MOSAICKING OF AERIAL PHOTOGRAPHIC MAPS VIA SEAMS DEFINED BY BOTTLENECK SHORTEST PATHS

ELENA FERNANDEZ

Universitat Politecnica de Catalunya, Barcelona, Spain

ROBERT GARFINKEL

University of Connecticut, Storrs, Connecticut

ROMAN ARBIOL

Institut Cartografic de Catalunya, Barcelona, Spain

(Received November 1996; revisions received September 1997, December 1997; accepted January 1998)

The Cartographic Institute of Catalonia (ICC) produces commercial aerial photographic maps of locations in Europe and South America. These maps are often so large that it is necessary to produce one map from two or more photographs, which are combined two at a time in a process called *mosaicking*. The objective is to make the final map appear to be the product of a single photograph by producing a seam that is invisible even to an expert cartographer. The problem and a variation are modeled via bottleneck shortest paths and cycles. Optimization algorithms are developed for both, and the first has been implemented with demonstrable impact on the company. The second represents a new class of constrained shortest cycle problems.

The Cartographic Institute of Catalonia (ICC) is the map agency of the government of Catalonia, an autonomous region of more than six million people in northeastern Spain. ICC was established in 1982 and has about 235 employees. Its budget is approximately 30 million dollars, of which about one-fourth comes from outside projects. Clients outside Spain are mainly cartographic institutes of other countries of Europe and South America. ICC's services are in demand by other cartographic institutes mainly because, perhaps due to its relative youth, it is very modern and technology-oriented. Thus it is able to produce maps using digital representations and computers that are beyond the scope of more traditional companies not having these capabilities. Competitors include Eurosense (Belgium), Geonex (United States), Hansa Luftbild (Germany), and Intera (Canada).

Because ICC is a state agency it is bound by certain bureaucratic regulations. For instance, it is virtually impossible for ICC to increase its work force, although its budget for computers and technology is ample. ICC wants to expand its level of outside funding and compete successfully for contracts involving the creation of thousands of maps. Seemingly this could be achieved only by automating processes involving large amounts of human time. The subject of the rest of this paper is one such process, namely a stage in combining (*mosaicking*) two or more aerial photographs into a single image in the production of photographic maps.

Digital Orthophotography

About one-fourth of ICC's business is production of aerial photographic maps. These are made from aerial photographs, which are then turned into *digital orthophotos*. To digitize a photograph, the given area is first divided into very small, equal-sized pixels. If the photo is black and white, every pixel is then assigned a single numerical value corresponding to its light intensity. Color photos are handled in an analogous way by transforming a vector of light intensities for different color bands into a single number. Here we assume that the photos are black and white and refer the reader to Fernandez et al. (1996) for the details of the transformation for color photos. A general reference for digital image processing is Jensen (1986).

The digital photograph becomes a digital orthophoto (i.e., is "registered") when every pixel is placed in its precise geographic position by a program that takes into account the location of the camera and the orientation of the camera platform, as well as the heights of all the points in a bidimensional grid of the area photographed. Because this work deals with orthophotos, it is simple to determine that a given pixel, i.e., the pixel in row i and column j in

Subject classifications: Cartography: mosaicking of photographic maps. Networks: bottleneck shortest path.
Area of review: OR PRACTICE.

0030-364X/98/4603-0293 $05.00

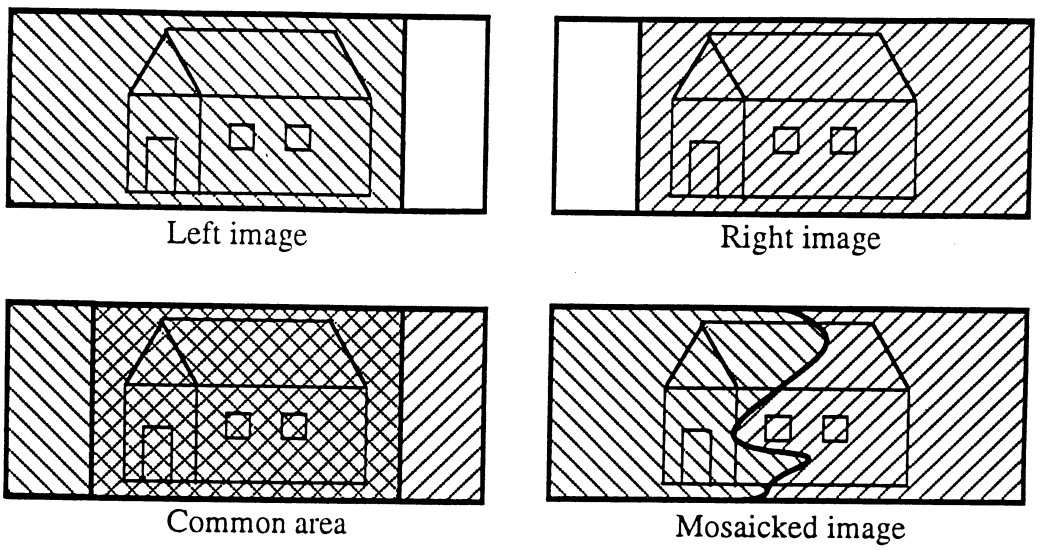

Figure 1. An illustration of seam-based mosaicking.

two different photos, corresponds to the same area on the ground in both photos.

Mosaicking

The maps produced by ICC are often so large that it is necessary to produce one map from two or more photographs, which are mosaicked two at a time. In mosaicking it is desired both to make the map realistic and to make it appear to be the product of a single photograph. At ICC and other modern cartographic companies, the mosaicking steps are performed using a digital, rather than an analog, representation of the photos. Commercial mosaicking packages exist but are not designed to produce the high-quality maps that companies such as ICC are hired to produce. Consequently, such companies typically design their own software.

Combining two (initial) photos of a scene into a single one—so that one cannot tell that the result is the product of multiple photos—is not a simple task, as indicated by the following quote from Milgram (1975): "The most serious problem, however, is due to changes in what is being scanned—the earth and its atmosphere . . . These factors and others make it all but impossible for manual or photographic methods to produce a photomosaic that does not include distracting, *artificial edges*." The process is done digitally at ICC by adjusting the light intensity of the pixels. (The analog version of the same process is performed by many photographers by controlling the development of the film.) Three basic steps are involved: *initial smoothing*, *combining*, and *final smoothing*. In the first step the two pictures are made radiometrically similar; that is, the levels of color or black/white are made to be as close as possible.

In the second step some common area of the two photos is chosen, and the photos are then combined within that area. There are various ways to do this digitally. It is feasible to have the light intensity of each pixel in the area be some combination of its content in the two photos. For instance, the pixel in row i and column j could have light intensity given by 0.4 times the intensity in one initial photo plus 0.6 times the intensity in the other. A special case is *seam-based mosaicking*, where each pixel is represented entirely by one photo or the other, based on which side of a seam it lies, as in Figure 1.

This is the preferred method at ICC because, as is true in most instances, if the two photos have been taken from even slightly different angles, so that the shadows or glares are different on given objects, then a nonseam-based mosaic is likely to yield a picture that does not correspond closely to reality. To the best of our knowledge, all commercial mosaicking packages (e.g., *Erdas* 1996) are nonseam-based. In seam-based mosaicking there is the specific objective of making the seam invisible in the final photo.

In the final step the combined photograph is developed again while attempting to make the area around the seam even more radiometrically uniform. At ICC initial and final smoothing are done by simple, efficient adaptations of well-known techniques, but seam drawing had been a bottleneck until the implementation of the work described below.

Although a final map may be produced from more than one seam, we will focus on methods for producing seams one at a time, since this is how seams are produced in practice. Thus the data in everything that follows will be assumed to be exactly two initial photos. We use the convention that the seam goes from the top border to the bottom rather than from side to side, so that the final photo uses pixels from one initial photo on the left of the seam and from the other on the right. The pixels that make up the seam itself can be chosen randomly from either photo.

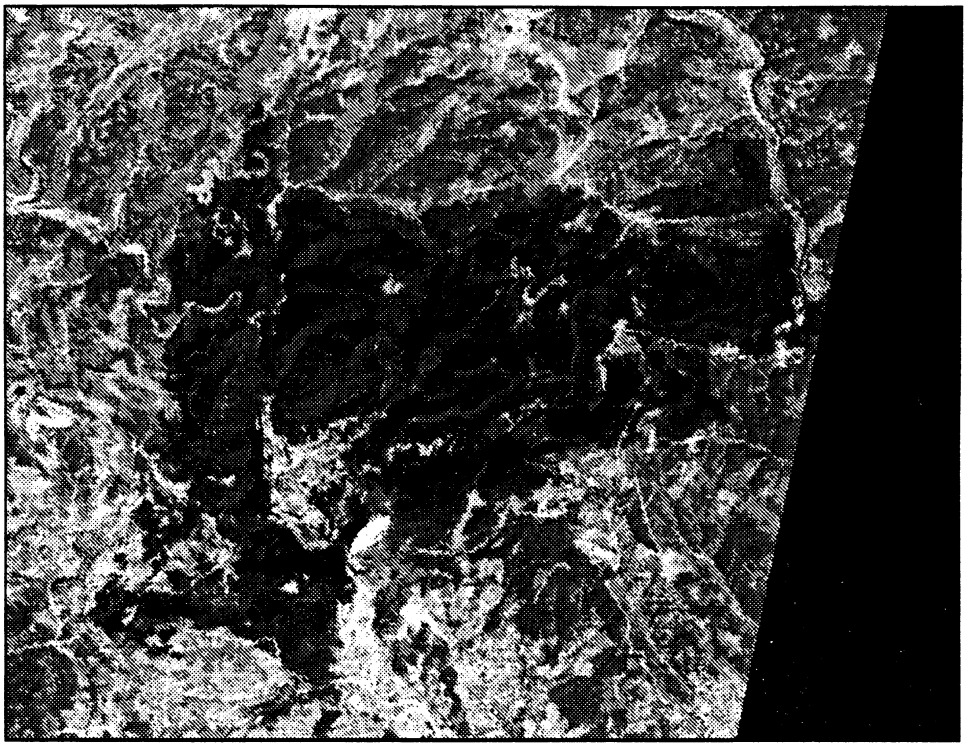

Figure 2a. The left image—after the fire.

Previous Method for Seam-Drawing at ICC

Prior to this work, seams required about an hour per seam and were produced at ICC by operators working at graphics terminals. The operators used a combination of experience and intuition to choose a sequence of points, which were then connected with straight lines by the computer. They chose an initial bandwidth for the seam but would often have to enlarge it or move it sideways in the course of the construction. A typical problem size is 4,000 by 6,000 pixels, but forcing such a large picture on a screen would entail great loss of detail. Instead the operators worked on subproblems of size 1,000 by 1,000, so it was necessary to zoom in and out of these smaller work areas. At the same time, they were able to view the partial seam in up to eight surrounding areas, which was useful in determining the direction to follow next.

They knew that, in the absence of natural borders of farms, city blocks, etc., they had to avoid "dangerous areas" where the light intensities of many of the pixels were very different in the two photos. Seams that go through such areas are quite likely to be visible. Their most common strategy was to first seek natural borders and follow them whenever possible. Although such seams may go through dangerous areas, the idea was that they would be disguised by the natural borders. In places where natural borders were not available, they would generally have less confidence in their decision making and would therefore choose points that were relatively close together, making the process quite time consuming.

Eventually it was always possible to produce an acceptable seam, but the difficulty of the process depended significantly on the data. Previously ICC tried to make this step easier by making consecutive photos as similar as possible, i.e., taking photos from angles of gradually increasing difference. Normally the common overlapping area of two consecutive photos would be 80 percent. A down side of this approach was that it required many flights, a good deal of fuel, and correspondingly large amounts of film.

We now make the reason for avoiding dangerous areas more concrete.

Defining the Cost Matrix

Let the light intensity of pixel (i, j) be l_{ij} for the left photo and r_{ij} for the right. These numbers range from 0 to 127 where larger numbers denote more light. Define the "cost" $d_{ij} = |l_{ij} - r_{ij}|$ so that $d_{ij} \in [0, 127]$.

The photos of Figure 2 give an extreme example of why a seam that goes through areas of high cost may be visible to the naked eye. Figures 2a and 2b contain the left and right images respectively of an area in which the right image was taken first, and between the taking of the two a forest fire occurred. Figure 2c shows the image of differences along with the seam produced by the algorithm of Section 3. Pixels of low difference are represented by grays of medium darkness while those of large difference are very light or very dark. Figures 2d and 2e contain mosaicked images produced by a very visible seam going directly through the area of large differences and by the

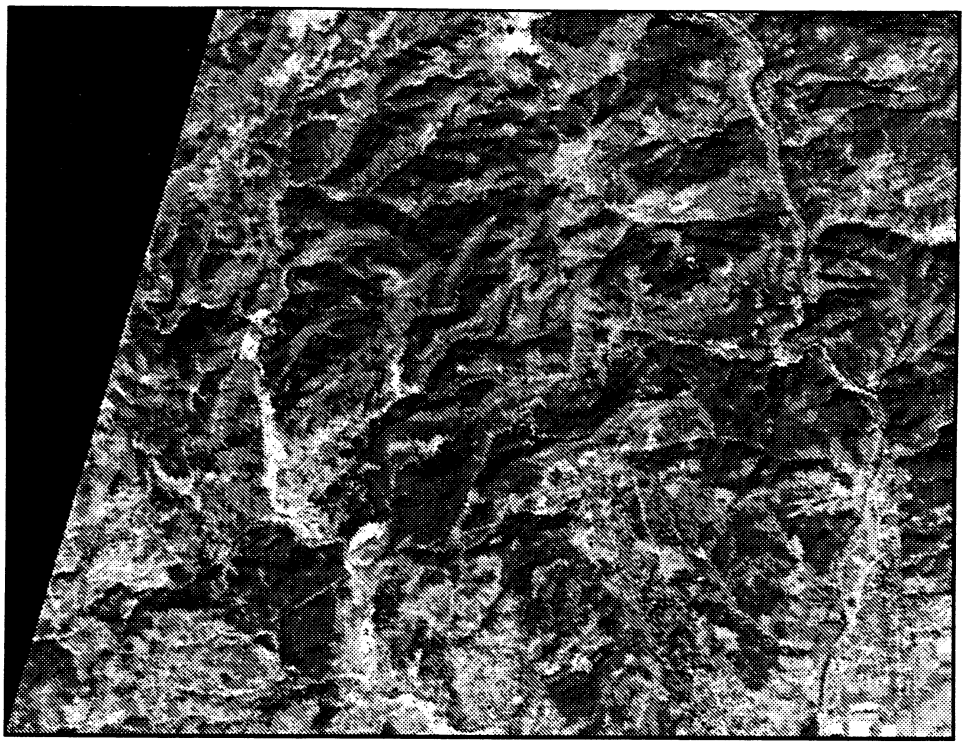

Figure 2b. The right image—before the fire.

invisible seam of Figure 2c, which avoids that area. Note that a good seam could pass either to the left or right of the area of the forest fire, which would determine whether the area appears burned in the final map. In this case a seam passing to the right would probably be preferred, since the left photo is more recent.

This example was chosen to make it easy to appreciate the difference between visible and invisible seams. For

Figure 2c. The image of differences and a seam produced by the algorithm.

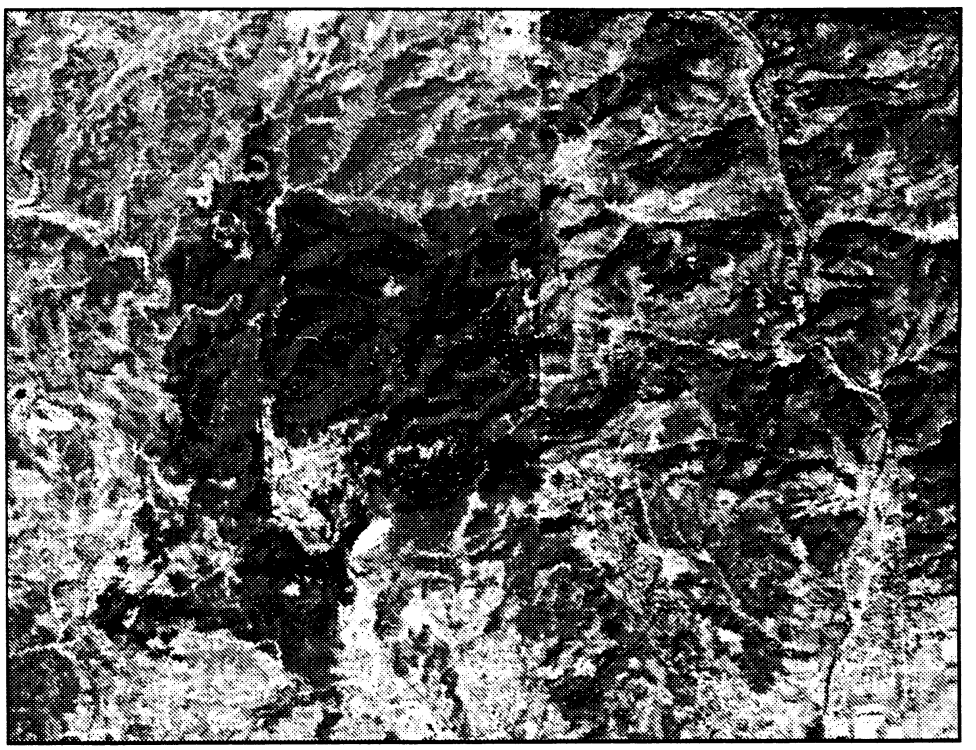

Figure 2d. A mosaicked image resulting from a disastrous seam.

most problems, the difference is likely to be more subtle because the two photos will have been taken moments apart on a single flight. Then it might take the practiced eye of a cartographer to notice a visible seam.

In general it is preferable that the seams not resemble straight lines since these may be more easily detectable by the human eye than seams that are "less straight." For this reason, and also because of the desire to avoid areas of

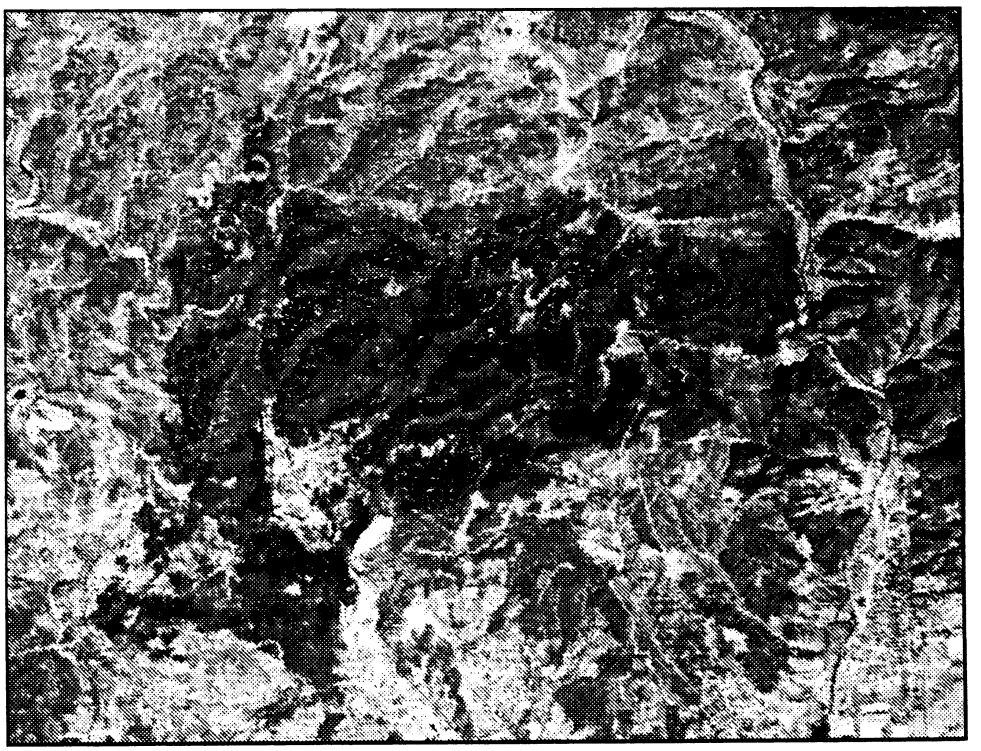

Figure 2e. A mosaicked image resulting from the seam produced by the algorithm.

high difference, it is important that any technique to produce seams automatically can yield a path that is able to change both vertical and horizontal directions. Attempts had been made at ICC to develop an effective procedure that had the property of being able to change vertical direction, but without success. Previous published approaches (see Section 1.4) also suffer from the same limitation. Depending on the data, the model and algorithm of Sections 1 and 2 produce seams of virtually any imaginable shape.

Outline

In Sections 1 and 2 we develop a model and algorithms based on bottleneck shortest paths for the problem of finding optimal seams. The implementation and impact of the algorithm of Section 2 at ICC is the subject of Section 3. Sections 4 and 5 contain a model and algorithm for a variation of the problem in which an otherwise usable photo contains an area that must be removed and replaced by the same area from another photo. The problem of finding the optimal seam is modeled as a bottleneck shortest cycle problem with a side constraint. Conclusions are given in Section 6.

1. THE MODEL FOR AN OPTIMAL PATH SEAM

1.1. An Overview

The model we use to define an optimal seam is given below. This definition of optimality is a surrogate for "*the seam is invisible to the human eye.*" Unfortunately, that definition is impossible to quantify a priori. Instead we qualitatively define a seam to be optimal if the most "dangerous" area it visits is as "safe" as possible.

1.2. The Model

Two pixels are *adjacent* if they share a vertical or horizontal border. The vertices V of the grid network $N = (V, E)$ are the cells of the $m \times n$ matrix. There are a total of $2mn - m - n$ undirected edges, corresponding to all pairs of adjacent vertices. A *path seam* PS is a simple path from any cell of Row 1 to any cell of Row m. Let the cost of any PS be given by

$$f(PS) = \text{maximum } d_{ij}, \quad (i, j) \in PS.$$

Then the *path seam problem* is

$$\text{minimize } f(PS), \quad PS \text{ is a path seam.} \tag{1}$$

Note that the costs d_{ij} in (1) are associated with the vertices of N rather than with the edges. One could think of redefining the network so that the costs would be associated with the edges. This is not necessary because algorithms for shortest paths can handle vertex costs as easily as edge costs.

1.3. Justification of the Model

We chose the model (1) for a variety of reasons. First, optimal solutions to (1) produce seams that are virtually always invisible and therefore acceptable to ICC. Second, optimal solutions can be found within a reasonable amount of computer time. Third, with this type of objective it is at least possible to quantify half of the question of whether a seam will be visible. That is, if the value of the optimal solution is less than 20 we can be almost sure that the seam will be invisible. On the other hand, the converse is generally not true. Even if the seam passes through a few values of high cost it is possible that it will be invisible. Thus (1) is a conservative model.

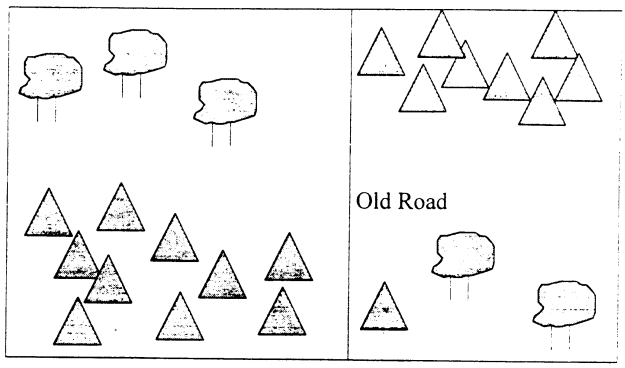

Figure 3a. Before the new road.

Although it may seem more intuitive to use the more standard summation shortest path, that model is much more likely to produce visible seams. This follows because a summation shortest path may trade off passing through some areas of high cost if it can balance them by passing mainly through areas of extremely low cost. Yet passing through any area of high cost can be disastrous. For an extreme example where a few high-cost pixels in a seam can be calamitous, consider Figures 3a and 3b, in which a new road was built in the time period between the taking of the two photos and would therefore be represented by pixels of large d_{ij}. If the seam were to pass through the new road, as shown in Figure 3b, then the final photo would contain only part of that road.

A summation model will also tend to produce paths with fewer pixels because the cost of every pixel is added to the

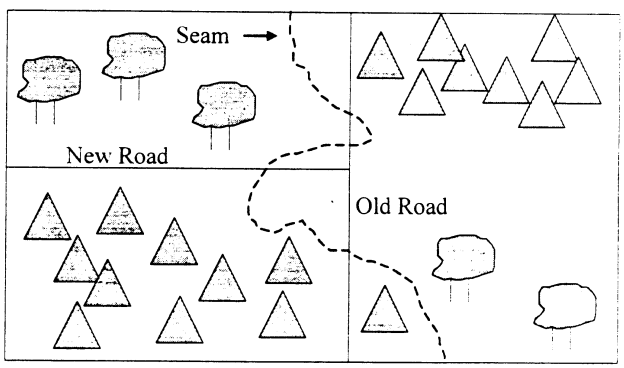

Figure 3b. The new road intersected by a seam.

Table I
An Optimal Seam

2	6*	7	1	10	12	15	7
1	3*	5	23	18	16	17	4
11	8*	19	10	2*	8*	4*	9
13	2*	4*	19	6*	21	1*	11
15	17	5*	7*	3*	10	2*	6*
18	1	17	13	17	14	15	2*
1	16	14	16	18	9	3	7*

objective function. With the bottleneck model (1), however, no penalty is paid for having longer paths. This allows the seam to have the flexibility to meander through the matrix creating the desired irregular patterns. (For instance, an optimal seam of cost 8 is given by the starred cells in the example of Table I.)

An alternative objective would be to minimize $\sum_{(i,j)\in PS} d_{ij}/\text{num}(PS)$, where num($PS$) is the number of pixels in the seam. This could be more appealing than the simple summation model in that it would be less likely to produce straight lines. However, such a model is essentially nonlinear and could still suffer from the problem of accepting a high-cost pixel if it allows very many low-cost pixels to be used. In addition, developing a good exact algorithm for it would be problematic.

The model (1) is one of a class of minimax or bottleneck optimization models introduced by Edmonds and Fulkerson (1968). Variations of (1) have been introduced to model entirely different applications, such as the problem studied by Berman and Handler (1987). A characteristic of (1), which is true of minimax models in general, is that there are likely to be very many alternate optimal solutions because the costs of any cell other than the worst are irrelevant. One could then think of incorporating a secondary objective to try to choose among the alternate optima. In any case, the model (1) has performed so well that there has not yet been an incentive to consider alternatives or to add a secondary objective.

1.4. Other Approaches

It has previously been recognized that an ideal seam should go through pixels of low d_{ij}. Milgram (1977) suggests the model

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}, \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \text{all } i, \tag{3}$$

$$x_{ij} + x_{i-1,k} \leq 1, \quad i = 1, \ldots, m-1 \text{ and } |j-k| > r, \tag{4}$$

$$x_{ij} \in \{0, 1\} \quad \text{all } i, j, \tag{5}$$

where r is a given constant. The model (2)–(5) determines exactly one point of the seam in each row. When the solution yields points in consecutive rows but in different columns, the seam is defined by first drawing a horizontal line in the upper row and then dropping vertically to the lower one. Solution is by dynamic programming.

While acceptable seams are sometimes produced by this model, two deficiencies of it are noted by Milgram. First, the seams cannot change vertical direction: they proceed one row at a time from top to bottom and are thus limited in the paths they can find. Second, the d_{ij} values of the cells in the lines connecting the points chosen by the model are ignored. If r is large, very bad line segments can be chosen, while if r is small the final seam is guaranteed to resemble a straight line, which is undesirable.

Heuristic variations of this approach, which do not propose a model but simply try to follow low values of d_{ij}, have been proposed by Milgram (1975) and Shiren et al. (1989). Both produce seams that suffer from the same deficiencies as the model of Milgram (1977). ICC had experimented with all these algorithms, with results that were often unacceptable in that they produced visible seams.

2. AN ALGORITHM FOR THE PATH SEAM PROBLEM

As pointed out by Berman and Handler (1987), the algorithm of Dijkstra (1959) can be used to solve (1) with very minor modifications. Rather than labeling each vertex with the sum of the d_{ij} to that point, the vertex is labeled with the maximum d_{ij} of any vertex encountered along the way. Since $d_{ij} \leq 127$, the algorithm can be implemented in $O(mn(1 + \log(mn))$ (see Ahuja et al. 1990). We have experimented with this straightforward approach and found that it was computationally unwieldy for very large problems because the file of labeled vertices often grew to be very large. Instead we have implemented a modification that uses the same labeling imbedded in a bisection search over the range of potential values of the objective function, f.

Such bisection search algorithms have been proposed for other bottleneck optimization models, e.g. in Garfinkel et al. (1977). The bisection algorithm offers some advantages over the one-pass algorithm. Each iteration tends to go very quickly, in part because the file of labeled vertices never grows too large, eliminating the main problem with the one-pass algorithm. Also, as seen below, the order of the bisection algorithm is actually somewhat less than that of the one-pass algorithm.

The bisection algorithm for (1) proceeds as follows. Lower and upper limits f^- and f^- are established (at worst these could be 0 and 127, respectively), and the test value z is the midpoint of the interval $[f^-, f^-]$. A path seam of cost z or less is sought. If one is found, then its cost becomes f^-; while if none is found, f^- is set to $z + 1$. The number of iterations cannot exceed the logarithm to the base two of the range of the initial interval. It follows that, in our case, the number of iterations cannot exceed seven.

Table II
Computational Results of the Bisection Algorithm

Type	Rows	Cols	Time	Iter	Opt	No.
Sim	250	250	3	2.4	9.6	40
Real	500	500	10	5.3	25.3	4
Sim	500	500	11	3.2	9.8	20
Real	1000	1000	63	5.8	18.5	4
Sim	1000	1000	70	4.3	12.5	20
Real	2000	2000	383	6.3	15.5	4
Sim	2000	2000	259	4.6	13.7	20
Real	3000	4000	1144	6.4	16.0	4

Better initial bounds than 0 and 127 are generally easier to come by. Since the path must visit every row, it is valid to let

$$f^- = \max_{i=1}^{m} \min_{j=1}^{n} d_{ij}.$$

The upper bound f^- can be the objective value of any solution. We let f^+ be the largest-cost pixel visited by a simple greedy heuristic. In particular, the heuristic begins at any lowest-cost element of the first row and then visits any lowest-cost neighbor in any of the four directions except upwards until the last row is encountered. For the example of Section 1.3, the bisection algorithm initially has $f^- = 2$ and $f^+ = 11$. Then z takes on the values 6, 8, and 7, and the only solution is found at $z = 8$.

It is not hard to show that the complexity of each iteration is $O(mn)$. Since there can be no more than seven iterations, the overall complexity of the algorithm is $O(7mn)$, which is actually lower than that of the one-pass variation of the Dijkstra algorithm. This result takes advantage of the observation that it is never necessary to take into account the cost of pixel (i, j), except to determine whether or not $d_{ij} \leq z$.

2.1. Computational Results

The bisection algorithm has been coded in FORTRAN and is running on the VAX 6410 at ICC. In order to get a sense of the performance of the algorithms in a controlled fashion, many test problems have been run. We summarize the most informative results in Table II. Some are from real problems; these are supplemented with larger numbers of problems generated by a simulator developed at ICC to approximate real difference matrices.

The entries in Table II indicate problem type, size, average times to the nearest second, average number of iterations, average optimal solution value, and number of problems of this type. Not surprisingly, the time to solve an m by n problem is more or less $O(mn)$. The number of iterations shows minimal growth with problem size, which is also not surprising. The cost of the optimal solution for simulated problems will clearly be affected by the parameters used in problem generation so that this value is less interesting here than in the experience with real problems. It is comforting, however, that except for the four smallest real problems, all seams have cost less than 20, thus falling within the range in which invisibility is virtually guaranteed. (Intuitively, for a given type of data, the absence of a low-cost seam is most likely to occur in small problems because relatively few high-cost pixels might be able to block all low-cost seams.)

3. IMPLEMENTATION AND IMPACT ON ICC

3.1. Implementation

The bisection algorithm of Section 2 was installed at ICC in May of 1995. It has completely replaced the operator for drawing seams and is performing so well that virtually no seam has ever been rejected, and the seam-drawing process is almost completely automated. Only two steps of the process now require human intervention. After initial smoothing the operator chooses a set of about 1,500 columns through which the seam will pass. The choice is based on a quick sampling of the costs to find a contiguous column set that seems to have few dangerous areas and many low costs. It typically takes about one minute to make that selection.

When the operators did this same selection in the manual process, the initial set would be based on a combination of natural borders and dangerous areas. The former are no longer taken into account, and there now is almost never a need for the operator to revise the initial selection. The algorithm is run at night, and the next morning the operator spends about three minutes checking the final quality of the product.

3.2. Impact on ICC

The total human time needed to produce a map that requires mosaicking has been reduced from about an hour to a few minutes. As a consequence, virtually all the work can be done at night by the computers, and throughput is increased enormously. ICC now has the capability to produce perhaps 15 times as many such maps without an increase in manpower. The end result is that ICC can now compete for much larger projects than it has ever handled in the past, including those requiring the production of thousands of photographic maps.

Substantial cost savings have been realized from the automation of the mosaicking process. Approximately one person-hour per seam is saved, freeing the operators to work on other tasks. The resulting cost reduction from 1996 to 1997 has been estimated to exceed $40,000. In addition, because there is no longer any need to make it easier for an operator to find a good seam, ICC has reduced the overlap between consecutive photos from 80 percent to about 60 percent, on average. This means that fewer photos have to be taken, resulting in an estimated $50,000 cost reduction in the first year (savings as a result of fewer flights and less film). It also means that fewer total seams need to be drawn, which is a second reason for the increased throughput. Of course the total annual savings in the future will depend on the number of seams

produced. The overall cost of producing a map that requires mosaicking has been reduced by about 50 percent on average.

The major impact has been a substantial increase in the value of contracts signed to produce photographic maps. In 1997 ICC has thus far signed contracts to make 1,500 orthophotos (out of 5,000 single photos) at a value of about $4 million. That is an increase of about 200 percent over the total value of similar contracts for 1996. The primary reasons for this success are the increased production capability and the cost savings realized from the automation of the production process, which allow ICC to reduce its bid prices by about 12.5 percent.

4. A MODEL FOR THE HOLE PROBLEM

4.1. An Overview

This section deals with a variation of the seam-drawing problem of Section 2. It occurs when part of a photo (perhaps the most recently taken) is unusable due to cloud cover, shadow on the side of a mountain, or any of a number of other causes. We call such an area a *hole*. Then, in seam-based mosaicking, it is necessary to cut out the hole and replace it with the same area from a second photograph. An application that is solved without seams can be found in Soofi et al. (1991). Their algorithm simply tried to determine if each pixel in a given photo represented a cloud. If so, the same pixel in the other photo would be used.

Of course, there could be more than one hole. One area of northern Venezuela is so cloudy that an acceptable photo has never been taken. The best current photo has many small white clouds that have to be cut out in a systematic way. Here we deal only with the single-hole problem and leave the multiple-hole problem for future research.

Hundreds of instances of this hole problem occur yearly at ICC. For now the hole problems are solved as if they were path seam problems, with the assumption that the pixels within the hole will have high costs so that the optimal seam will avoid the hole. These solutions, however, may be less than satisfactory if there really is a primary photo that is better (perhaps more recent) than the other and it is desired to use that primary photo everywhere except for the hole. Thus we are motivated to develop separate models and algorithms for these problems.

The algorithm of Section 5 has not yet been implemented at ICC for two reasons. First, ICC has an adequate, if not ideal, method for solving the problem now. Second, a fast, efficient method for determining the pixels that make up the hole has not yet been developed. This step is needed as a preprocessing step for the algorithms of the next section. ICC is currently changing its computer system and prefers to delay implementation, including the preprocessing step, until the new computer system is in place.

Table III
A Counterexample to the Intuitive Definition of Surrounding Cycle

*	*	*	*	*			
*				*			
*		*	*	*			
*		*	x	x	*	*	*
*		*	x	x	*		*
*		*	*	*	*		*
*							*
*	*	*	*	*	*	*	*

4.2. Cycle Seams

To solve the hole problem we define a *cycle seam* that serves the same purpose as the path seam of the previous model. The cycle seam *surrounds the hole*, and the area inside it is taken from a second photo. While there are no given starting and ending points, it is inherently desirable that the cycle be relatively close to the hole. Here we assume that this is implemented by simply redefining the domain of the problem to be those cells within an acceptable distance of the hole.

The absence of a given vertex that must lie in the cycle is one of the characteristics distinguishing this problem from others in which optimal cycles are required, e.g., traveling salesman problems. Another unique feature is that the cycle must have a predetermined relationship to a set of vertices, that is, it must encircle the hole. Both features make modeling and solution more complex than for the path seam problem.

Given a simple cycle it is easy to see pictorially if it is a cycle seam. If the problem were defined in the plane instead of on a grid network, so that movement in any direction from a point were possible, then a cycle is a cycle seam if and only if every path from a point in the hole to the border of the photo passes through the cycle. Here, because only horizontal and vertical movement is possible, it is easy to construct counterexamples to that definition; that is, there may be cycles that are not cycle seams but that still block every path from the hole to the border of the photo. An example is Table III, where elements of the hole are indicated by x and elements of the seam by stars.

We offer the following definitions (see Figure 4), which lead to the algorithm of the next section. Denote the hole by H and let T be a segment of a column from the first row to the first intersection of the column with H. Formally, let j^* index any column such that there exists any row i' such that $(i', j^*) \in H$. Denote the column segment $\{(1, j^*), \ldots, (i, j^*), \ldots, (k, j^*), \ldots, (i^*, j^*)\}$ by T, where $T \cap H = \phi$ but $(i^* + 1, j^*) \in H$. A simple path $P = \{(i, j^*), (i, j^* + 1), \ldots, (k, j^* - 1), (k, j^*))$ is *surrounding* if $P \cap T = \{(i,$

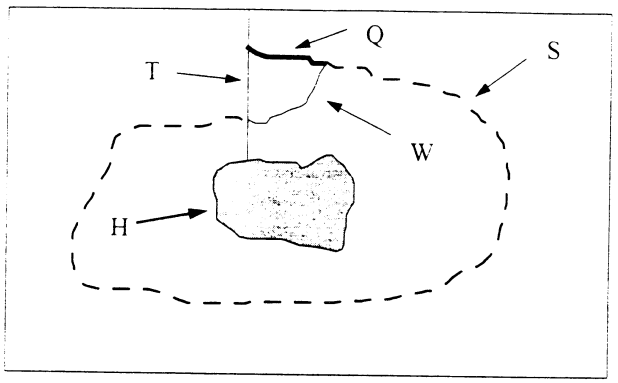

Figure 4. Two surrounding paths, (Q, S) and (W, S).

$j^*)$, $(k, j^*)\}$ and $P \cap H = \phi$. Thus (Q, S) and (W, S) in Figure 4 are both surrounding paths. Intuitively a surrounding path leaves T to the right, comes back to T from the left, and surrounds without intersecting H.

If, as in (W, S) of Figure 4, the end points i and k of a surrounding path are the same point, then the surrounding path is a cycle; otherwise, i and k must be connected by another path. A simple path C is *connecting* if C begins and ends at pixels of T and no subpath of C is surrounding. For example, the path $C = (Q, W)$ in Figure 4 is connecting. A connecting path is *degenerate* if it is the null set. Then a *cycle seam* CS is the symmetric difference of a surrounding path P and a connecting path C where P and C have the same end points. That is, $CS = (P \cup C) - (P \cap C)$. Thus (W, S) in Figure 4 fits that definition either by using $P = (Q, S)$ and $C = (Q, W)$ or $P = (W, S)$ and C degenerate. Note that the condition that no subpath of C be surrounding is needed to avoid creating a cycle, such as that of Figure 5, that does not surround H.

4.3. The Cycle Seam Model

Let the cost of any cycle seam CS be given by

$$f(CS) = \text{maximum } d_{ij}, \quad (i, j) \in CS.$$

Then the cycle seam problem is

$$\text{minimize } f(CS), \quad CS \text{ is a cycle seam.} \tag{6}$$

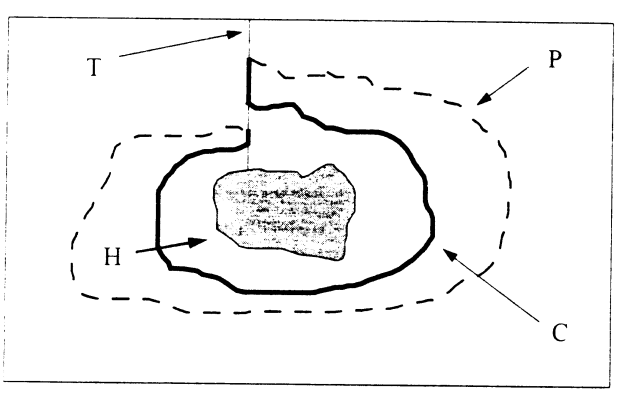

Figure 5. A simple path C that is not a connecting path.

Table IV
An Optimal Cycle Seam

11	15	18	16	3	7	8	6	8
13	21	7	11	10	6*	5*	3*	10
6	8*	4*	9*	7*	8*	13	6*	11
7	5*	23	x	x	x	15	5*	4*
12	4*	7*	x	x	x	23	17	2*
14	5	9*	7*	12	6	11	5*	3*
3	12	7	8*	2*	13	14	4*	15
4	7	10	5	1*	7*	5*	8*	4

For the example of Table IV, with the hole indicated by x, an optimal cycle seam of cost 9 is indicated by the starred pixels.

5. AN ALGORITHM FOR THE HOLE PROBLEM

We present a labeling algorithm for (6) that, like the algorithm for (1), is embedded in a bisection search. It is more complex than the algorithm of Section 2 for the reasons given in Section 5.2.

5.1. Overview and Definitions

As in Section 2, let the trial objective function value within the binary search be z. Set $d_{ij} = 128$ for every $(i, j) \in H$, so that in the labeling algorithm no element of the hole will ever be labeled. Then define *z-surrounding paths*, *z-connecting paths*, and *z-cycle seams* to be those which contain no pixel with cost greater than z. Rows s and t of T are *z-equivalent* if there is a z-connecting path between them. For instance, if $j^* = 5$ in the example, rows one and three are z-equivalent for $z \geq 8$. It follows that if there is a z-surrounding path between two z-equivalent rows then there exists a *z-cycle seam.*

Surrounding paths are the building blocks of cycle seams in the sense that they are needed to ensure that the hole is surrounded by the cycle. On the other hand, z-surrounding paths can be made into z-cycle seams only if their end points are z-equivalent. This leads to a labeling algorithm with two stages.

The first stage, which is executed for every trial value of z, simultaneously builds the classes of z-equivalent rows and attempts to find z-surrounding paths. The second stage is executed only once, after the optimal value of z has been determined, and is used to trace the optimal cycle seam. This stage is needed since the labels of the first stage, as opposed to those of most labeling algorithms, e.g., the algorithm of Section 2, serve only to determine if a z-cycle seam exists. Since the algorithm has only been tested on simulated data we will limit ourselves to sketching the first stage in some detail and the second phase, which is much more straightforward, in less detail.

5.2. The Labeling Algorithm

5.2.1. The Labeling in Stage 1. In Stage 1 any vertex that can be reached by a simple z-path from an element of T is labeled once. The label of any vertex v (it will be simpler in this section to refer to a vertex as v rather than (i, j)), has two components, $y_1(v)$ and $y_2(v)$. If $y_1(v) = t$ then the origin of the z-path to v is $t \in T$, while $y_2(v) = l$ or r depending on whether the z-path to v departed t toward the left or the right.

Every labeled vertex v is *scanned* by examining both its unlabeled and labeled neighbors. Let v' be any neighbor of v. If v' is unlabeled it can be given the labels $(y_1(v), y_2(v))$ as long as $d_{v'} \leq z$; otherwise, if v' is labeled, two possible cases can occur. These are:

(i) the second components of the two vertices are the same, i.e., $y_2(v) = y_2(v')$. Then the rows $y_1(v)$ and $y_1(v')$ are z-equivalent since the union of the two paths to v and v' is a z-connecting path;

(ii) the second components of the two vertices are different, e.g., $y_2(v) = l$ and $y_2(v') = r$. Then there exists a z-surrounding path that is the union of the two paths to v and v'.

5.2.2. A Skeleton of the Stage 1 Algorithm. For any z, a skeleton of the algorithm to determine if a z-cycle seam exists is given below.

The Algorithm

Initialization: Record as z-equivalent all neighboring vertices $v \in T$ such that $d_v \leq z$. Label all vertices v' such that $v' \notin T$, v' is a neighbor of some $v \in T$, and $d_v \leq z$ and $d_{v'} \leq z$, with (v, l) or (v, r) depending on whether v' lies to the left or right of v.

Iterative Step:

```
While there is a labeled unscanned vertex
  select a labeled unscanned vertex v
  scan v:
    For any neighbor v' of v:
      If v' is labeled and y2(v) = y2(v') then
        record that rows y1(v) and y1(v') are
        z-equivalent.
      If v' is labeled and y2(v) ≠ y2(v') then
        record that there is a z-surrounding path
        from y1(v) to y1(v').
      If v' is unlabeled and d_v' ≤ z then
        label v' with (y1(v), y2(v)).
```

Final step: If a z-surrounding path has been found between z-equivalent origins then a z-cycle seam exists.

5.2.3. Stage 2—Tracing the Optimal Cycle. The labels of Stage 1 serve only to determine if a z-cycle seam exists but do not enable one to trace the seam. Once it has been determined that there is an optimal cycle seam of cost z^*, which begins and ends at vertex v^* of T, it is a simple matter to enter a second stage of labeling in which the cycle is traced. The details are omitted.

Table V
Computational Results for the Cycle Seam Problem

Rows	Cols	Time	Iter	Opt	No.
250	250	10	4.6	41.3	20
500	500	68	4.8	21.9	20
1000	1000	246	4.5	12.2	20

5.2.4. Complexity. It is not hard to show, again using the above observation on the maximum cost, that the complexity of each iteration is the same as for the path seam problem, namely $O(mn)$. Again, since there can be no more than seven iterations, the overall complexity of the algorithm is $O(7mn)$.

5.2.5. Computational Results. Computations were done on the VAX 6420. Problems were generated via a simulator developed at ICC to approximate real hole problems. We summarize the most informative results in Table V.

As for the path seam problem, the required computation time seems to grow as $O(mn)$, and the number of iterations remains more or less constant. It should also be observed that for smaller problems the optimal values are somewhat larger for the cycle seam problem than for path seam problems of the same size. This may have to do with the parameters of the problem generator, although it may also have to do with the fact that cycle seams are more restricted than path seams. It is encouraging, and not surprising, that the optimal objective values decline as the problem size increases and that those in the last row are well within the range of "invisibility." Of course it remains to be seen whether this algorithm will consistently produce invisible seams, as has the algorithm for the path seam problem.

6. CONCLUSIONS AND FUTURE RESEARCH

Models and algorithms have been presented for seam-based mosaicking of aerial photographic maps. The path seam algorithm has been implemented at ICC. The immediate impact on ICC has been a dramatic increase in the value of contracts received in 1997 because of cost savings and increased throughput in the production process. Because of the automation of that process, they are now ready to bid on even larger projects involving the production of thousands of maps, which they could not consider before. We expect that the cycle seam algorithm will be implemented in the future with comparable effect.

If ICC, or other companies, face problems larger than the current ones it might not be possible to deal with the entire matrix (d_{ij}) in memory at one time. In that case, other data-handling techniques could be used. For instance, one could use a compact data structure, taking advantage of the fact that in real problems, neighboring elements in a given row are likely to have similar costs, and there could be strings of identical elements. For instance, the data below can be represented by the two vectors (765284) and (692579), where the second vector indicates in which column of the matrix each string ends.

7	7	7	7	7	7	6	6	6
5	5	2	2	2	8	8	4	4

This data structure, depending on the data, may provide dramatic savings in storage. On the other hand, such a data structure also makes labeling somewhat more complex. To implement the compact data structure efficiently, additional information is needed for identification of the neighbors, and especially the vertical neighbors, of a given node in the original matrix.

In simulated problems the storage savings using compact data were dramatic, often approximating an 80-percent reduction in storage. Although this seemed encouraging, for real problems we found an average savings of only about 5 percent. Examination of the data indicated that this occurred because, with real data, neighboring cells tend to have costs that, while similar, are not identical. This difficulty could be alleviated by compacting the data more crudely, since much greater efficiencies could potentially be achieved if the data were grouped into classes of, say, 0–4, 5–9, etc. at a cost of a possibly degraded solution to the original problem. Future research will investigate the trade-off between crudeness of compacting and the visibility of the resulting seams.

ACKNOWLEDGMENTS

The authors wish to thank the area editor, associate editor, and referees for their comments which have led to substantial improvements in the presentation of this paper. The collaboration of the authors has been partially supported by Grant C2071 from Institut Cartografic de Catalunya to Universitat Politecnica de Catalunya. Research of the second author was supported in part by Grant SAB94-0115 from the Spanish Interministerial Commission of Science and Technology.

REFERENCES

AHUJA, R. K. ET AL. 1990. "Faster Algorithms for the Shortest Path Problem." *J. Association for Computing Machinery*, **37,** 213–223.

BERMAN, O. AND G. Y. HANDLER. 1987. "Optimal Minimax Path of a Single Service Unit on a Network to Nonservice Destinations." *Transp. Sci.* **21,** 115–122.

DIJKSTRA, E. W. 1959. "A Note on Two Problems in Connexion with Graphs." *Numerische Mathematik* **1,** 165–175.

EDMONDS, J. AND D. R. FULKERSON. 1968. "Bottleneck Extrema." RM-5375-PR, The Rand Corporation, Santa Monica, CA.

Erdas Imagina. 1996. Erdas, Inc., Atlanta, GA.

FERNANDEZ, E., R. GARFINKEL, AND R. ARBIOL. 1996. "Mosaicking of Aerial Photographic Maps via Seams Defined by Bottleneck Shortest Paths." Working Paper DR 96/13, Department of Statistics and Operations Research, Polytechnic University of Catalonia, Spain.

GARFINKEL, R., A. NEEBE, AND M. R. RAO. 1977. "The m-Center Problem: Minimax Facility Location." *Mgmt. Sci.* **23,** 1133–1142.

JENSEN, J. R. 1986. *Introductory Digital Image Processing*. Prentice-Hall, New York.

MILGRAM, D. L. 1975. "Computer Methods for Creating Photomosaics." *IEEE Trans. Comput.* **C-24,** 1113–1119.

MILGRAM, D. L. 1977. "Adaptive Techniques for Photomosaicking." *IEEE Trans. Comput.* **C-26,** 1175–1180.

SHIREN, Y., L. LI, AND G. PENG. 1989. "Two-Dimensional Seam-Point Searching in Digital Image Matching." *Photogrammetric Engineering and Remote Sensing*, **55,** 49–53.

SOOFI, K. A., R. S. U. SMITH, AND R. SIREGARD. 1991. "A Planimetrically Accurate SPOT Image Mosaic of Button Island," *Photogrammetric Engineering and Remote Sensing*, **57,** 1217–1220.