Reported is an experimental technique that has been developed for retrieving, by geometrical means, information related to city maps.

This paper emphasizes the analysis needed to translate a retrieval query into relationships among points, vectors, and polygons. An illustration of the technique is given.

INTERACTIVE GRAPHICS IN DATA PROCESSING

Geometric relationships for retrieval of geographic information

by J. D. Jacobsen

This paper discusses a package of subroutines, written in FORTRAN IV and executed under the IBM SYSTEM/360 Operating System (OS/360), that is designed to process geometric graphics data. The subroutines permit certain relationships to be determined between any two of the following geometric figures: points, vectors, convex or nonconvex polygons, and circles. The intended use of these programs is the retrieval of information indexed with geographic coordinate data.

After briefly mentioning the application for which the subroutines are primarily intended, the package design principles are discussed. The logic associated with several of the subroutines is explained in detail. The explanations are supplemented by an illustrative example of geographic information retrieval.

The subroutine package has application in the design of urban management information systems that use geographically oriented data. In systems of this type, a master file, or data base, is structured to include such file elements as streets, street intersection points, zoning districts (plus facts about them) in terms of a coordinate system. Geographic fact information may include, for example, block and parcel data, population and health statistics, traffic density, law enforcement, and planning and land use data.

Typically, system users—public works, public safety, transportation—seek information contained within an area of interest that is described to the system in coordinate form, as an input query. For example, an urban planner[1] might wish to select from a central data file all street intersection records (represented by a single pair of x-y coordinates) falling within a specified area that has a nonconvex polygon shape. Using this data base, such questions as the following could be answered:

- What are the property values, by type of land use, of parcels to be displaced by a proposed highway?
- What roads require resurfacing in a given geographic area?
- What is the school population within one-half mile of proposed sites for new school construction?
- How many police radio calls were received from a given area?

Retrieval objectives

Experience in military information retrieval suggests the usefulness of geographic search operators. One such system, using an "overlapping polygon technique" wherein the geometric shape representing the area of interest is required to be convex, has proved to be adequate for the applications for which it was designed.

However, with the increasing interest in urban management and planning, it is desirable to include areas of interest that assume a nonconvex shape (police beats, school districts, proposed road-building, etc.). Thus, more general graphic processing techniques are needed. In response to this need, an experimental project was undertaken that produced the subroutines to be described.

These subroutines serve a retrieval function by determining whether a record from a data file belongs to a specified geographic area of interest. It should be understood that the subroutines do not contain a complete query processing capability, i.e., the ability to select or reject records by logically combining data fields specified by the input query. The only query parameters needed by the subroutines are the coordinates of the area of interest.

Since the areas of interest are generally small, the earth is assumed to be locally flat, and ideally the geographic portion of the file data record is oriented to a Cartesian coordinate system. Therefore, no provision need be made for the subroutines to transform from one coordinate system to another. If transformations are necessary, they are included in the user program. The Cartesian coordinate system allows the user to describe his area of interest by means of x-y coordinate data. Moreover, he may enter polygon coordinates in either clockwise or counter-clockwise fashion, although he must enter them in order.

Because programming takes the form of subroutines, array names and coordinate counts are transmitted through an argument list in the calling sequence. Also appearing in this list is a parameter that serves as an output status code. The parameter is assigned a

value by the subroutine and represents the relationship existing between the file element and the query polygon. This code can then be interpreted by the calling program.

For example, an urban planner might wish to select all street intersection records from a central data file (geographically represented by a single pair of x-y coordinates) falling within a specified area of interest that graphically assumes the shape of a nonconvex polygon. The executed subroutine is named the point-polygon subroutine (POINTP). Its function is to determine whether a single point lies inside, outside, on a vertex, or on a polygon side. The determined status is returned to the calling program by assigning to a calling sequence parameter the value 1, 2, 3, or 4, as shown in Figure 1. In this manner, any status determined by the subroutine can be interpreted as an inclusion or a rejection by the calling program.

Figure 1 Point-polygon relationships

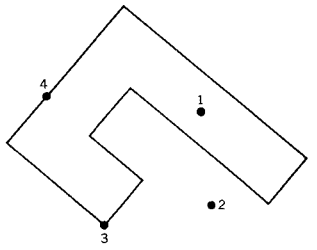
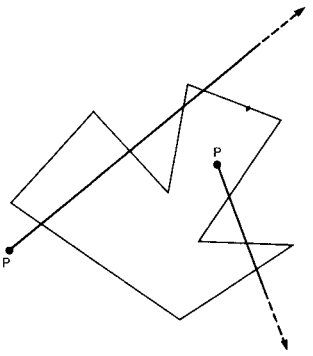


Figure 2 Determining an interior or exterior point

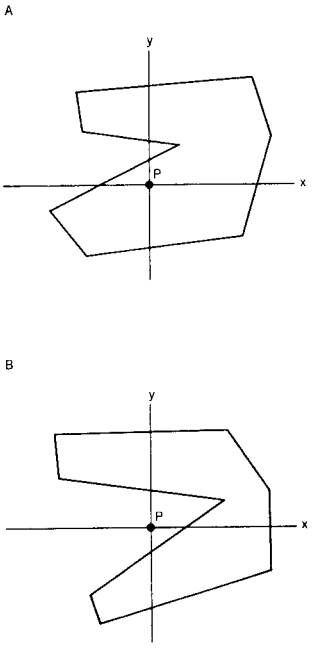


Analysis and design

The following is a discussion of the analysis used in designing some of the subroutines.[2,3] Presented is a solution to the "point nonconvex-polygon problem" and the resulting point-polygon subroutine (POINTP). Also outlined is the basic mathematics involved in determining the required relationships between two straight-line vectors, which result in the vector-vector subroutine (VECTV). Finally, a practical application of these two subroutines is presented in order to explain the vector-polygon subroutine (VECTP), which recognizes various relationships between a vector and a polygon.

A technique for determining whether a single point lies inside or outside a polygon is explained as follows. Construct a point P and a polygon as shown in Figure 2. Extend a line from the point indefinitely in any direction. If this line intersects an odd number of polygon sides, the point is inside the polygon. If the line intersects an even number of polygon sides, the point is outside the polygon. This technique is implemented in the POINTP subroutine by performing an axis translation and by letting point P be the origin of the translated coordinate system. The positive y axis in Figure 3 becomes the extended line.

Figure 3 POINTP subroutine geometry



Counting the number of polygon sides crossing the positive y axis is done by simple comparisons of the signs of the polygon-side end-point coordinates. Coordinates of the two end points falling in adjacent quadrants above the x axis indicate a crossing of the positive y axis as shown in Figure 4A. The positive y axis is a convenient choice because one of the cases that can arise is where polygon side coordinates lie in opposite quadrants. In that case, the y intercept

$$b = y_1 - \left(\frac{y_2 - y_1}{x_2 - x_1}\right)x_1$$

must be computed to determine whether the positive y or negative y axis is crossed as shown in Figure 4B.

Figure 4 Counting polygon-side crossings

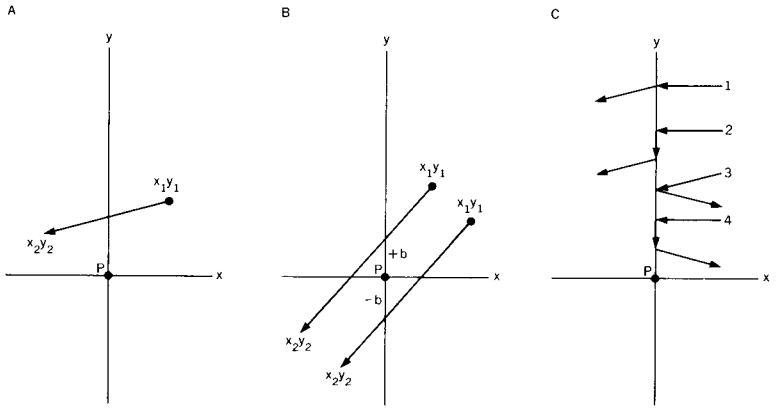


Figure 5 Vector-vector relationships

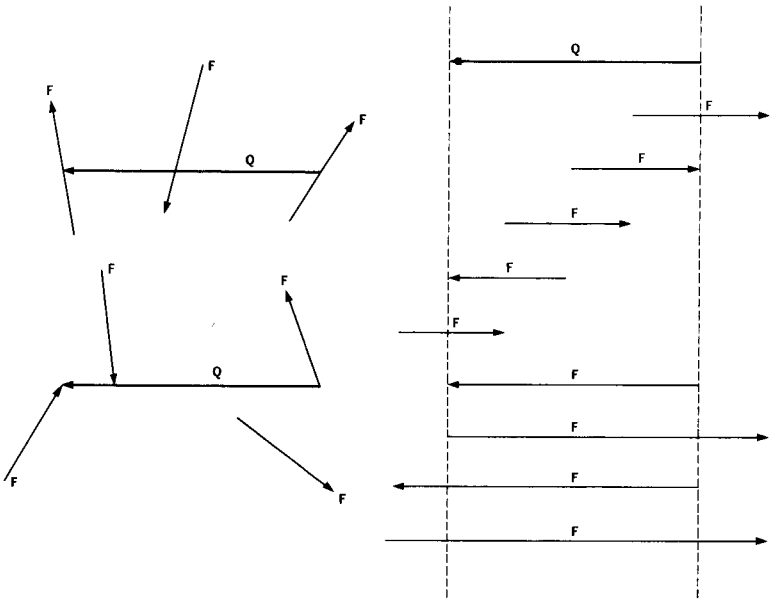


Figure 4C illustrates the remaining axis-intercept conditions that must be recognized by the subroutine POINTP. Vectors 1 and 2 intersect the positive y axis, whereas vectors 3 and 4 do not. Furthermore, if the polygon vector intersects the origin P, the point lies on the vector. If the vector head coincides with the origin, the point is at the vertex.

A subroutine is needed to determine the various relationships between vectors and polygons. The vector-vector subroutine (VECTV) serves this purpose by recognizing relationships between two vectors which, in the practical case, would be a street from a data file (**F**) and a polygon vector (query vector **Q**).

From Figure 5 it can be seen that the conditions that must be recognized basically involve intersections, superimposition, and end-to-end contact. (It is understood that the **Q** and **F** vectors could be directed oppositely to those shown in the figure.) The logic of VECTV takes advantage of the fact that in many cases there is no contact between the file vector and a vector from the query polygon. Therefore, intersection, parallelism, colinearity, and superimposition are looked for in that order. Since we are interested in the intersection of two line segments, we can represent them in parametric form and solve the resulting equations simultaneously.[4] Thus,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_2 & x_1 \\ y_2 & y_1 \end{bmatrix} \begin{bmatrix} t \\ 1 - t \end{bmatrix} \quad \text{where} \quad 0 \leq t \leq 1$$

$$\begin{bmatrix} w \\ z \end{bmatrix} = \begin{bmatrix} u_2 & u_1 \\ v_2 & v_1 \end{bmatrix} \begin{bmatrix} p \\ 1 - p \end{bmatrix} \quad \text{where} \quad 0 \leq p \leq 1$$

For intersection to occur, there must exist a t_0 and p_0 such that $0 \leq (t_0, p_0) \leq 1$, and, when these values are substituted into the above,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} w \\ z \end{bmatrix}$$

Therefore,

$$\begin{bmatrix} t_0 \\ p_0 \end{bmatrix} = \begin{bmatrix} x_2 - x_1 & u_1 - u_2 \\ y_2 - y_1 & v_1 - v_2 \end{bmatrix}^{-1} \begin{bmatrix} u_1 - x_1 \\ v_1 - y_1 \end{bmatrix}$$

whenever the inverse exists.
Simplifying the above,

$$\begin{bmatrix} t_0 \\ p_0 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} \begin{bmatrix} e \\ f \end{bmatrix}$$

so that the problem of solving for t_0, p_0 reduces to $D = ad - cb$, where D is the determinant. Finally,

$$t_0 = \frac{de - bf}{D}$$

and

$$p_0 = \frac{af - ce}{D}$$

Figure 6 graphically illustrates the effect of the calculations t_0 and p_0. If the determinant D equals zero, the two vectors are parallel.

The next step, then, is to determine if the vectors are colinear, and if so, whether they are superimposed. To test for colinearity, a vector cross product is formed as follows:

$$VCP = (y_1 - v_1)(u_2 - u_1) - (x_1 - u_1)(v_2 - v_1)$$

If $VCP = 0$, then the vectors are colinear; if $VCP \neq 0$, then the vectors are not colinear. Figure 7A illustrates the case where two vectors are not colinear and Figure 7B where they are colinear.

Figure 6 Graphic representation of t_0 and p_0

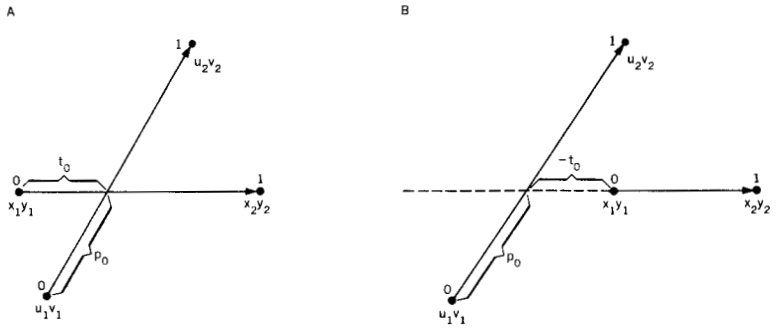
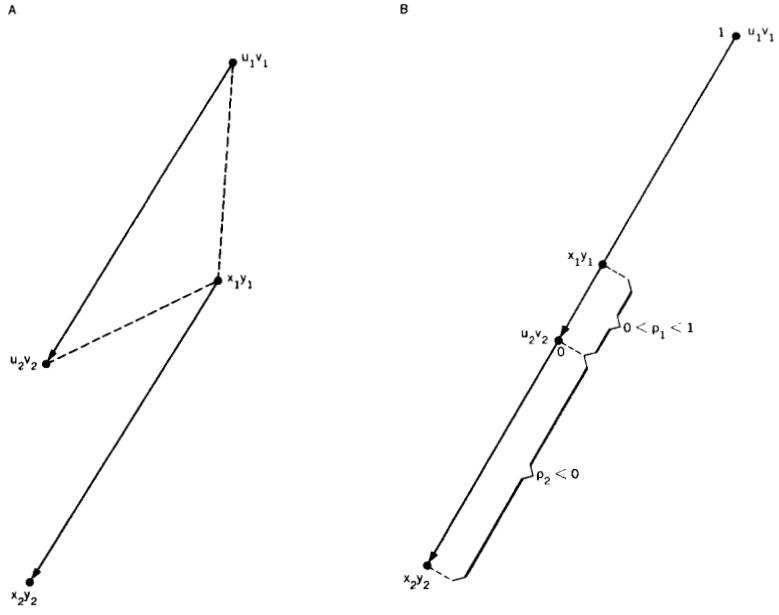


Figure 7 Vector colinearity test



Where the vectors are colinear, a test is made for superimposition by solving for p_1 and p_2, the parametrizations of (x_1, y_1) and (x_2, y_2) in terms of (u_1, v_1) and (u_2, v_2), respectively.[2,5] Thus,

$$(x_i, y_i) = p_i(u_1, v_1) + (1 - p_i)(u_2, v_2) \qquad \text{for } i = 1, 2$$

If $u_1 \neq u_2$, as shown in Figure 7B, then

$$p_i = \frac{x_i - u_2}{u_1 - u_2} \qquad \text{for } i = 1, 2$$

Otherwise, $v_1 \neq v_2$, and

$$p_i = \frac{y_i - v_2}{v_1 - v_2} \qquad \text{for } i = 1, 2$$

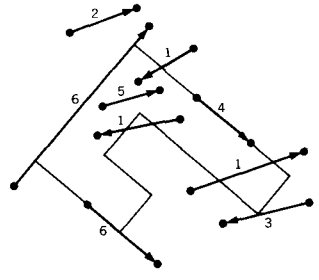
Both p_1 and p_2 must be evaluated to distinguish between the nine cases illustrated in Figure 5. Figure 7B shows superimposition, since $0 < p_1 < 1$. For superimposition to occur,

$$0 < [p_1 \cup p_2 \cup (p_1 \cap p_2)] < 1$$

Once it is established that there is some contact between the two vectors, a coordinate list is set up. This list is a four-word array whose name and number of entries appear in the VECTV calling sequence and contains either the coordinates of intersection, the coordinates involved in superimposition, or the coordinates of end-to-end contact.

The previously described subroutines POINTP (point-polygon) and VECTV (vector-vector) are now used to recognize various relationships existing between vectors and polygons. A vector may partially be included in a polygon under three conditions: (1) at least one vector point must be inside the polygon (as determined by POINTP), or (2) both points must be inside and the vector intersects the polygon, or (3) both points are outside the polygon and the vector intersects the polygon (as determined by VECTV). If both points are inside the polygon and the vector does not intersect any polygon sides, then the vector is totally included in the polygon. (Vector-polygon relationships are illustrated in Figure 8.)

Figure 8 Vector-polygon relationships



The subroutine logic is organized as follows. First, both vector end points are investigated to determine the combination of point-polygon relationships that exists. These relationships are shown systematically in the point-polygon status matrix in Table 1. The status matrix elements are pointers to segments of subroutine logic executed after the polygon is processed against the vector, which is done to look for intersections, superimpositions, etc.

Thus, when the relationship of one vector endpoint with respect to the polygon is established, we can refer to the POINTP output that reflects this relationship as i and similarly for the other endpoint j. Then, the point-polygon status matrix element (i, j) in Table 1 is the program pointer assigned. For example, the element 9 [matrix element $(i, j) = (2, 2)$] in the status matrix serves as a pointer to the subroutine segment to be executed when both vector end points are found to be outside the polygon. The segment is executed after processing the polygon against the vector.

Table 1 Point-polygon status matrix

POINTP *Output* (*i*)	POINTP *Output* (*j*)			
	1 In	2 Out	3 On vertex	4 On line
1 In	1		4	5
2 Out		9	12	15
3 On vertex	4	12	16	20
4 On line	5	15	20	25

Figure 9 Relating a vector to a polygon

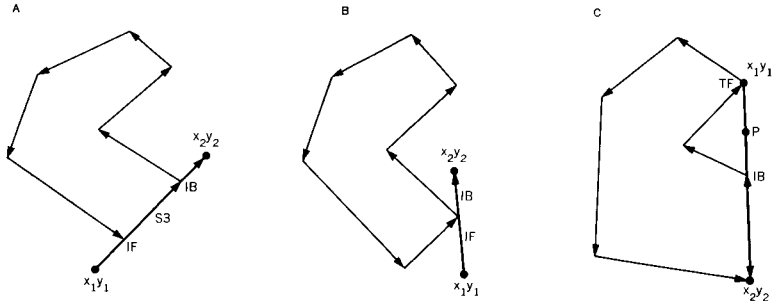
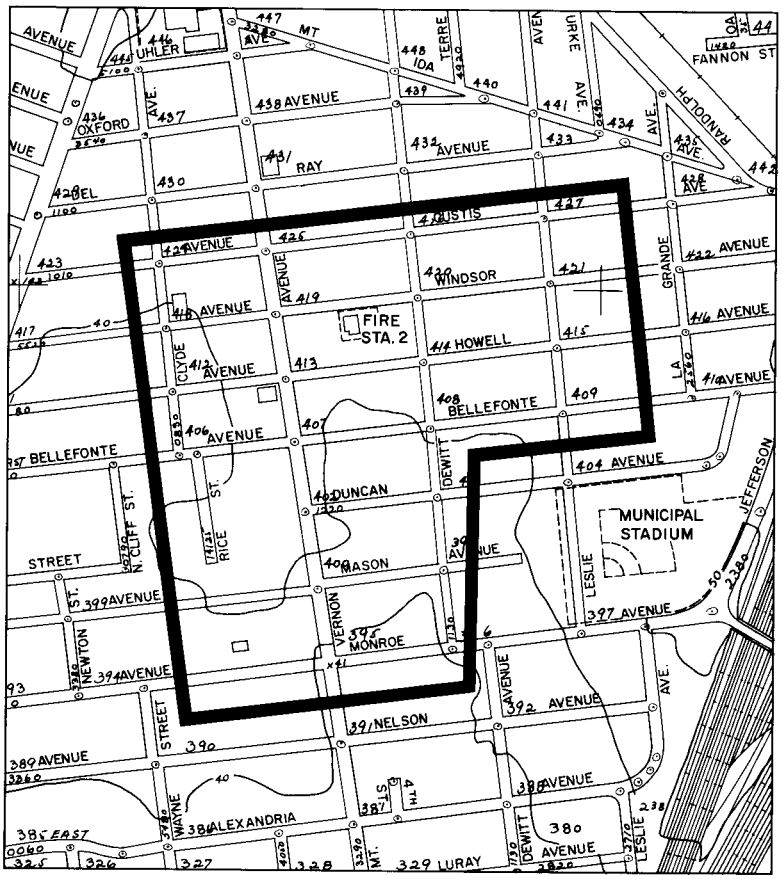


Figure 10 Map section with query polygon



As a vector is investigated by the VECTV subroutine to determine its relationship to each polygon vector, the VECTP subroutine records the VECTV output (transmitted through the calling sequence output parameter) by assigning a truth value to a corresponding FORTRAN IV logical variable. These logical variables are given in Table 2, and some of them are shown in Figure 10. In Table 2, for example, if a vector intersects a polygon vector at its initial point, a truth value is assigned to the logical variable IB.

Table 2 FORTRAN IV logical variables for a vector and a polygon vector

Polygon vector	*Vector*				
	Intersects	*Touches*	*Wholly contained in polygon vector*	*Extends beyond either initial or terminal point of polygon vector*	*Extends beyond both initial and terminal points of polygon vector*
Between initial and terminal points	IM	TM			
At initial point	IB	TB			
At terminal point	IF	TF			
Superimposition			S1	S2	S3

Whenever an intersection is encountered, processing terminates because the vector lies partially inside the polygon. If no intersections are found and if polygon processing ends, then the subroutine segment represented by the appropriate status matrix element (i, j) is selected. Each subroutine segment is designed to interpret the results of the polygon processing phase by logically interrogating and combining the results to produce the types of vector-polygon relations required. For example, Figure 9 shows three of a number of possible relations that might exist between a vector and a polygon. In Figures 9A and 9B, the POINTP subroutine determines that both vector end points are outside the polygon. In these cases, the status matrix indices (i, j) would have the values (2, 2). Processing the polygon against the vector yields no intersections. However, the logical variables IF, S3, and IB are "true" for Figure 9A, and IF and IB are "true" for Figure 9B.

The status matrix indices (2, 2) are used to select subroutine segment 9, the logic of which is as follows. If S3 is "true," the subroutine output parameter reflecting the vector-polygon relationship is assigned the value 6, as shown in Figure 8, and control is returned to the user program. Otherwise, if IB is "true," the output parameter is assigned the value 3, indicated in Figure 8, and control returns to the user program. If neither S3 nor IB is true, then there is no contact between the vector and the polygon. In this case, the value 2 from Figure 8 is assigned to the output parameter, and control returns to the user program.

If both of the vector end points coincide with the vertices of the polygon, as shown in Figure 9C, then a different segment of subroutine logic is executed (number 16 in the status matrix in Table 1) providing no intersections were encountered in processing the vector against the polygon. During this processing, the coordinates of the two points labeled IB and TF are placed in a list. Thus, inclusion or superimposition is determined by (1) averaging the

coordinates of IB and TF and (2) using the point-polygon program to determine whether the average (point P) is inside or outside the polygon.

Geographic information retrieval

Since the motivation for our work is to develop methods for processing geographically oriented data, a brief illustration is given here. (Other applications and subroutine testing are discussed in greater detail elsewhere.[6]) The example is a public-works query for which a listing is produced of streets within an area of a city bounded by a polygon. The data base consists of a street file and a land-parcel file in the city of Alexandria, Virginia. This data base is linked to city maps through a digitizing and file expanding process. An x-y coordinate reader was used to digitize the coordinates that identify streets and to inititate punching the coordinates into cards. The cards were used to insert the coordinates into the street file. Data files are stored on an IBM 2314 direct-access storage device. File manipulation is effected through IBM 1050 remote terminals and SYSTEM/360 operating under OS/360 and a generalized file maintenance, retrieval, and formatting program called FASTER.

A digital representation of the polygon area of interest enters the computer via the remote terminal and an x-y digitizer. A number that represents a preprogrammed query type (creating a listing of streets, in our example) is also entered by the digitizer through the 1050. (Query input data are displayed on a typewriter terminal, while the output is produced by both typewriter and an IBM 1403 printer.)

Test results demonstrate the feasibility of applying the geometric graphic subroutines to geographic information retrieval. Figure 10, in our example, shows the query area superimposed on a section of the Alexandria city map, and Figure 11 is the output street listing corresponding to our example query polygon. Street identifications are given by numbers rather than by street names. For example, the designation S-3290-1600 refers to street file S, Mt. Vernon Avenue (3290), and to the lowest street address (1600) in the block desired.

Concluding remarks

This discussion has sought to strike a balance between comprehensiveness and detail in the exposition of mathematical and programming methods for processing geometric graphics data. Therefore, it seemed necessary to omit discussions of several techniques and subroutines that have also been worked out.

Work described in this paper could benefit those engaged in or contemplating the design of urban management information systems geared to a geographically oriented data base. The experimental programming system described in this paper is evolving to meet and anticipate requirements of those seeking urban improve-

Figure 11 Street listing for query polygon

URE2 — CITY OF ALEXANDRIA PUBLIC WORKS-TRAFFIC AREA — PAGE 0001

STREET ID	METH CNST	YR CNS	LF YR	LAST RESUF	TYPE OF RESRFNG	RTG WY WIDTH	WIDTH PAVMT	WIDTH BTCRB	LGTH BLCK	SURFACE TYPE	SURF THCK	SURF COND	BASE TYPE	BASE THCK	BASE COND	MEDIAN TYPE
S-027C-0100	CITY	38	84	-		40	20	24	877	PLANT MIX	2	GOOD	GRAVEL	8	GOOD	NO MEDIAN
S-027C-0150	UNK			-						UNPAVED		N/A	N/A		N/A	NO MEDIAN
S-027C-0200	CITY	33	72	-		40	20	24	520	UNPAVED	1	FAIR	GRAVEL	8	FAIR	NO MEDIAN
S-027C-0300	CITY	33	72	-		40	20	24	516	UNPAVED	1	FAIR	GRAVEL	8	FAIR	NO MEDIAN
S-317C-0300	CITY	49	84	-		50	36	40	482	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-1600	CITY	29	84	-		60	36	40	290	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-1700	CITY	29	84	-		60	36	40	270	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-1800	CITY	29	84	-		60	36	40	270	PLANT MIX	2	FAIR	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-1900	CITY	29	84	-		60	36	40	230	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-2000	CITY	29	84	-		60	36	40	230	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-329C-2100	CITY	29	84	-		60	36	40	230	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-4125-0000	UNK			-						UNPAVED		N/A	N/A		N/A	NO MEDIAN
S-552C-0100	UNK		84	-		40	20	24	403	PLANT MIX	2	GOOD	PORTLD CEMENT	8	GOOD	NO MEDIAN
S-552C-0200	CITY	36	72	-		40	20	24	525	UNPAVED	1	FAIR	GRAVEL	8	FAIR	NO MEDIAN
S-552C-0300	CITY	36	72	-		40	20	24	516	UNPAVED	1	FAIR	GRAVEL	8	FAIR	NO MEDIAN

ments by applying systems engineering techniques to management, planning, and administrative functions.

ACKNOWLEDGMENTS

The author gratefully expresses his appreciation to the Metropolitan Washington Council of Governments for its assistance in acquiring test data and to Mr. R. Brieman, Director of Data Processing, City of Alexandria, Virginia, for making the test data files available. We also acknowledge and appreciate the technical contributions of Messrs. J. Kohn and K. Tuggle, also associated with the Metropolitan Washington Council of Governments. IBM personnel who made helpful contributions are S. H. Brounstein, J. Hess, R. Hogan, Dr. T. Putney, R. Siegmann, C. Touchton, and R. Vaughan.

CITED REFERENCES

1. L. Lessing, "Systems engineering invades the city," *Fortune* **77,** No. 1, 155–157, 217–221 (January 1968).
2. Adams Associates, Inc., "Computer display fundamentals—software techniques," *Computer Display Review* **1,** 11.66.0–11.118.0.
3. L. G. Roberts, *Machine Perception of Three-Dimensional Objects,* Lincoln Laboratory Technical Report 315–22, Lexington, Massachusetts (May 1963).
4. R. B. Dial, *Street Address Conversion System,* Research Report No. 1 (1964), may be obtained from the Urban Data Center, University of Washington, Seattle, Washington.
5. D. V. Ahuja and S. A. Coons, "Geometry for construction and display," in this issue.
6. S. H. Brounstein, "Some concepts and techniques for constructing and using a geographically oriented urban data base," to be published in the *Journal of Socio-Economic Planning Sciences,* Pergamon Press, New York, New York (1968).