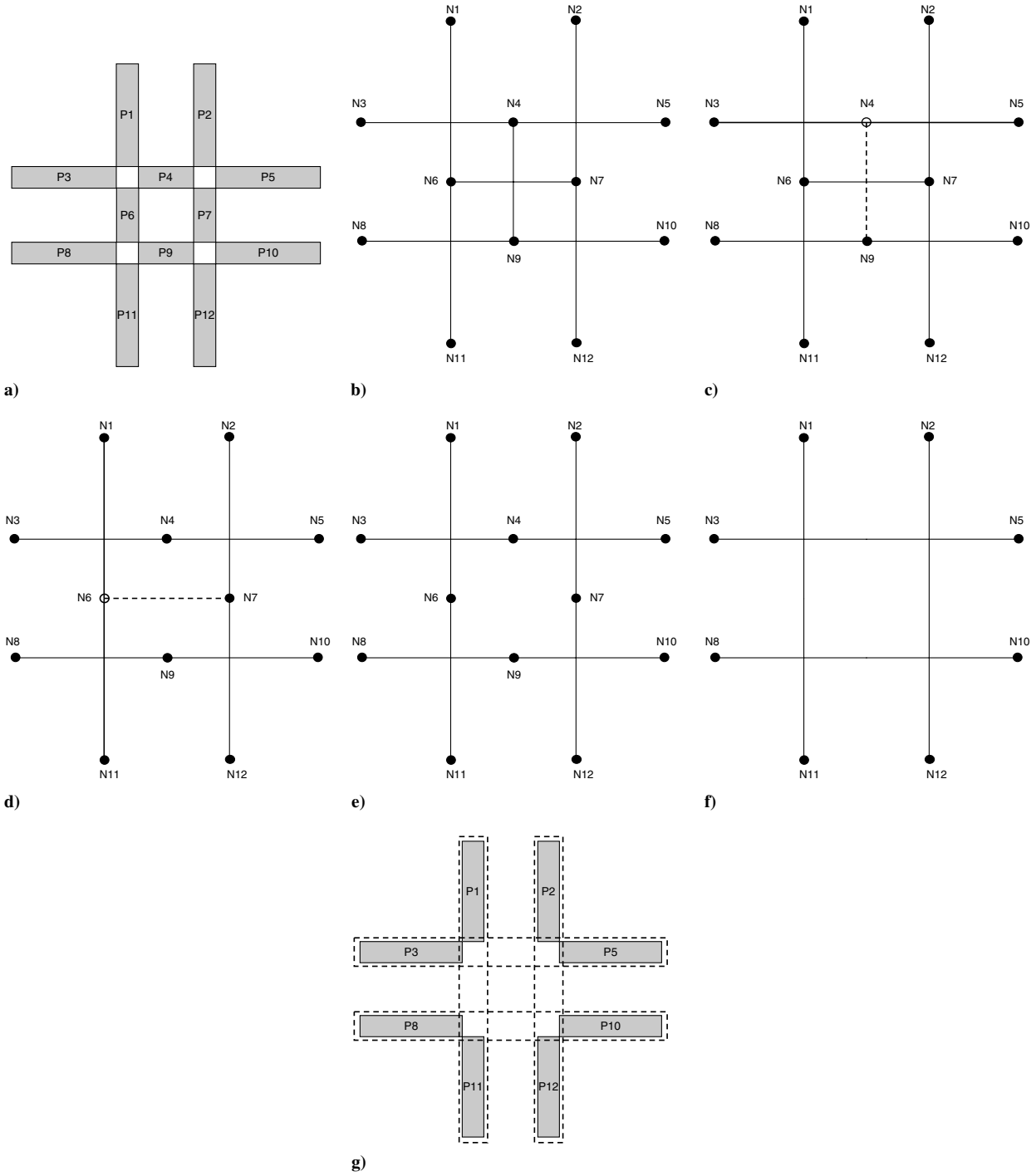


Fig. 12 Procedure of identifying each individual runway from a set of runway polygons.

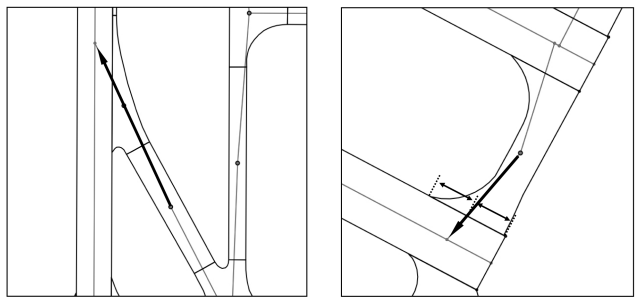
runway links. These links are divided by placing nodes at the runway crossings.

D. Connecting Runways and Taxiways

Once the basic runway node links are generated, they have to be connected to the previously generated taxiway node links. They are connected by placing runway entry and exit nodes along the runway links. First, a taxiway polygon that is connected to a runway polygon is identified using the same shard border method as in the taxiway node link with a higher threshold value, as in Eq. (3), because the runway polygons are approximated by effective rectangles:

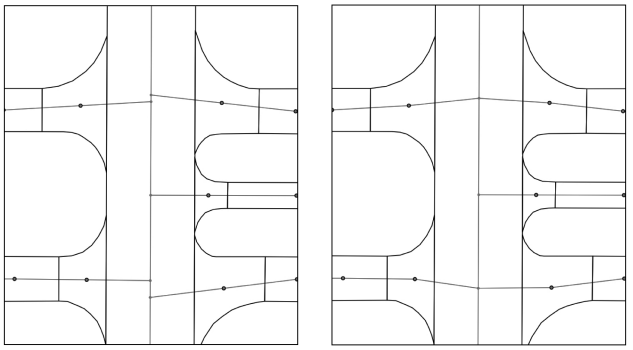
$$\epsilon_e = 5\epsilon_{b0} \tag{3}$$

Then, the link ending at this taxiway polygon is extended along a line that passes through the upstream node toward the runway



a) Primary procedure **b) Secondary procedure**

Fig. 13 Creating runway entry and exit links.



a) Initial linking between taxiway and runway

b) Runway entry/exit nodes are consolidated

Fig. 14 Consolidating nodes for runway entry and exit.

centerline, as shown in Fig. 13a. A node is placed at the intersection of the extension and centerline. This method, called the primary method, is desirable because it generally preserves the runway entry and exit angles. However, if only a single taxiway polygon is adjacent to a runway polygon, so that there is no line passing through the upstream node, or the link extension of the primary method does not cross the runway centerline, a secondary method is applied. Connection to the runway is made by creating a link from the centroid that goes through the midpoint of the shared border, as shown in Fig. 13b.

If a runway has entry and/or exit to both sides, as shown in Figs. 13b and 14a, two nodes are initially generated along the runway centerline: one for each entry or exit. These nodes should be consolidated to a single node, as shown in Fig. 14b. Node consolidation is performed by checking the distance between every pair of nodes along the runway centerline. If the distance is smaller than ϵ_c m, the two nodes are consolidated to a single node by taking the midpoint between the two. This process is repeated until there is no node pair with a distance smaller than ϵ_c. Fifty meters is assigned for ϵ_c, because the taxiway width is typically 20 m, and the spacing between consecutive entry and exit points is generally larger than 50 m.

E. Postprocessing

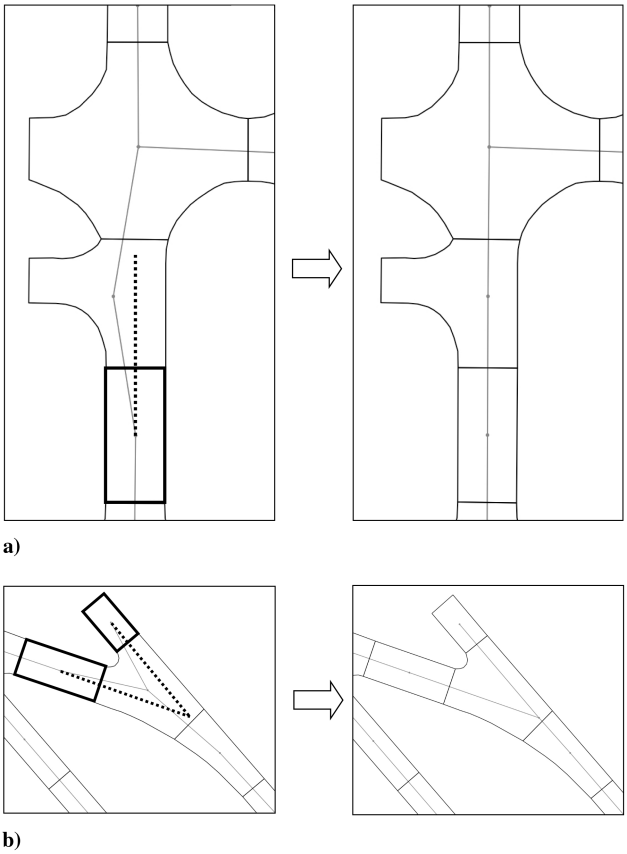
The node-link graphs created using the previously mentioned procedures capture the layout and connectivity of taxiways and runways. However, they can be further improved.

Nodes are relocated using the readily identifiable centerlines of the rectangular polygons. For every nonrectangular polygon, the number of connected rectangular polygons is counted. If no rectangle is connected to this nonrectangular polygon, the node remains at the centroid. If only one connected polygon is rectangular, the centerline of the rectangle is extended toward this polygon and the node is projected to this line, as shown in Fig. 15a. If two or more rectangles are connected, crossing angles between the extended centerlines are calculated for every possible pair, and the pair that forms an angle closest to 90 deg is selected. This crossing point becomes the new node location for the polygon, as shown in Fig. 15b.

After the node positions are adjusted using the rectangle-based correction, the graph is simplified by applying the same Ramer–Douglas–Peucker algorithm as the one used for polygon simplification to the node-link graph with a higher threshold value of $\epsilon_g = 5$ m. The value of ϵ_g depends on the tradeoff between the simplicity and accuracy. By simplification, redundant nodes are removed. Redundant nodes are nodes that do not represent junctions or changes in the directions in the node-link model (see Fig. 16).

IV. Results

The SF21 source data provide geometries of 79 airports. Node-link graphs are automatically created for 76 airports. Manual interventions are needed for the three remaining airports, which are discussed later in this section. Among those 76 airports, four representative ones [DFW, ORD, SFO, and the Eastern Iowa Airport (CID)] are presented in this section.

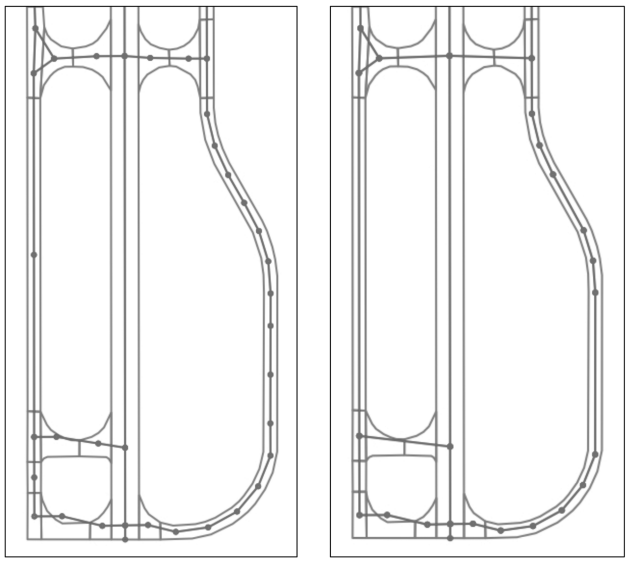


a)

b)

Fig. 15 Rectangle-based correction examples.

Figure 17 shows the result for DFW. Solid black dots and lines denote automatically generated nodes and links, while gray lines denote runway and taxiway polygons from the source data. DFW is chosen as the primary test case, because it has one of the most complicated taxiway networks among the U.S. airports, and many of NASA's new air traffic concepts are tested on DFW. While processing DFW, the rectangle-based correction algorithm is found to be effective, as more than 40% of the taxiway polygons are rectangles.



a) Before simplification

b) After simplification

Fig. 16 Graph simplification by removing redundant nodes.

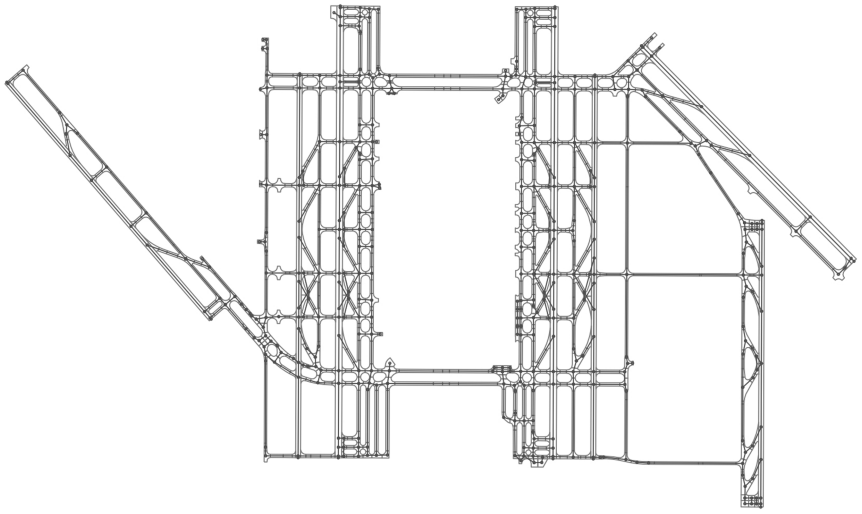


Fig. 17 Node-link graph of DFW.

Figure 18 shows the node-link graph for ORD. Contrary to DFW, which is relatively new and designed with generous space, ORD represents a relatively old and complex airport with limited space. While processing ORD, the principal axis-based polygon division algorithm is particularly effective, since many of the taxiway polygons are curved.

SFO is of medium complexity. It serves as the main test case for the runway-grouping algorithm, since it has two pairs of parallel runways crossing each other. Figure 19 shows the result for SFO; it indicates that all the runways are correctly recognized.

CID is one of the simplest airports in the data set. It is used to compare the node-link graph generated from this study in Fig. 20a to the one generated by the Voronoi-diagram-based method of [9] in Fig. 20b. As shown in the single polygon examples in Fig. 10, the links generated by the Voronoi diagram are much shorter. A total of 3548 links are created when the polygons are densified to have around 9 m vertex spacing. Figure 20c shows the simplified node-link graph using the Ramer–Douglas–Peucker algorithm. Even after simplification, some jagged links are not fully removed, with remaining 364 links, which is significantly more than the 52 links in Fig. 20a.

It takes 2.4 s on a Mac Pro desktop using a code written in Java 1.5 to create the graph in Fig. 20a. For the Voronoi-diagram-based method in Fig. 20c, it takes 29.8 s. Details about the computational environment are described in Appendix C. As described in [9], it is necessary to merge all the polygons to a single large set of vertices to create the Voronoi diagram. The merging process uses the same shared border identification process described in Sec. III, which is the most time-consuming component of the current principal axis-based method. In addition to the merging process, the Voronoi diagram has to be created and trimmed with a larger number of vertices due to densification, which further increases the processing time.

A complete list of results for 79 airports with the number of links and execution times is shown in Table C1 in Appendix C. Three airports that fail in full automation are Albuquerque International Sunport Airport (ABQ), San Antonio International Airport (SAT), and William P. Hobby Airport (HOU). ABQ and SAT fail because the runway identification algorithm cannot properly group the runway polygons. Figure 21a shows the runways of ABQ. An enlarged view of the circled area in Fig. 21b shows three runways crossing at a

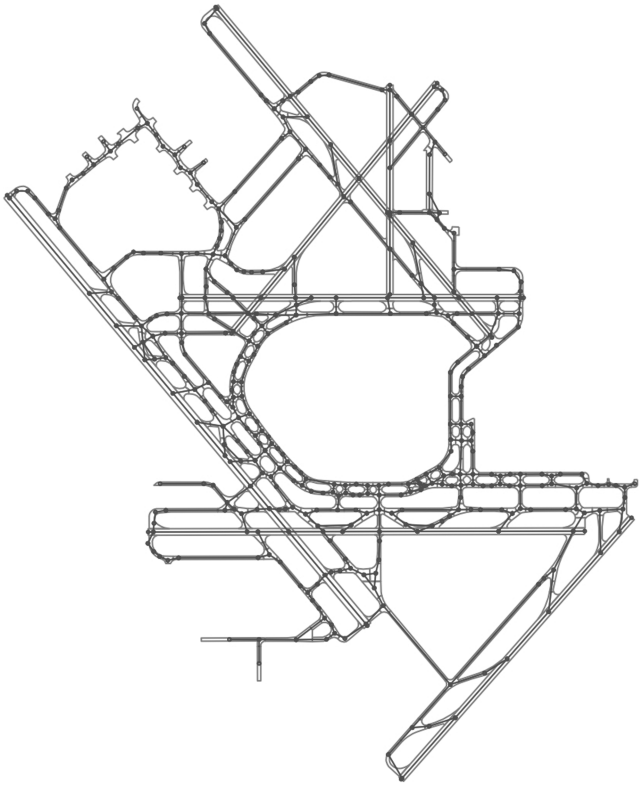


Fig. 18 Node-link graph of ORD.

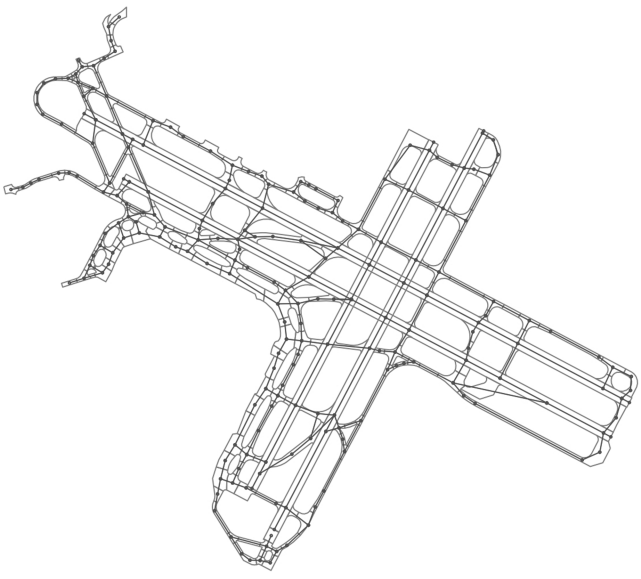
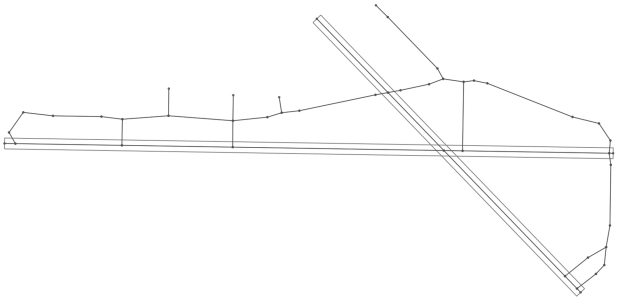
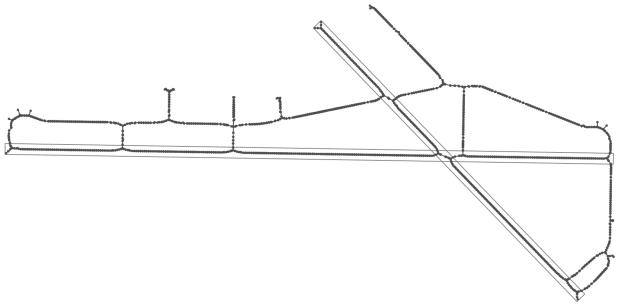


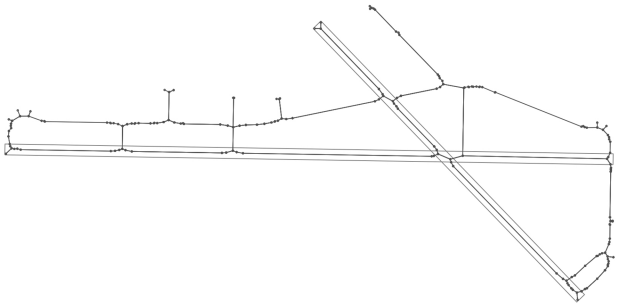
Fig. 19 Node-link graph of SFO.



a) Generated by procedures in current study



b) Generated by a Voronoi diagram-based method



c) Simplified using Ramer-Douglas-Peucker algorithm

Fig. 20 Node-link graphs of CID.

junction. Nine shaded polygons, labeled from P1 to P9, represent this runway intersection. For example, both P1 and P2 are part of runway 3/12. However, there is no polygon that commonly connected to both P1 and P2, which leads to the failure in the runway-grouping algorithm. SAT fails due to a T-shaped junction that has the same problem. HOU failed because the runway data are defective, having duplicated polygons. For all three cases, when the runways are grouped manually, node-link graphs are generated without any issue.

V. Conclusions

This study proposes a means to automatically generate airport surface node-link graph models. Several geometric algorithms are developed to manipulate the polygons of an airport layout provided by the SF21 geographic information systems data. Using these algorithms, node-link graphs of the airport surfaces are extracted by identifying polygon connectivity. Among the 79 airports of which airport layout data are provided, node-link graphs for 76 airports are automatically created. The resulting graphs, which accurately capture the connectivity and layouts of runways and taxiways, demonstrate the robustness and efficiency of the automated procedures.

Node-link graphs created from this study can be used as the foundation for creating node-link models of terminal areas. Since the automation can significantly reduce the time and effort required for generating models, it will enable simulations and analyses of many airports, as well as of a network interconnecting those airports.

Appendix A: Coordinate System and Projection

Lambert conformal projection is used to convert the latitudes and longitudes to plane coordinates. Each airport is separately projected using the reference latitudes and longitudes based on the maximum and minimum latitudes and longitudes of the taxiway polygons, as shown in Eq. (A1):

$$\lambda_0 = \lambda_{\min} \qquad \lambda_1 = \lambda_{\min} + \tfrac{1}{6}(\lambda_{\max} - \lambda_{\min})$$
$$\lambda_2 = \lambda_{\min} + \tfrac{5}{6}(\lambda_{\max} - \lambda_{\min}) \qquad \tau_0 = \tfrac{1}{2}(\tau_{\min} + \tau_{\max}) \tag{A1}$$

For subsequent formulation, complementary latitude ψ is used instead of λ:

$$\psi_i = \frac{\pi}{2} - \lambda_i \tag{A2}$$

The longitude scaling factor k is computed using Eq. (A3):

$$k = \frac{\ell_n(\sin\psi_1 / \sin\psi_2)}{\ell_n\{[\tan(\psi_1/2)]/[\tan(\psi_2/2)]\}} \tag{A3}$$

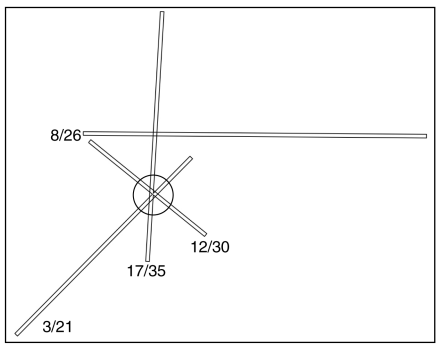
The latitude scaling factor R, which is a function of ψ, is computed using Eq. (A4), where R_0 is the average Earth radius of 6371 m:

$$R(\psi) = R_0 \frac{\sin\psi_1}{k} \frac{\tan^k(\psi/2)}{\tan^k(\psi_1/2)} \tag{A4}$$

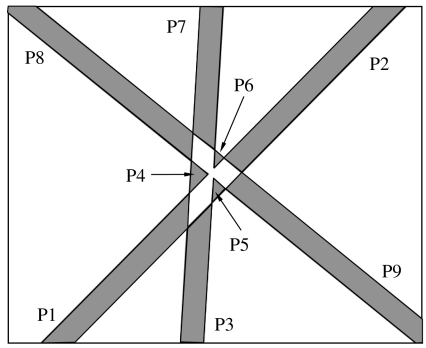
Finally, latitudes and longitudes are converted to plane coordinates using Eq. (A5):

$$x = x_0 + R(\psi)\sin[k(\tau - \tau_0)]$$
$$y = y_0 + R(\psi_0) - R(\psi)\cos[k(\tau - \tau_0)] \tag{A5}$$

Conversely, the plane coordinate can be converted back to latitudes and longitudes using Eq. (A6):



a) Runway layout



b) Enlarged view of the circled area

Fig. 21 Runways at ABQ: runways 3/21, 12/30, and 17/35 meet at a single point.

$$\tau = \tau_0 + \frac{1}{k}\tan^{-1}\left(\frac{x - x_0}{y_0 + R(\psi_0) - y}\right)$$

$$\psi = 2\tan^{-1}\left[\left(\frac{x - x_0}{R_0(\sin\psi_1/\{k[\tan^k(\psi_1/2)]\})\sin[k(\tau - \tau_0)]}\right)^{1/k}\right] \quad \text{(A6)}$$

Appendix B: Moment of Inertia and Principal Axis

The principal axes for each polygon are computed to find the effective aspect ratio and to divide polygons that have high effective aspect ratios. Area-based moments of inertia are computed to find the principal axes. First, each polygon should be translated such that the centroid becomes the origin, as described in Eq. (B1):

$$(X_i, Y_i) = (x_i - x_{\text{cen}}, y_i - y_{\text{cen}}) \quad \text{where } i = 1, 2, \ldots, N+1 \quad \text{(B1)}$$

After the translation, moments of inertia I_{xx}, I_{yy}, and I_{xy} are computed using Eqs. (B2–B4). Signs of moments of inertia are dependent on the direction of vertex numbering. If the vertices are numbered in a clockwise direction, signs on I_{xx}, I_{yy}, and I_{xy} should be reversed so that I_{xx} and I_{yy} are always positive:

$$I_{xx} = \sum_{i=1}^{N} \frac{S_{ii+1}(Y_i^2 + Y_i Y_{i+1} + Y_{i+1}^2)}{12} \quad \text{(B2)}$$

$$I_{yy} = \sum_{i=1}^{N} \frac{S_{ii+1}(X_i^2 + X_i X_{i+1} + X_{i+1}^2)}{12} \quad \text{(B3)}$$

$$I_{xy} = -\sum_{i=1}^{N} \frac{S_{ii+1}(2X_i Y_i + X_i Y_{i+1} + X_{i+1} Y_i + 2X_{i+1} Y_{i+1})}{24} \quad \text{(B4)}$$

where

$$S_{ij} = X_i Y_j - X_j Y_i \quad \text{(B5)}$$

The moment-of-inertia tensor $\mathbf{I}$ is constructed as shown in Eq. (B6). Eigenvalues of $\mathbf{I}$, I_{11}, and I_{22} become the moments of inertia along the principal directions (v_{1_x}, v_{1_y}) and (v_{2_x}, v_{2_y}), respectively. These directions are the corresponding eigenvectors:

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} = \begin{bmatrix} v_{1_x} & v_{2_x} \\ v_{1_y} & v_{2_y} \end{bmatrix} \begin{bmatrix} I_{11} & 0 \\ 0 & I_{22} \end{bmatrix} \begin{bmatrix} v_{1_x} & v_{2_x} \\ v_{1_y} & v_{2_y} \end{bmatrix}^T \quad \text{(B6)}$$

Since the moments of inertia, I_{11} and I_{22}, of a rectangle with width w and height h are expressed as in Eq. (B7), the effective aspect ratio AR can be defined as the square root of the ratio between I_{11} and I_{22}, as in Eq. (B8):

$$I_{11} = \frac{1}{12} w h^3 \qquad I_{22} = \frac{1}{12} w^3 h \quad \text{(B7)}$$

$$\text{AR} = \frac{h}{w} = \sqrt{\frac{I_{11}}{I_{22}}} \quad \text{(B8)}$$

Appendix C: Complete Results and Computational Environment

Table C1 lists the complete results for 79 airports. All of the computations are performed using a Mac Pro desktop computer with dual 3.0 GHz Intel Zeon quad-core processors with 10 GB of total memory. The graph generation code was written in 32 bit Java 1.5 without particular attention to performance optimization and with a 2 GB memory limit. A node-link graph for each airport was computed using a single thread.

Table C1 Full list[a] (in no particular order)

Airport	Links	Runtime, s
ABQ	Failed	
Andrews Air Force Base (ADW)	110	6.8
Centennial (APA)	88	7.1
Austin–Bergstrom Int'l (AUS)	118	9.2
Rocky Mountain Metropolitan (BJC)	143	10.6
BUR	88	6.1
CID	52	2.4
Charlotte/Douglas Int'l (CLT)	269	35.9
City of Colorado Springs Municipal (COS)	125	14.4
Daytona Beach Int'l (DAB)	201	14.8
Ronald Reagan Washington National (DCA)	148	6.2
DFW	678	283.4
Phoenix Deer Valley (DVT)	120	7.6
Fairbanks Int'l (FAI)	102	10.0
Fort Lauderdale/Hollywood Int'l (FLL)	271	39.9
Honolulu Int'l (HNL)	238	29.6
Washington Dulles Int'l (IAD)	259	44.2
Indianapolis Int'l (IND)	177	33.0
Juneau Int'l (JNU)	36	1.5
Los Angeles Int'l (LAX)	360	78.2
Long Beach (LGB)	201	14.1
Orlando Int'l (MCO)	302	66.9
Memphis Int'l (MEM)	362	73.6
Crystal (MIC)	160	13.7
Minneapolis–St. Paul Int'l (MSP)	331	61.4
Metropolitan Oakland Int'l (OAK)	274	38.6
Ontario Int'l (ONT)	172	20.1
Portland Int'l (PDX)	177	23.0
Phoenix Sky Harbor Int'l (PHX)	336	70.5
Ernest A. Love Field (PRC)	132	11.6
Raleigh–Durham Int'l (RDU)	221	25.0
San Diego Int'l (SAN)	76	2.8
Santa Barbara Municipal (SBA)	120	7.4
Seattle–Tacoma Int'l (SEA)	143	8.7
Norman Y. Mineta San Jose Int'l (SJC)	169	13.3
McNary Field (SLE)	113	6.0
John Wayne–Orange County (SNA)	94	4.5
Lambert–St. Louis Int'l (STL)	282	45.9
Tampa Int'l (TPA)	300	48.3
North Las Vegas (VGT)	100	6.3
Atlantic City Int'l (ACY)	54	3.3
Ted Stevens Anchorage Int'l (ANC)	173	20.2
ATL	380	91.0
Bradley Int'l (BDL)	80	5.2
General Edward Lawrence Logan Int'l (BOS)	207	19.8
Baltimore/Washington Int'l (BWI)	240	25.3
Cleveland–Hopkins Int'l (CLE)	221	16.8
Port Columbus Int'l (CMH)	137	14.3
Cincinnati/Northern Kentucky Int'l (CVG)	259	44.3
Dallas Love Field (DAL)	219	18.5
Denver Int'l (DEN)	512	140.2
Detroit Metropolitan Wayne County (DTW)	533	155.9
Newark Liberty Int'l (EWR)	274	36.4
Flying Cloud (FCM)	164	14.7
Fort Lauderdale Executive (FXE)	227	24.0
HOU	Failed	
George Bush Int'l (IAH)	414	99.0
John F Kennedy Int'l (JFK)	431	114.1
McCarran Int'l (LAS)	306	62.9
La Guardia (LGA)	193	17.9
Kansas City Int'l (MCI)	258	42.5
Chicago Midway Int'l (MDW)	223	20.6
Miami International (MIA)	336	63.7
General Mitchell Int'l (MKE)	276	26.3
Louis Armstrong New Orleans Int'l (MSY)	67	4.0
Will Rogers World (OKC)	242	32.6
ORD	566	182.6
Philadelphia Int'l (PHL)	302	48.4
Pittsburgh Int'l (PIT)	335	60.4
Theodore Francis Green State (PVD)	121	5.7
Reno/Tahoe Int'l (RNO)	179	14.6
SAT	Failed	
Louisville Int'l (SDF)	292	36.4

(continued)

Table C1 Full list[a] (in no particular order) (Continued)

Airport	Links	Runtime, s
SFO	301	48.3
Salt Lake City Int'l (SLC)	297	42.8
Sacramento Int'l (SMF)	63	4.9
Sarasota/Bradenton Int'l (SRQ)	108	5.9
Teterboro (TEB)	74	3.7
Tucson Int'l (TUS)	173	17.4

[a]Int'l denotes International.

References

[1] Brinton, C., Krozel, J., Capozzi, B., and Atkins, S., "Improved Taxi Prediction Algorithms for the Surface Management System," AIAA Guidance, Navigation, and Control Conference and Exhibit, Monterey, CA, AIAA Paper 2002-4857, Aug. 2002.

[2] Windhorst, R. D., and Meyn, L. A., "The Airspace Concept Evaluation System Terminal Area Plant Model," AIAA Modeling and Simulation Technologies Conference and Exhibit, Hilton Head, SC, AIAA Paper 2007-6555 , Aug. 2007.

[3] "SIMMOD Simulation Airfield and Airspace Simulation Report," ATAC Corp. Rept. EP 04118, Sunnyvale, CA, 2006.

[4] Bodoh, D. J., and Wieland, F., "Performance Experiments with the High Level Architecture and the Total Airport and Airspace Model (TAAM)," 17th Workshop on Parallel and Distributed Simulation, IEEE Paper 1087-4097, Piscataway, NJ, Jun. 2003.

[5] Atkins, S., Carnoil, T., Feldman, M., and Neskovic, D., "Surface Management System (SMS) Adaptation Procedures Manual," NASA Contract NNA04BB31D, Dec. 2004.

[6] "SODAA User's Guide Version 1.7," Mosaic ATM, Leesburg, VA, Mar. 2008.

[7] Nolan, M. S., *Fundamentals of Air Traffic Control*, 4th ed., Brooks Cole, Pacific Grove, CA, 2003, pp. 482–483.

[8] McAllister, M. and Snoeyink, J., "Medial Axis Generalization of River Networks," *Cartography and Geographic Information Science*, Vol. 27, No. 2, April 2000, pp. 129–138. doi:10.1559/152304000783547966

[9] Stanislawski, L., and Whitaker, S., "Automated Centerlining of Areal Features Using Thiessen Polygons for Completing Hydrographic Networks in Vector Data," *ASPRS 2003 Annual Conference Proceedings*, Anchorage, AK, Curran Assoc., Red Hook, NY, May 2003.

[10] Yuqing, P., Wenchao, C., and Yehua, S., "The Shortest Hypotenuse-Based Centerline Generation Algorithm," *IJCSNS International Journal of Computer Science and Network Security*, Vol. 9, No. 8, Aug. 2009, pp. 155–159.

[11] Kim, T., Park, S., Kim, M., Jeong, S., and Kim, O., "Tracking Road Centerlines from High Resolution Remote Sensing Images by Least Squares Correlation Matching," *Photogrammetric Engineering and Remote Sensing*, Vol. 70, No. 12, Dec. 2004, pp. 1417–1422.

[12] Yang, Y., and Zhu, C., "Extracting Road Centerline from High-Resolution Satellite Images Using Active Window Line Segment Matching and Improved SSDA," *International Journal of Remote Sensing*, Vol. 31, No. 9, May 2010, pp. 2457–2469. doi:10.1080/01431160903019288

[13] Ramer, U., "An Iterative Procedure for the Polygonal Approximation of Plane Curves," *Computer Graphics and Image Processing*, Vol. 1, No. 3, Nov. 1972, pp. 244–256. doi:10.1016/S0146-664X(72)80017-0

[14] Douglas, D. H., and Peucker, T. K., "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, Vol. 10, No. 2, Dec. 1973, pp. 112–122. doi:10.3138/FM57-6770-U75U-7727