Registration Algorithms for Geophysical Maps

Behzad Kamgar-Parsi
Advanced Information Technology Branch
Naval Research Laboratory
Washington, DC 20375

Behrooz Kamgar-Parsi
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory
Washington, DC 20375

Abstract
Data registration is a recurring problem in making geophysical maps. When we want to merge two partially overlapping data patches to make an accurate, seamless mosaic, the patches must be registered (aligned, matched, fitted) relative to each other. The registration problem can be alleviated by the use of widely available GPS systems. However, for surveys conducted under water, where GPS access is unavailable or limited, errors in the true coordinates of the collected data can be significant.

An application of accurate geophysical maps is in non-traditional (map based) navigation, where a vehicle measures geophysical properties and attempts to locate itself on existing maps [10]. The geophysical map must have a sufficient number of distinctive features or large gradients to be useful for navigation. The Earth’s gravitational field in many regions has sufficiently large gradients that it can be used for navigation, particularly as a means of bounding the errors caused by INS drifts [3]. Gravimeters are already on-board certain submarines and used in underwater navigation [8]. Our motivation in this paper is to develop registration algorithms for making accurate gravity maps for use in underwater navigation.

To register two partially overlapping data patches, one must first identify corresponding features in the patches and then transform (rotate and translate) one patch so as to align the corresponding features. Corresponding features, or matching primitives, should ideally be distinctive, such as peaks, ridges, etc. However, in many surveys there is a dearth of such distinctive features. In a method developed by Kamgar-Parsi et al. [7] for registering swath bathymetry data, this difficulty is overcome by using iso-value contours as the matching primitives. The sensitivity of current gravimeters are such that gravity maps are even smoother than the seafloor topography, hence we believe this method is also most suitable for registering gravity surveys. An advantage of using iso-contours as matching primitives is that an unlimited number of them may be extracted from the data patches. The approach proposed in [7] involves several modules. In this paper, we present new algorithms that improve upon those modules used in [7]. The improvements include extracting contour shapes, an efficient approach to contour matching, and closed-form solutions for the optimum transformation, which is critical in finding correspondences as well as in making the final match. These improvements make the registration more accurate and computationally faster. Below, we discuss the registration approach and the recent advances, and demonstrate its effectiveness by applying it to a synthetic data set that mimics a featureless gravitational field.

1 Introduction
Data registration is a recurring problem in making geophysical maps. When we want to merge two partially overlapping data patches to make an accurate, seamless mosaic, the patches must be registered (aligned, matched, fitted) relative to each other. The registration problem can be alleviated by the use of widely available GPS systems. However, for surveys conducted under water, where GPS access is unavailable or limited, errors in the true coordinates of the collected data can be significant.

An application of accurate geophysical maps is in non-traditional (map based) navigation, where a vehicle measures geophysical properties and attempts to locate itself on existing maps [10]. The geophysical map must have a sufficient number of distinctive features or large gradients to be useful for navigation. The Earth’s gravitational field in many regions has sufficiently large gradients that it can be used for navigation, particularly as a means of bounding the errors caused by INS drifts [3]. Gravimeters are already on-board certain submarines and used in underwater navigation [8]. Our motivation in this paper is to develop registration algorithms for making accurate gravity maps for use in underwater navigation.

To register two partially overlapping data patches, one must first identify corresponding features in the patches and then transform (rotate and translate) one patch so as to align the corresponding features. Corresponding features, or matching primitives, should ideally be distinctive, such as peaks, ridges, etc. However, in many surveys there is a dearth of such distinctive features. In a method developed by Kamgar-Parsi et al. [7] for registering swath bathymetry data, this difficulty is overcome by using iso-value contours as the matching primitives. The sensitivity of current gravimeters are such that gravity maps are even smoother than the seafloor topography, hence we believe this method is also most suitable for registering gravity surveys. An advantage of using iso-contours as matching primitives is that an unlimited number of them may be extracted from the data patches. The approach proposed in [7] involves several modules. In this paper, we present new algorithms that improve upon those modules used in [7]. The improvements include extracting contour shapes, an efficient approach to contour matching, and closed-form solutions for the optimum transformation, which is critical in finding correspondences as well as in making the final match. These improvements make the registration more accurate and computationally faster. Below, we discuss the registration approach and the recent advances, and demonstrate its effectiveness by applying it to a synthetic data set that mimics a featureless gravitational field.

2 Registration Algorithms
Suppose we have two data patches (denoted by A and X) that may have an overlap, but the true position and orientation of the patches relative to each other are not known. We wish to find the transformation that moves patch X and aligns it as best as possible with patch A. (Note that we register X relative to A. To obtain the absolute map we need to peg down patch A by knowing the exact coordinates of at least two points in A.) The approach proposed by Kamgar-Parsi et al. [7] (see also [4]) has the following steps: (1) grid the data; (2) extract iso-contours; (3) find corresponding portions of iso-contours; and (4) determine the best transformation. After the best transformation is obtained, the data patches can be correctly merged and regridded to create a mosaic. In this paper we present improvements in the way steps 2, 3, and 4 are performed.

2.1 Gridding the Data
For efficient processing and storage we first place a regular grid (with square cells) over the data patch and resample. The grid size is selected such that the number of
grid points is roughly equal to the number of data points. The resampling can be done in various ways. We have used a weighted averaging approach. In this approach we find the closest data points (with at least one point in each quadrant to ensure interpolation, rather than possible extrapolation) to grid point \( i \) and then estimate the field value, \( g_i \), at this point from

\[
g_i = (1/A) \sum_{j \in N_i} f_j / d_{ij}^\nu,
\]

where \( N_i \) is the neighborhood of the grid point, \( f_j \) is the field value at data point \( j \), \( d_{ij} \) is the distance between the grid point and data point \( j \), \( A = \sum_{j \in N_i} (1/d_{ij}^\nu) \) is the weight normalization factor and \( \nu = 1 \) or \( 2 \).

Another common resampling technique is by the Delaunay triangulation of the data points. In this approach the field is assumed to be planar in each triangle. To estimate the field at a grid point, the triangle in which it lies is found and its value is computed from the planar grid points is roughly equal to the number of data points.

The resampling can be done in various ways. We have found the closest data points (with at least one point in the neighborhood of the grid point) to grid point \( i \) and then estimate the field value, \( g_i \), at this point from

\[
g_i = (1/A) \sum_{j \in N_i} f_j / d_{ij}^\nu,
\]

where \( N_i \) is the neighborhood of the grid point, \( f_j \) is the field value at data point \( j \), \( d_{ij} \) is the distance between the grid point and data point \( j \), \( A = \sum_{j \in N_i} (1/d_{ij}^\nu) \) is the weight normalization factor and \( \nu = 1 \) or \( 2 \).

2.2 Extracting Contours

Extracting contours of constant field value, or iso-contours, in a square grid is straightforward. However, there are certain situations where following an iso-contour becomes ambiguous. This ambiguity was previously dealt with by adopting a convention, which can result in the wrong continuation of the contour, and an incorrect shape. Since the registration algorithm uses contour shapes this may cause errors. In Appendix A, we present an approach for contour extraction that removes this ambiguity in a manner consistent with the linear interpolation technique used in contour extraction.

Note that converting the data points to a regular grid is done for ease and efficiency of processing and map storage. This resampling is not necessary for iso-contour extraction. The data points may be triangulated and used directly for contour extraction. The triangulation of an irregular set of points, however, is not unique. Delaunay triangulation, which favors fat triangles, is the most widely used approach [9]. The computational complexity of Delaunay triangulation is \( O(n \log n) \), while the complexity of gridding is \( O(n) \), where \( n \) is the number of data points.

2.3 Finding Correspondences

This step is a very challenging and crucial part of the registration problem. Suppose iso-contours \( C_A \) and \( C_X \) with value \( v \) are extracted from the two patches, \( A \) and \( X \). We have to determine which portions, if any, of these iso-contours overlap with each other, i.e. the corresponding portions. Three cases may arise: (1) the two contours have no overlapping portions; (2) they fully overlap, i.e. the shorter contour lies completely on the longer one; and (3) the two contours partially overlap. To find the corresponding portions, we have to find the pieces of \( C_A \) and \( C_X \) that have the best match. If the mismatch (for the best matching pieces) is below a tolerance, then we have found the corresponding portions. Note that we do not know either the length or the locations of the best matching portions. This task is further complicated by measurement noise and contour quantization. Therefore, contour overlaps must be sufficiently long to be reliable.

To find the best matching portions, we hypothesize an overlap length, \( l \), and match all \( l \)-long pieces of \( C_X \) to all \( l \)-long pieces of \( C_A \). We do this for many overlap lengths. Those \( l \)-long pieces that have the best mismatch measure are the likely corresponding portions. (See [5][6] for an algorithm.) To find the best match between two contours we have to define a distance measure between them, find the transformation that minimizes it, and compute the mismatch measure. The best transformation, and the technique for finding it, depends on the definition of the distance measure. In this paper, we define a Euclidean distance square between the two contours to be the distance measure:

\[
M(C_A, C_X) = \frac{1}{|lnl|} \int_0^l du \|P_A(u) - P_X(u)\|^2.
\]

Here \( C_A \) and \( C_X \) denote the corresponding portions of the contours, \( l \) is the overlap length, \( u \) is a distance parameter along the corresponding portions, and \( P_A(u) \) and \( P_X(u) \) are the corresponding points on the two contours, i.e. points with the same distance from the start points of \( C_A \) and \( C_X \). The normalization factor \( |lnl| \), rather than \( l \), favors longer pieces. To obtain the best match we have to minimize \( M(C_A, T_{C_X}) \) over all transformations \( T \). In Sec. 3, we derive a closed-form solution for the optimum transformation.

We note that this is an improvement over the technique used in [7], where orientational and translational mismatches are calculated separately and a heuristic is used to find the best match.

2.4 Finding the Transformation

When a sufficiently large number of corresponding iso-contours are obtained, we compute the distance measure between the two patches \( A \) and \( X \):

\[
M(A, X) = \sum_v M(C_A, C_X).
\]

The transformation \( T \) that minimizes \( M(A, TX) \) is then the transform that gives the best alignment of \( A \) and \( X \). A closed-form solution, similar to that given in Sec. 3, can be derived for the optimum \( T \).

To obtain the best transformation, [7] used a small angle approximation to avoid iterative techniques and numerical instabilities. Such an approximation is not needed in the present approach.

3 Matching Equal-Length Curves

Suppose we have two polygonal arcs (piecewise linear curves) denoted by \( A \) and \( X \), which have the same total length, \( l \). Note that extracted iso-contours are piecewise linear. Each arc is directed, that is, one end is labeled as the start and the other as the end. In Fig. 1, for example, \( A \) is composed of four line segments while \( X \) has six segments, and the arrows indicate the arc directions. Suppose we want to rigidly transform (translate and rotate) arc \( X \) so as to get the best match with arc \( A \). Best in the sense of a distance measure between the two arcs. Although different distance measures are used in the computer vision literature for matching (see [6] for discussion and references), the most common is the \( L_2 \) norm distance measure which we use in this paper. The distance measure, \( M(A, X) \), between the two arcs
is thus the sum of the Euclidean distance squares of all the corresponding points on the two arcs. Corresponding points on $A$ and $X$ are the points that have the same distance from the start point of each arc. If we parametrize the arc lengths by $u$, then $M(A, X) = \int_0^1 D^2(u)$, where $D(u)$ is the Euclidean distance between the corresponding points a distance $u$ from the start points. We leave out the normalization factor $1/\ln l$ in the distance measure, because for arcs with constant length it does not play any role. The best match is obtained by transforming $X \rightarrow TX$ and minimizing $M(A, TX)$ with respect to the transformation $T$. The minimum distance measure is called the mismatch measure. It is the optimum transformation that we need for registering iso-contours.

To compute the distance measure we may represent arcs $A$ and $X$ by a large number of equally spaced points along their lengths, and then add the distances of the corresponding points. This representation, however, can be computationally expensive. The most efficient representation is achieved by decomposing the two arcs into corresponding line segments of equal lengths \cite{5,6}. In this representation $A = \{A_1, \ldots, A_N\}$ and $X = \{X_1, \ldots, X_N\}$, where $A_n$ and $X_n$ are two corresponding line segments that have the same length $l_n$, with $l = \sum_{n=1}^N l_n$. In Fig. 1, for example, the arcs are represented by eight corresponding line segments of equal lengths (the dotted lines indicate the start and end points of each segment). Below, we first describe the decomposition algorithm, and then we calculate the distance measure and derive the closed-form solution for the transformation that gives the best match between the two arcs.

### 3.1 Decomposition Algorithm

Suppose arc $A$ is composed of $m$ successive line segments with lengths $(s_1, s_2, \ldots, s_m)$, and arc $X$ is composed of $n$ successive line segments with lengths $(t_1, t_2, \ldots, t_n)$, where $s_1 + s_2 + \cdots + s_m = t_1 + t_2 + \cdots + t_n$. The decomposition proceeds as follows. Compare $s_1$ and $t_1$; the length of the first segment is $l_1 = \min(s_1, t_1)$. Suppose $s_1 < t_1$, thus the first segment has length $l_1 = s_1$. The remainders of arcs $A$ and $X$ are composed of $(s_2, \ldots, s_m)$ and $(t_1, t_2, \ldots, t_n)$, respectively, where $t_1' = t_1 - l_1$. Next, compare $s_2$ and $t_1'$. The length of the second segment in the decomposition is $l_2 = \min(s_2, t_1')$, and so on. The number of corresponding line segments of equal length, $N$, depends on the relative shapes of the two arcs, and ranges between the best case $N = \max(m, n)$ and the worst case $N = m + n - 1$. As an example, in Fig. 1, arc $A$ has $m = 4$ line segments, while arc $X$ has $n = 6$ segments. The decomposition yields $N = 8$ corresponding line segments.

### 3.2 Distance Measure

Here we discuss the more general problem of matching 3D curves; 2D curves which we actually use for map registration in this paper are a special case. We represent the line segment $A_n$ with its center point $a_n$, the unit vector along its direction $b_n$, and its length $l_n$, or $A_n = (a_n, b_n, l_n)$. The end points of segment $A_n$ are given by $a_n \pm l_n b_n/2$. Similarly, we denote $X_n = (x_n, y_n, l_n)$, where $x_n$ is its center point, $y_n$ its direction, and $l_n$ its length. As we mentioned earlier, we define the distance measure between directed line segments $A_n$ and $X_n$, whose lengths are equal, to be the sum of square of distances of their corresponding points. If we use the variable $u$, where $-l_n/2 \leq u \leq l_n/2$, to parametrize $A_n$ and $X_n$, then the coordinates of corresponding points on $A_n$ and $X_n$ are given by $a_n + u b_n$ and $x_n + u y_n$. The square of the Euclidean distance between these corresponding points is $D^2(u) = \|a_n + u b_n - x_n - u y_n\|^2$. Thus, the distance measure between arcs $A$ and $X$, $M(A, X) = \sum_{n=1}^N \int_{-l_n/2}^{l_n/2} D^2(u)$, is given by

$$M(A, X) = \sum_{n=1}^N \left[ l_n \|a_n - x_n\|^2 + \frac{b_n}{6} (1 - b_n^T y_n) \right]. \quad (1)$$

Vectors are column matrices and the superscript $T$ denotes matrix transpose. We hold arc $A$ fixed while we transform arc $X$. Since rotation and translation are noncommuting operations the order in which they are performed must be specified. Here we first rotate $X$ and then translate it. We denote the rotation matrix by $R$, and the translation vector by $t = (t_1, t_2, t_3)^T$. Thus, to transform $X$ we do the following:

$$\begin{align*}
x_n & \rightarrow x_n + t + Rx_n, \\
y_n & \rightarrow y_n + Ry_n.
\end{align*} \quad (2)$$

The distance measure between $A$ and the transformed $X$ is obtained by substituting (2) into (1), that is,

$$M(A, TX) = \sum_{n=1}^N \left[ l_n \|a_n - (t + Rx_n)\|^2 + \frac{b_n}{6} (1 - b_n^T Ry_n) \right]. \quad (3)$$

Below, we derive closed-form solutions for the translation vector and the rotation matrix that minimize the distance
measure (3). Note that if we set all $a_i = 1$ and ignore the second term in (3), then it reduces to the problem of $N$ equally spaced corresponding points.

### 3.3 Translation Vector

To find the translation vector minimizing (3) we solve the set of equations $\partial M / \partial t_i = 0$ for the translation parameters $t_i$. These equations are linear in $t_i$ which are readily solved to give the translation vector:

$$ t = \tilde{a} - R\tilde{x}, $$

where

$$ \tilde{a} = \sum_{n=1}^{N} w_n a_n, \quad \tilde{x} = \sum_{n=1}^{N} w_n x_n, \quad w_n = l_n / \sum_{n=1}^{N} l_n. $$

The weight $w_n$ is the relative importance of the $n$-th line segment, and $\tilde{a}$ and $\tilde{x}$ are the centers of mass of $A$ and $X$, respectively. Eq. (4) expresses that by rotating $X$ first its best orientational alignment with $A$ is found, and then the center of mass of $X$ is moved to the center of mass of $A$. It can be proved that (4) gives the minimum of $M$ (rather than a maximum, a saddle point, etc.), since all the eigenvalues of the Hessian of $M$ are positive [5].

### 3.4 Rotation Matrix

By substituting (4) into (4) in (3), we obtain an expression for the distance measure that involves only the rotation matrix,

$$ M(A, X) = \sum_{n=1}^{N} l_n (|| \tilde{a}_n ||^2 + || \tilde{x}_n ||^2 + \frac{l_n^2}{\delta} ) 
- 2 \sum_{n=1}^{N} l_n (\tilde{a}_n^T R\tilde{x}_n + \frac{\delta}{12} b_n^T R\tilde{y}_n), $$

(5)

where

$$ \tilde{a}_n = a_n - \tilde{a}, \quad \tilde{x}_n = x_n - \tilde{x}. $$

In deriving (5) we used the property that the length of a vector is invariant under rotation. The first sum in (5) is a constant independent of the rotation matrix, therefore, to minimize $M(A, X)$ we must maximize the second sum. The rotation matrix can be calculated in closed-form. Below we give two approaches. One is based on singular value decomposition (SVD), and the other is based on the quaternion representation of rotations. Before we proceed we define the following $3 \times 3$ matrix which is needed in both methods,

$$ S = \sum_{n=1}^{N} l_n (\tilde{a}_n \tilde{x}_n^T + \frac{l_n^2}{12} b_n \tilde{y}_n^T). $$

Matrix $S$ is called the cross-covariance matrix.

In the SVD method, we compute the SVD of matrix $S$ (see [1]),

$$ S = U W V^T, $$

(7)

where $W$ is a diagonal matrix with non-negative elements, and $U$ and $V$ are orthogonal matrices. The rotation matrix that maximizes (7) can then be constructed from

$$ R = U V^T. $$

(8)

For more details we refer to [5].

The second method of finding the rotation matrix is to represent the rotation in terms of a unit quaternion. See [2] for a detailed discussion of quaternions. The unit quaternion $\hat{q} = (q_0, q_1, q_2, q_3)^T$ is a $4 \times 1$ matrix with unit norm, $|| \hat{q} ||^2 = \sum_{n=0}^{3} q_n^2 = 1$. A rotation through angle $\theta$ around the axis $\hat{v} = (v_1, v_2, v_3)^T$ may be represented by the unit quaternion $\hat{q} = (c, s v_1, s v_2, s v_3)^T$, where $c = \cos (\theta/2)$ and $s = \sin (\theta/2)$. The rotation matrix can be constructed from the unit quaternion by

$$ R = \left( \begin{array}{cccc} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 & 0 \\ 2q_1q_2 - 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & q_1q_3 + q_0q_2 & 0 \\ 2q_1q_3 + 2q_0q_2 & q_1q_2 - q_0q_3 & q_0^2 - q_1^2 - q_2^2 + q_3^2 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right). $$

(9)

The rotation matrix that maximizes the second sum in (5) can be obtained as follows. First, construct the $4 \times 4$ symmetric matrix

$$ \omega = \left( \begin{array}{cccc} s_{11} + s_{22} + s_{33} & s_{23} - s_{32} & s_{31} - s_{13} & s_{12} - s_{21} \\ s_{23} - s_{32} & s_{22} + s_{33} - s_{11} & s_{13} + s_{31} - s_{22} & s_{21} + s_{12} - s_{33} \\ s_{31} - s_{13} & s_{13} + s_{31} - s_{22} & s_{22} + s_{33} - s_{11} & s_{32} + s_{23} - s_{11} \\ s_{12} - s_{21} & s_{21} + s_{12} - s_{33} & s_{32} + s_{23} - s_{11} & s_{33} + s_{22} - s_{11} \end{array} \right), $$

where $S_{ij}$ are the elements of $S$ given in (6). Then compute the four eigenvalues of matrix $\omega$, which are real since $W$ is Hermitian. And finally calculate the eigenvector corresponding to the largest eigenvalue. This normalized $4 \times 1$ eigenvector is the unit quaternion that represents the rotation matrix. The eigenvalues and eigenvectors can be calculated in closed-form. See [5] for more details.

### 4 Experiments

Consider the synthetic gravity field shown in Fig. 2, which depicts a fairly smooth and featureless region. Selected iso-contours at regular field value intervals are shown. Suppose the regions inside the rectangular boxes are surveyed. Further suppose that coordinates of patch $A$ (the large box) are obtained accurately, while patch $X$ (the small box) coordinates are incorrect. See Fig. 3. We wish to register $X$ with respect to $A$.

First, we grid the patches: patch $A$ is a $59 \times 45$ grid, while $B$ is a $31 \times 59$ grid. Then we extract several iso-contours in each patch: the iso-contours used in registration are shown in Fig. 3. Then we find the corresponding portions of the iso-contours when the two patches actually overlap. Finally, after we find the optimum transformation, we rotate and translate $X$ to line up with $A$. The result is shown in Fig. 4. The transformation obtained from the registration algorithm is very close to the transformation we used originally to move $X$. This is an ideal, noiseless experiment; for real data, of course, we would not expect to get such a good alignment.

As we mentioned in Sec. 2, finding the corresponding portions of iso-contours is the most challenging part of the registration algorithm and should be handled with care. Even this idealized case presents difficulties, which
Figure 2: A synthetic gravity field.

Figure 3: Data patches before registration.

we will discuss below. In Fig. 5, we plot the best mismatch measure (BMM) for a given overlap length. The unsmooth nature of these plots are due to quantization errors. The plot for contour 1 is shown in Fig. 4 is typical for contours that have full overlap (contour 1 in $A$ is fully contained in contour 1 in $X$): the BMM is always small, and degrades slightly as the overlap length is decreased (this is due to the $\ln$ normalization used in the mismatch measure). The plots for contours 2 and 3 (also shown in Fig. 4) are also typical of contours that have partial overlap: the BMM decreases rapidly as the overlap length is decreased until it reaches a small value and stays constant (with a slight increase as the overlap length is further decreased). The overlap length below which the mismatch does not improve shows the corresponding portions of the iso-contours.

For iso-contours that do not have any overlapping portions, the BMM typically remains large for all hypothesized overlap lengths. However, for smooth iso-contours, which we are likely to encounter in gravity maps, this may not always be the case. Suppose we reverse the contours. The BMM for any hypothesized overlap length is expected to be large. In Fig. 6, we plot the BMM for the reversed contours. For contour 1 the BMM is off the scale, however, for contours 2 and 3 they are comparable to the true matches. Thus, we cannot always infer correspondences based on the mismatch measure alone. Therefore, we also look for the consistency of transformation parameters. For all matches representing true correspondences, the transformation parameters are nearly identical. While accidental matches that can potentially lead to false correspondences yield widely differing transformation parameters. During the process of finding correspondences, we accumulate histograms of transformation parameters. The dominant peak of each histogram gives the true transformation parameter. By comparing the transformation parameters with the histogram peaks, we can eliminate false correspondences.

5 Discussion

The approach presented in [4][7] for dependable registration of geophysical maps in featureless environments involves several steps. In this paper, we presented new algorithms that improve upon several steps of the approach. These improvements increase both stability and efficiency of the algorithms used in the approach.

In this paper, we discussed registering data patches. These patches are two-dimensional manifolds, but otherwise can be arbitrary. They may be the results of two different surveys, or they may be intersecting swaths of data from the same survey, or one patch may be the data collected along track-lines and the other patch the data taken along tie-lines in a survey, etc. We are also developing similar algorithms for registering a 1D manifold (e.g., point data collected along a single track-line) with 2D as well as 1D manifolds.

Appendix A: Contour Extraction

The technique most commonly used for iso-value contour extraction in a rectangular grid of field values is based on linear interpolation. Consider the cell formed by the four grid points with coordinates $(x_i, y_i), (x_{i+1}, y_i), (x_i, y_{i+1}),$ and $(x_{i+1}, y_{i+1})$, and denote the field values at these grid points by $g_0, g_1, g_2,$ and $g_3$, respectively (Fig. A.1). Suppose we wish to extract the contour with field value $c$. This contour passes through the cell only if some of the $g_i$'s are smaller than $c$ and the others are greater than $c$. For example, if $g_1 < c < g_0$ then the contour enters the cell at point $A$ with coordinates:

$$\begin{align*}
x_A &= x_i + (x_{i+1} - x_i)(g_0 - c)/(g_0 - g_1), \\
y_A &= y_i.
\end{align*}$$
The contour exits the cell at point $B$ if $c < g_2, g_3$; it exits at point $C$ if $g_2 < c < g_3$; and exits at point $D$ if $g_2, g_3 < c$. These are straightforward cases where the contour intersects the cell boundary at only two points; and the contour within the cell is then approximated by a line segment connecting the entry and exit points.

However, the case $g_1, g_3 < c < g_0, g_2$, where the two opposite vertices have high values, presents an ambiguous situation, because there are four entry/exit points (one on each cell edge). In this case, for example, if $A$ is the entry point, then it is not clear if the contour should exit at point $B$ or at point $D$. Note that since the iso-contour cannot cross itself it cannot exit through point $C$ (except in a special case which we will discuss). Below, we show how to resolve this ambiguity in a way that is consistent with the linear interpolation technique.

In the bilinear interpolation method, the field at point $(x, y)$ inside the cell is given by

$$f(x, y) = (1 - u)(1 - v)g_0 + u(1 - v)g_1 + uvg_2 + (1 - u)vg_3,$$

where $(u, v)$ are the cell-normalized coordinates,

$$u = (x - x_i)/(x_{i+1} - x_i), \quad v = (y - y_j)/(y_{j+1} - y_j).$$

Contours of constant value $f(x, y) = c$ inside the cell, are hyperbolic curves given by

$$auv - (g_0 - g_1)u - (g_0 - g_3)v + g_0 - c = 0. \quad (A.1)$$

where

$$a = g_0 + g_2 - g_1 - g_3. \quad (A.2)$$

The two asymptotic lines of the curve in (A.1) are parallel to the $u, v$-axes and are given by the lines

$$u = (g_0 - g_3)/a,$$

and

$$v = (g_0 - g_1)/a.$$

The hyperbolic curve cannot intersect the asymptotes. Hence, if the entry point is $A$ and the exit point is $B$, then point $A$ must be to the right of the $u = (g_0 - g_2)/a$ asymptote, and point $B$ must be below the $v = (g_0 - g_1)/a$ asymptote. The coordinates of point $A$ are $(u, v) = ((g_0 - c)/(g_0 - g_1), 0)$, thus the above condition reduces to $c < b$, where

$$b = (g_0 g_2 - g_1 g_3)/a. \quad (A.3)$$

The condition for point $B$ to be below the $v = (g_0 - g_1)/a$ asymptote also gives the same inequality. Likewise, if $A$ is the entry point and $D$ is the exit point, then we get $c > b$. Once the corresponding pairs of entry/exit points are determined, we approximate the hyperbolic contour by the line segment connecting the entry and exit points.

For the special case when point $A$, as well as other points $B, C, D$, lie exactly on the asymptotes, i.e. when $c = b$, then the contour does indeed intersect itself. In this case points $A$ and $C$ on the opposite edges are the corresponding entry/exit points, while points $B$ and $D$ form the other pair.

Let us denote the line connecting the two opposite vertices with low field values $g_1$ and $g_3$ by the low diagonal, and the line connecting the vertices with high values $g_0$ and $g_2$ by the high diagonal. The following recipe, thus, solves the contour extraction ambiguity: (1) if $c < b$, then the contour crosses the low diagonal; (2) if $c > b$, then the contour crosses the high diagonal; and (3) if $c = b$, then the contour crosses itself passing through the entry/exit points lying on opposite edges. The quantities $a$ and $b$ are given in (A.2) and (A.3).

Acknowledgements

This work was supported by the Office of Naval Research.
Figure 6: Mismatch for false correspondences.

Figure A.1: Contour entry/exit points in a square cell.

References


