Multiple Paths Extraction in Images Using a Constrained Expanded Trellis

Changming Sun and Ben Appleton

Abstract—Single shortest path extraction algorithms have been used in a number of areas such as network flow and image analysis. In image analysis, shortest path techniques can be used for object boundary detection, crack detection, or stereo disparity estimation. Sometimes one needs to find multiple paths as opposed to a single path in a network or an image where the paths must satisfy certain constraints. In this paper, we propose a new algorithm to extract multiple paths simultaneously within an image using a constrained expanded trellis (CET) for feature extraction and object segmentation. We also give a number of application examples for our multiple paths extraction algorithm.

Index Terms—Multiple paths extraction, constrained expanded trellis, feature extraction, object segmentation.

1 Introduction

Single shortest path extraction techniques have been used for a number of applications such as crack detection in borehole images [1], road extraction in satellite images [2], disparity estimation for stereo images [3], [4], [5], optical flow estimation [6], object boundary extraction in images [7], and optimization problems in network and transportation analysis [8]. The result of ordinary shortest path extraction is usually a single optimal path.

In some applications, such as high-performance communication, fault-tolerant routing, transportation networks, and radar multitarget tracking, it is often necessary to obtain multiple paths which are in some sense significantly distinct from each other. In earlier work, such paths have been variously called mutually exclusive paths, disjoint paths, completely unmerged paths, or K best paths [9], [10], [11], [12], [13], [14]. In this paper, we are interested in multiple paths in images where the total sum of values along all the K paths is a minimum. These paths could correspond to multiple features in images or boundaries of objects in images.

Successive application of a single shortest path extraction algorithm to obtain multiple paths, although a simple solution, clearly will not produce the desired optimal solution in general. Nikolopoulos and Samaras proposed solutions to the multiple track detection problem using graph theoretic concepts for detecting discontinuous lines and for achieving an efficient solution [15]. However, their solutions are suboptimal.

For optimal multiple paths finding, there are mainly two classes of algorithms: one based on dynamic programming or the Viterbi algorithm (as described in [14]) and one based on minimum cost network flow (MCNF) algorithms (as described in [12], [13]). The computation cost of the first class of algorithms grows exponentially with the number of paths due to the large network or trellis generated. In the MCNF approach, an input image is converted into an augmented graph and an MCNF algorithm is applied to this graph to obtain the multiple paths which are mutually exclusive (see [13] for details). This approach takes a topological rather than a geometric viewpoint of the paths and, so, manages to avoid explicitly representing the configuration space. As a result, it requires $O((mn)^3 \log(mn))$ computation and $O(n^2 m)$ memory. Although this class of algorithms is relatively efficient, it is difficult to impose necessary constraints on the structure or geometric relationship of the multiple paths. The application of the MCNF method to image analysis has several difficulties which will be discussed in Section 4.3.

In this paper, we propose a new algorithm using constrained expanded trellises (CETs) for feature extraction and object segmentation in images. Our algorithm is much more computationally efficient than the first class of algorithms, as exemplified in [14], and we can easily apply constraints to control the geometry of the multiple paths where the second class of algorithms is unable.

The paper is organized in the following sections: Section 2 gives the problem definition and cost functions. Section 3 describes multiple paths extraction problems and the initial expanded trellis. Section 4 develops a number of geometric and heuristic methods which greatly reduce the computation and memory requirements. Section 5 gives our efficient implementation of dynamic programming in the constrained expanded trellis. We show the results of our algorithm on a range of real images from different applications in Section 6. Section 7 gives concluding remarks.

2 Problem Definition and Cost Functions

2.1 Finite Sequences and Paths

In this paper, $\mathbb{Z}_m$ denotes the sequence of integers $(1, 2, \cdots, m)$. A path is represented by a finite sequence $P : \mathbb{Z}_m \rightarrow R$ which is a function labeling each element of $\mathbb{Z}_m$ with an element from the range R. In this paper, the range is a discrete set consisting of a finite number of points.

- *C. Sun is with CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 1670, Australia. E-mail: changming.sun@csiro.au.*
- *B. Appleton is with Electromagnetics and Imaging, School of ITEE, The University of Queensland, QLD 4067, Australia. E-mail: appleton@itee.uq.edu.au.*

Manuscript received 20 Dec. 2004; revised 28 Feb. 2005; accepted 30 Mar. 2005; published online 13 Oct. 2005.
Recommended for acceptance by A. Srivastava.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0674-1204.

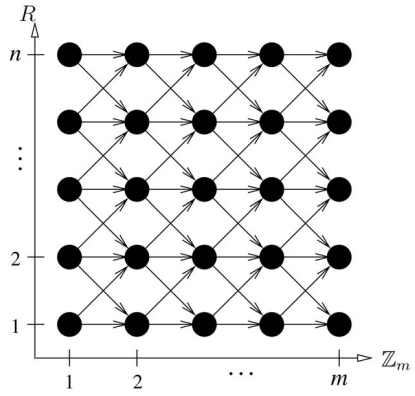

Fig. 1. A simple $n \times m$ trellis. A vertex is only connected to three vertices on its left or right column (apart from vertices on the top or bottom row).

Many problems in image analysis may be phrased as the search for a path or a set of paths in an image or perhaps in a space derived from an image. Paths may often be modeled as finite sequences so that, in this paper, we will consider sequences and paths to be equivalent. This places the restriction on the path that it may not "fold back" on itself. So, the sequence P is single-valued at each point and, for every index $i \in \mathbb{Z}_m$, there is a unique element $p_i \in R$. In many applications, this is a reasonable or even desirable restriction.

The number of paths or sequences of length m with elements drawn from R is n^m, where $n = |R|$ is the number of elements of the range R. As the number of paths grows exponentially in m, it is important that path extraction algorithms avoid enumerating all of these paths. In this paper, we will also consider very large label spaces R, necessitating further analysis to reduce the memory and computational costs in practical implementations. The search for a discrete sequence is usually performed on a trellis, which is a graph on the discrete grid $\mathbb{Z}_m \times R$. Fig. 1 shows a trellis with $n \times m$ vertices.

2.2 A Class of Objective Functions for Discrete Sequences

In image analysis, noise, occlusions, and irrelevant objects often make it difficult to select the correct paths. There may be a very large selection of seemingly good solutions. Rather than simply selecting a good path on an ad hoc basis, in this paper, we frame the path extraction problem as the minimization of an objective function as has been done in the past [16]. This objective function, is designed so that reasonable solutions have low cost while poor solutions have high cost. Ideally, the path of minimal cost is then the correct path. This approach is widely used in image analysis because it is able to produce accurate and robust solutions [16], [17], [18].

We first define the cost function for a single path. Let $C(P)$ denote the real-valued cost of path P. We consider a simple but effective class of cost functions:

$$C(P) = \sum_{i=1}^{m} c_0(i, p_i) + \sum_{i=1}^{m-1} c_1(i, p_i, p_{i+1}). \quad (1)$$

Each vertex (i, p_i) of the trellis represents the potential assignment of label p_i to index i of the sequence. A point (i, p_i) is an element of the trellis $\mathbb{Z}_m \times R$. Edges connect vertices (i, p_i) and $(i+1, p_{i+1})$ in sequential columns of the trellis. A column of a trellis is a subset of the trellis given by fixed values of i. We assign costs to the vertices and edges of the trellis which are taken directly from the path cost defined in (1).

The $c_0(i, p_i)$ in the above equation denotes the cost of the element p_i of the sequence P at index i, i.e., it is a function of elements of the trellis. We refer to this as a zero order term, or vertex cost, in the cost function. $c_1(i, p_i, p_{i+1})$ denotes the cost of the sequence of elements (p_i, p_{i+1}), which we refer to as a first order term, or edge cost, in the cost function. It is a function of "neighboring" pairs of elements.

The choice of vertex and edge costs is specific to an application. For example, when detecting predominantly dark cracks on a brighter background, we may search for a path composed of dark pixels. In this case, a simple but effective vertex cost function is the intensity of each pixel, $c_0(i, p_i) = I(x, y)$, which is the image intensity at position (x, y). Likewise, to detect bright lines on a dark background, we may invert the image, setting $c_0(i, p_i) = -I(x, y)$. Object boundaries are often strongly correlated with intensity gradients, so we may use the inverse of the image gradient as a vertex cost function. In this case, a common choice is $c_0(i, p_i) = \frac{1}{1+|\nabla G_\sigma * I|}$, where $|\nabla G_\sigma * I|$ computes the absolute gradient of the image convolved with a Gaussian of scale σ [19].

In both image segmentation and stereo matching, it is important that the resulting paths be continuous, yielding connected object boundaries or smooth reconstructed surfaces, respectively. This also improves the results in regions where the correct path is obscured and must be inferred from surrounding information. In order to ensure that results are smooth, an edge cost function may be derived to penalize discontinuities or large jumps. A simple approach is to only allow neighboring labels to differ by a small amount:

$$|p_i - p_{i+1}| \leq k.$$

This corresponds to setting the edge cost function to

$$c_1(i, p_i, p_{i+1}) = \begin{cases} 0, & |p_i - p_{i+1}| \leq k \\ \infty, & \text{otherwise.} \end{cases}$$

Here, k controls the connectivity of the initial trellis with $2k+1$ edges per vertex. In the applications considered in this paper, it is sufficient to consider $k = 1$ or 2 at most. More sophisticated edge cost functions have been used in, for example, [20] and [21].

A shortest path is then a path which has minimum total cost,

$$P_{\min} \in \arg\min_{P} \{C(P)\}.$$

Observe that there may be more than one such path in the case of ties, so we speak of obtaining *a* shortest path rather than *the* shortest path.

A common method of extending this to detecting closed object boundaries is to take the polar transform of the image about a point inside the object. In this case, pixels are referenced by their polar coordinates θ and r and we may compute a sequence of the form $r(\theta)$. There are several ways to ensure that the endpoints of this shortest path meet to form a closed contour, see [7] or [22] for details. These approaches are only able to deal with point-convex objects; however, this is sufficient in many applications. A more sophisticated

approach which is able to deal with arbitrary closed curves is presented in [23]. This method extends a planar image to a spiral space in which points on the shortest path represent both the spatial position and the number of times that the curve has passed around the center at that point.

This simple definition of paths and their associated objective functions may be extended to computing several paths in one image or to computing joint paths in several images. The next section extends the single-path framework above to the case of multiple paths.

3 Multiple Paths Cost and Expanded Trellis

In some applications, several paths must be extracted. This includes the detection of multiple linear features across an image, such as crack detection or aerial road detection, or in finding the boundaries of objects composed of several layers. A set of K paths $(P^1, P^2, \cdots, P^K) \equiv \vec{P}$ across an image or trellis may be represented as the sequences $p_i^1, p_i^2, \cdots, p_i^K$ indexed by i. We refer to the set of paths $\vec{P}$ as a *multipath*. The points from different paths will be disjoint; this disjointness constraint will be discussed in the next section along with other constraints.

The cost of the multipath $\vec{P}$ is the sum of the costs of the component paths:

$$C(\vec{P}) = \sum_{j=1}^{K} C(P^j).$$

From this, we may derive equivalent vertex and edge cost functions for the vector path $\vec{P}$ from the sums of the cost functions of the individual sequences:

$$C(\vec{P}) = \sum_{j=1}^{K}\sum_{i=1}^{m} c_0(i, p_i^j) + \sum_{j=1}^{K}\sum_{i=1}^{m-1} c_1(i, p_i^j, p_{i+1}^j). \qquad (2)$$

In order to find a combination of bright and dark paths, one could attach a sign function $\mathrm{sign}(j)$ to the vertex cost function $c_0(i, p_i^j)$ so that the new vertex cost is $\mathrm{sign}(j)c_0(i, p_i^j)$. This sign function will be the same for every column of the input image or trellis and the possible values for $\mathrm{sign}(j)$ can be either 1 or -1, where sign value 1 corresponds to a dark path and sign value -1 corresponds to a bright path. Therefore, by providing a sign function with different combinations of 1 and -1s, one could find multiple paths with combined dark and bright paths. For example, if all the values are 1, the result will be the K darkest paths in the image.

The range R of the multipath is an expanded space formed from the outer product of the ranges of the constituent paths and is known as the configuration space. With this approach, when one is to compute K paths across an $n \times m$ image, the optimal path search can be performed on an *expanded trellis* with dimensions $n^K \times m$ if no constraints are applied. Fig. 2 depicts an example of the expanded trellis for locating the two best paths ($K = 2$). Unfortunately this translates to time and space requirements of $O(n^K m)$, so the problem quickly becomes infeasible as the number of paths K grows. We will apply several novel constraints in the next section to reduce this number.

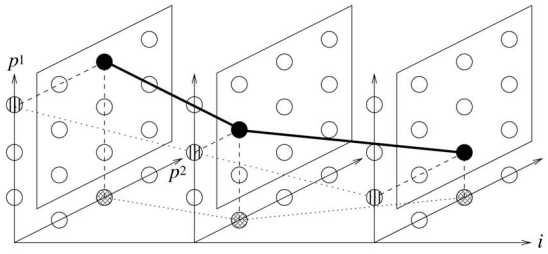

Fig. 2. The expanded trellis for $K = 2$ paths. Depicted is a single best path in the expanded trellis (solid line) and the corresponding K best paths (dotted lines).

4 Constraints on Multiple Paths

In this section, we consider a number of constraints which may be used to reduce the running time and memory requirements of the shortest multipath problem. These constraints also regulate the structures of the multiple paths. We first consider the effect of geometric (disjointness, ordering, and spacing) constraints on the multiple paths. We then consider constraints motivated by heuristic arguments which are designed to further reduce the computational load of the multiple paths extraction problem.

4.1 Disjointness Constraint

In order to avoid obtaining a trivial multipath composed of K copies of a single shortest path, we require that the paths be mutually disjoint at every point. That is, we require, for all indices $i = 1, \cdots, m$ and paths $j \neq k$, that $p_i^j \neq p_i^k$. This constraint may be added into the trellis framework by simply setting the vertex cost to ∞ for the vertices in the trellis with a repeated coordinate or removing those vertices in the expanded trellis which do not satisfy this disjointness constraint.

4.2 Ordering Constraint

For a path $\vec{P}$ in the expanded trellis, each vertex contains K coordinates corresponding to the K individual paths. The order of these paths may be permuted to produce a new, equivalent path in the expanded trellis with the same or lower cost (when there is no sign function attached to $c_0(i, p_i^j)$ or this sign function contains all 1s).

In practical applications of shortest paths, the edge costs component $c_1(i, p_i, p_{i+1})$ is monotonically nondecreasing with the size of the jump $|p_i - p_{i+1}|$. In this case, if an order is imposed on the components of a path in one column of the trellis, then all further columns of the trellis can be so ordered without increasing the path cost.

Proceeding inductively, we observe that, in each column of the expanded trellis, the paths may be represented in order without increasing the cost of the path. This also implies that the shortest K paths do not need to cross. The disjointness and ordering constraints together reduce the second dimension of the trellis from n^K to $\binom{n}{K}$, which is a factor of approximately $K!$.

Fig. 3a shows paths with crossing and Fig. 3b shows paths without crossing for two neighboring columns. The noncrossing constraint helps reduce the computational complexity of our constrained expanded trellis algorithm significantly.

The two paths obtained by the MCNF approach shown in Fig. 4 cross each other. However, this crossing can be

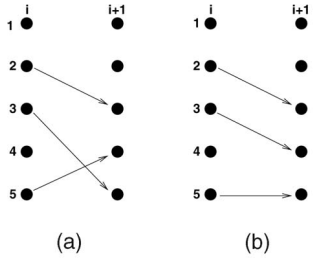

Fig. 3. (a) Paths can cross each other. The number of possible combinations of K path positions on column i and K path positions on column $i+1$ is $K!$. (b) Paths do not cross each other. The number of possible combinations of K path positions on column i and K path positions on column $i+1$ is just one. In this example, K is 3.

postprocessed or switched after the K paths have been obtained so that no crossing is present if needed.

4.3 Spacing Constraints

A practical problem with computing disjoint paths in an image is that they may be arbitrarily close together. In particular, paths which travel close to the shortest path tend to have low costs, resulting in paths bunching together. Fig. 4a contains two paths obtained by using the MCNF approach. The two paths are next to each other and we can hardly see the path's separation. Figs. 4b and 4c gives detailed illustrations of two paths that are next to each other, one path in white and one in black.

This problem may be easily remedied in the expanded trellis framework by removing vertices whose constituent coordinates are too close together. We can set a minimum spacing constraint δ for the paths to prevent them from coming too close to each other. Similarly, a maximum spacing constraint Δ between adjacent paths can also be used so that paths will not be too far apart from each other. In addition to controlling the spacing of the paths, these spacing constraints have the added benefit of reducing the number of combinations of vertices. Vertices that are too close ($< \delta$) or too far ($> \Delta$) from each other will be removed from the expanded trellis. This greatly reduces the size of the expanded trellis, leading to substantial savings in computation time and memory.

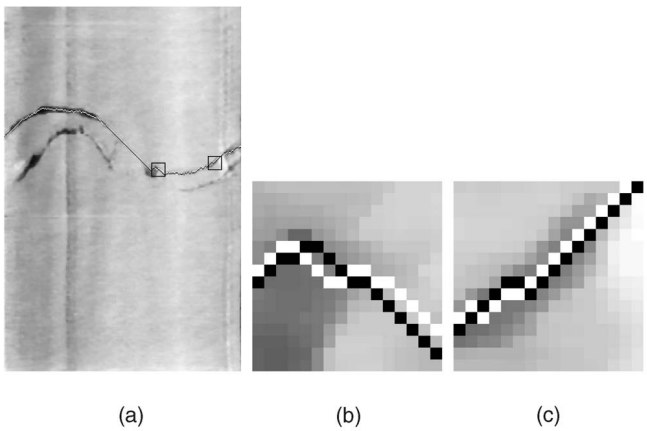

Fig. 4. Paths can be close to each other without constraints. (a) An example of two paths that are next to each other (shown in white and black) obtained using the MCNF approach. (b) and (c) Two small regions (the black squares in (a)) showing the paths details.

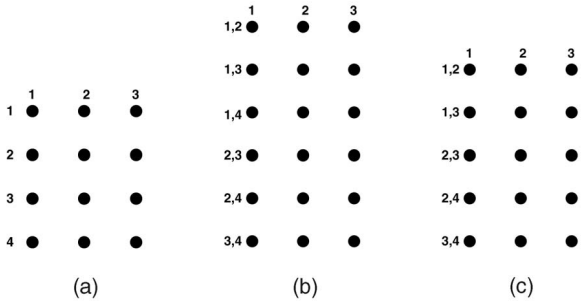

Fig. 5. Building the initial and the constrained expanded trellis. $K = 2$ and $\binom{n}{K=6}$. (a) Input image/grid. (b) Expanded trellis with disjointness and ordering constraints. (c) Expanded trellis with spacing constraints (vertices with the (1, 4) entries in (b) are removed because the maximum spacing Δ between candidate rows is larger than the given range with $\delta = 1$ and $\Delta = 2$). The grid in (a) is in the input space as shown by black dots, while the grids in (b) and (c) are grids in the expanded trellis space shown by gray dots.

While the MCNF approach is more efficient for a large number of paths K, it cannot be modified to include spacing constraints in the same way as the expanded trellis approach. As a result, the MCNF approach usually produces paths that are next to each other instead of locating several significant paths in an image, as shown in Figs. 4 and 11.

Fig. 5 shows an example of the expanded trellis structure and its reduction in size using geometric constraints. Fig. 5a is the input image shown as a grid of points. Fig. 5b shows the vertices of the expanded trellis with disjointness and ordering constraints (the connections between vertices from neighboring columns are not shown). In the expanded trellis shown in Fig. 5b, the number of rows equals $\binom{n}{K}$, i.e., choosing K from n. The number of columns of the expanded trellis is the same as that of the input image. Each vertex on a column of this expanded trellis contains one possible combination of K (in this example, $K = 2$) row positions from the same column of the input image. Fig. 5c illustrates the reduced trellis using spacing constraints. In this small case, the vertices containing the combination (1, 4) are removed with $\delta = 1$ and $\Delta = 2$.

4.4 Heuristic Constraints

The geometric constraints just developed may be applied to greatly reduce the size of the expanded trellis. However, the computational and memory costs remain formidable when extracting several paths. So, we are interested in further savings which may be made by heuristic methods.

Vertex Cost Constraint. We can reduce the size of the expanded trellis further by using the information of the vertex costs derived from the image. This can be carried out by checking the sum of the vertex costs at the K candidate positions at each column of the image, equivalently for each vertex of the expanded trellis. Vertices which have a very high cost probably do not belong to a path of interest and can therefore be discarded beforehand to further reduce the size of the expanded trellis. Each column of the expanded trellis is sorted beforehand and only a fixed fraction of lowest-cost vertices is retained for further processing.

Table 1 shows some example sizes of the expanded trellis with and without the geometric and vertex cost constraints. It can be seen from the table that the size of N_3 is significantly smaller than N_1 for large K. The numbers in Table 1 are for

TABLE 1
Combination Numbers (Related to the Size of the Expanded Trellis) without Constraint and after Using Different Constraints

K	N_1	N_2	N_3
1	128	128	38
2	8128	1023	307
3	341376	7018	2105
4	10668000	30613	9184

K: number of paths, N_1, N_2, N_3: the number of vertices at each column, N_1: with disjointness and ordering constraints, N_2: with further spacing constraints, N_3: with vertex cost constraint on N_2, (image column and row numbers: $n = 128,\ m = 128,\ \delta = 30,\ \Delta = 40$, retaining only the lowest 30 percent of vertices).

noncrossing paths. If path crossing is permitted, the numbers in the table need to be scaled by $K!$.

Local Minima Constraint. We may also investigate the use of just those local minimum values of each column of the input image to reduce the computational cost in future studies. The intensity values in each column of an image can be treated as a 1D function and the positions of the local minima of the function can be obtained. These local minimum positions in each column can be used to build the initial trellis for multiple paths extraction. Because the number of local minimum positions on a column is usually much smaller than the length of the column, n, computational cost can be reduced. In addition, using only these local minima reduces the possibility that the paths are very close.

MCNF Preprocessing Constraint. It is also possible to use the MCNF algorithm as a preprocessing step in the expanded trellis approach when the number of paths K is significant. In this case, we first run the MCNF algorithm on the full image to obtain a large number of candidate paths which are mutually exclusive. For example, we could compute the best $5K$ paths by this approach. We then consider only these $5K$ pixels for each column in the construction of the expanded trellis to greatly reduce the size of the trellis and the consequent running time. This heuristic is more robust than the simpler heuristic based on vertex costs alone. However, it is still possible that the combination of MCNF and the expanded trellis algorithm will produce a set of paths which is not optimal. Note that applying one or more of these heuristics may render the solutions approximate.

5 Practical Implementation

The constraints presented in the previous section can reduce the number of vertices in each column of the expanded trellis from its original size n^K to a tiny fraction $N \ll \frac{n^K}{K!}$, making vast savings in computation time and memory. However, the vertex cost heuristic and the MCNF heuristic produce trellises which are extremely sparse. For heuristics which produce an irregular trellis, great care must be taken to avoid increasing the complexity of the shortest path algorithm in practice.

A fundamental operation of the dynamic programming algorithm in the forward sweep is locating all neighbors of a vertex in the previous column. As the set of vertices is sparse, a simple approach as taken in [14] is, for each vertex in the current column, to search the entire previous column to locate its neighbors. However, this increases the computational complexity of our dynamic programming algorithm from $O(mN)$ to $O(mN^2)$, i.e., taking time proportional to the square of the number of vertices in each column. Even with the constraints outlined above, the number of vertices can still be extremely large, leading to a very inefficient implementation.

In order to search more efficiently for the neighbors of a vertex from one column to the previous column, we place the vertices of each column in an indexing structure. Then, rather than scanning the entire column, we may simply query the indexing structure to determine if a particular neighbor exists and, if so, where it is located in memory. The coordinates of each column's N remaining vertices are stored in a sequential array of length N.

Our implementation uses a multiway trie, which is a data structure for efficiently mapping between symbol strings and their unique keys. Vertex coordinates may be reinterpreted as strings and their keys are their position in the table of vertices. The trie that we use is a tree of order 256. Beginning at the root of the tree, we branch down the tree following each byte of the vertex string to reach the appropriate leaf node. The leaves of the tree hold the keys corresponding to their associated strings. Strings which have not been added to the trie lead to a null leaf at some point, which indicates that the corresponding vertex does not exist.

As each column is processed in sequence, it is only necessary to maintain a trie on one column at a time leading to negligible overhead memory. When processing a column, the trie is initially empty, consisting of a single null node. Each vertex in the current column is then added to the trie along with its associated index in the table of vertices. As strings are added, the trie branches down the tree following the bytes of the string toward the associated leaf. This data structure allows dynamic programming to operate in $O(mN)$ time and space as desired. For a more detailed description of multiway tries, see [24].

The steps for obtaining a single shortest path in the constrained expanded trellis are the following:

1. Compute the vertex cost at each and every vertex in the constrained expanded trellis as the sum of the K component vertex costs. This is to calculate the first term of (2). The sign function may also be used for detecting both bright and dark paths simultaneously.
2. For each of the remaining columns (from the second to the last column):
   a. Initialize the trie for the current column.
   b. For each vertex in this column, calculate the partial shortest path cost (this includes calculating the second term of (2)) and add in the vertex cost obtained in Step 1 for every neighboring vertex in the left column.
   c. Take the least of these costs to obtain the shortest path to the current vertex.
   d. Save the index of the vertex on the left column which gives the least cost.
   e. Delete the trie for the current column.
3. Backtrack the path from the last column to the first.

In addition, tries may be used for sparse but structured trellises such as those created by geometric constraints. In this case, the structure of the trellis is the same in each column, so the trie only needs to be calculated once. This may lead to substantial savings in practice.

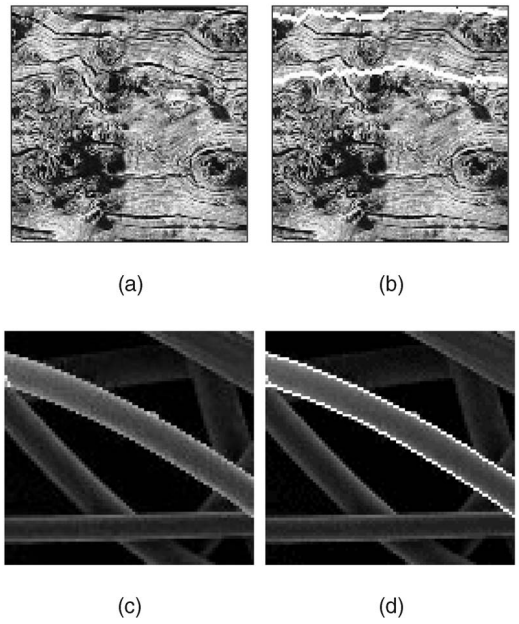

Fig. 6. Input and multiple paths. (a) and (b) Tree bark, $K = 2$. (c) and (d) Fiberglass, $K = 2$.

Finally, it should also be noted that the overzealous application of heuristics may lead to a disconnected trellis. In the implementation of dynamic programming, it is simple to detect such instances and halt gracefully.

After a shortest path is obtained in the constrained expanded trellis (e.g., Fig. 5c), the K paths in the input trellis or image can be obtained. For example, if the single shortest path obtained in the constrained expanded trellis passes through vertices $(1, 2)$ in column 1, $(2, 3)$ in column 2, and $(2, 4)$ in column 3 of Fig. 5c, then the $K(= 2)$ best path positions in Fig. 5a are: 1) (row 1, column 1) then (row 2, column 2) and then (row 2, column 3) and 2) (row 2, column 1) then (row 3, column 2) and then (row 4, column 3).

The steps of our complete CET algorithm for multiple paths extraction in images are the following:

1. If necessary, transform the image. For example, Cartesian to polar coordinate transformation for circular shaped objects (the rough position of the object center will be provided by users) or rotate the image by 90 degrees if vertical paths are to be extracted. Also, carry out any required preprocessing such as computing the image gradient.
2. Obtain K optimal paths:
   a. Build an initial expanded trellis from the input image or from the transformed image obtained in Step 1.
   b. Eliminate vertices on the initial expanded trellis using constraints (except the local minima and MCNF preprocessing constraints) as described in Section 4.
   c. Perform dynamic programming on the constrained expanded trellis to find a single path with minimum cost. The ordering (noncrossing and path connection) constraint is applied here and tries are used for an efficient neighbor search (see the algorithm given above).
   d. Convert this single path on the constrained expanded trellis to the K optimal paths on the input image.
3. If Cartesian to polar coordinate transformation was carried out in Step 1, a reverse transformation is necessary to obtain closed multiple paths in the original image. A rotation of -90 degrees on the result image is needed when vertical paths are extracted.

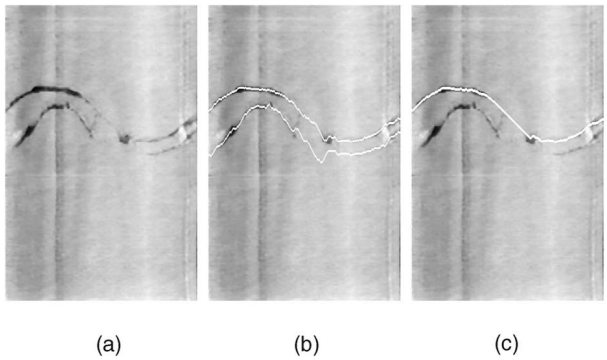

Fig. 7. Input and multiple paths. (a) Input borehole image. (b) Two paths obtained using our CET algorithm, $K = 2$, $k = 2$. (c) Two paths obtained using the MCNF algorithm (the two paths are actually next to each other).

6 Experimental Results

This section shows some of the feature extraction and object segmentation results on images obtained using our CET algorithm described in previous sections. A variety of real images have been tested. Comparisons with the MCNF approach are also carried out.

Fig. 6a shows an input image of tree bark and Fig. 6b shows the two paths found. Fig. 6d gives the two boundaries of some fiberglass shown in Fig. 6c. Fig. 7a shows a borehole image with cracks and Fig. 7b shows the two paths found using our CET algorithm. Fig. 7c shows two paths obtained using the MCNF method; the two paths are actually next to each other. It is clear from this example that the result obtained using our CET algorithm is more desirable than that obtained using the MCNF method.

Fig. 8 gives more results of using our multiple paths detection algorithm on different images (aerial and microscopic). Fig. 8b shows the two sides of the road boundary in an aerial image and Fig. 8d shows two gross boundaries of wood cells, with the inputs given in Figs. 8a and 8c, respectively.

Fig. 9 gives further examples of multiple paths extraction in images. Figs. 9a and 9b show road crack images and Figs. 9c and 9d show an image of lightning bolts. Two of the computed paths approach each other at one point because the minimum spacing constraint δ is set to 1. Figs. 9e and 9f are the results of linear feature detection of a rope underwater.

Fig. 10 shows results for multiple paths extraction in seismic data. Fig. 10a is the input image. Fig. 10b is the result of detecting only one path. Fig. 10c is the result of obtaining three paths in the image. Notice that the single path shown in Fig. 10b is not present in the three paths in Fig. 10c. This confirms that successive application of single shortest path extraction will not give optimal results for multiple paths extraction.

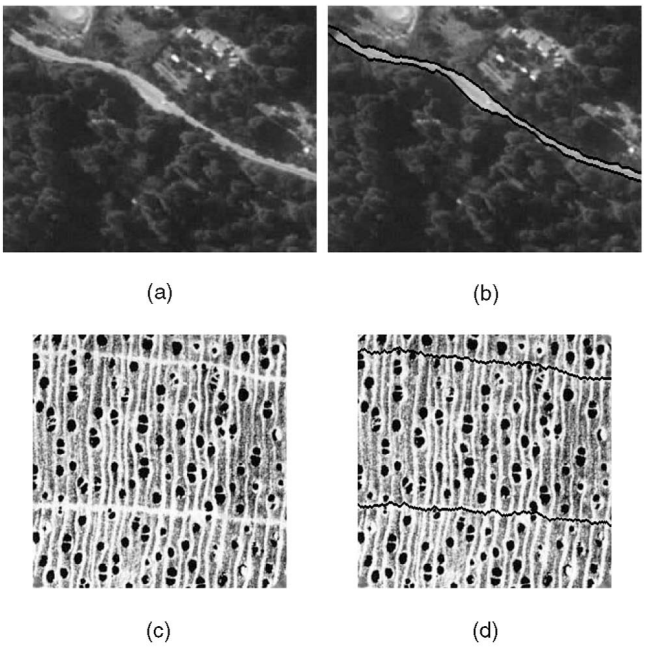

Fig. 8. Features extraction for road boundaries in aerial images and gross boundaries of wood cells. (a) and (c) Input images. (b) and (d) Object boundaries extracted (black lines). (b) $K = 2$, $\delta = 5$, $\Delta = 30$. (d) $K = 2$, $\delta = 5$, $\Delta = 180$.

Figs. 11b and 11c give the results of our algorithm for crack detection in borehole images (the input shown in Fig. 11a) with $K = 2$ and different δ and Δ values. For Fig. 11c, a $\Delta = 50$ is used rather than $\Delta = 80$ as in Fig. 11b. This reduces the maximum spacing that the two paths can have. Two different pairs of paths are obtained in these two images as, in Fig. 11b, the paths are separated by more than 50 pixels at some points. The rest of the figures show results using the MCNF approach with different values of K. The two paths obtained using our CET algorithm shown in Fig. 11b are similar to the three or four paths shown in Figs. 11e and 11f obtained using the MCNF approach. It can be seen from this figure that the MCNF approach usually produces a number of paths next to each other due to a lack of geometric constraints.

Fig. 12 gives more results of using our multiple paths extraction algorithm on different medical images. For the input images shown in Figs. 12a and 12d, we convert them into polar coordinates with the origin roughly at the image center before applying our CET algorithm, as shown in Figs. 12b and 12e. The inverted gradient image provides the vertex costs for obtaining the object boundaries. Figs. 12c and 12f show the object boundaries obtained with $K = 2$.

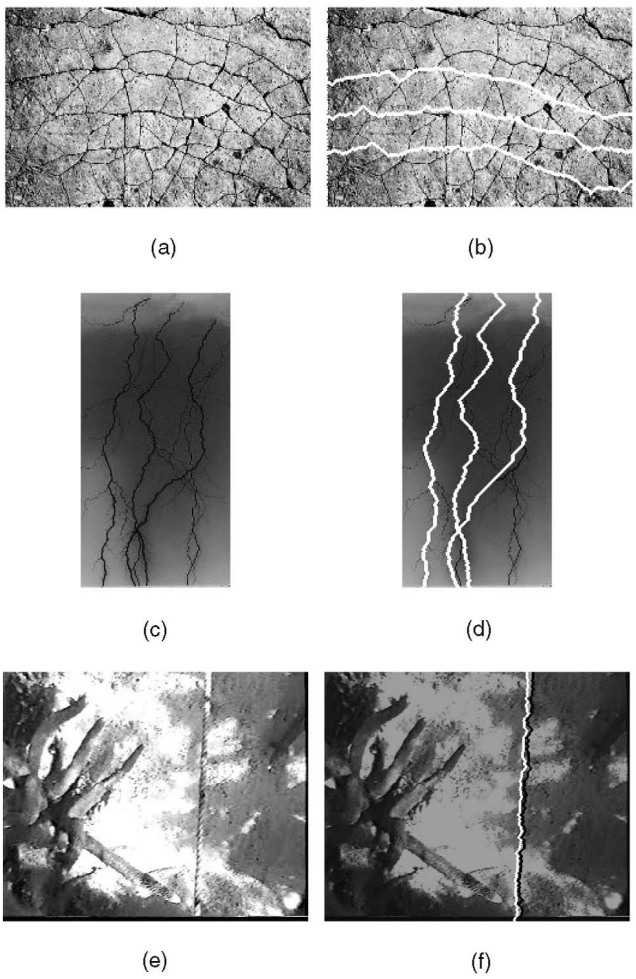

Fig. 9. More examples of multiple paths extraction in images. (a) and (b) Road crack images and paths extracted, $K = 3$, $\delta = 10$, $\Delta = 40$. (c) and (d) Image of lightning bolts and paths extracted, $K = 3$, $\delta = 1$, $\Delta = 60$. (e) and (f) Image of a rope underwater and rope boundaries extracted. The intensity of the image in (f) is adjusted so that the two paths (one black and one white) can be seen easily. $K = 2$, $\delta = 4$, $\Delta = 7$.

Figs. 13 and 14 show some more results of our algorithm on different images: an eye, a cut orange, a flower, wood rings, a dog jumping through a hoop, a donut, a broken tire, a meter, and a cardiac MR image. For these examples, the input images are first transformed into polar coordinates and gradient

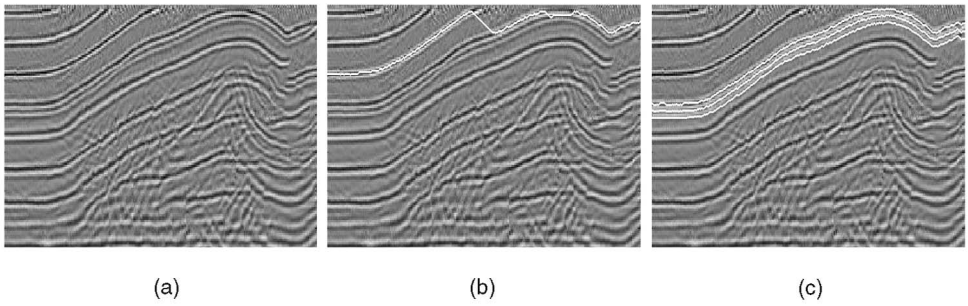

Fig. 10. Multiple paths obtained for seismic data. (a) Input image. (b) Single path obtained, $K = 1$. (c) Three paths with constraints obtained, $K = 3$, $\delta = 5$, $\Delta = 10$. Notice the different positions of paths in (b) and (c).

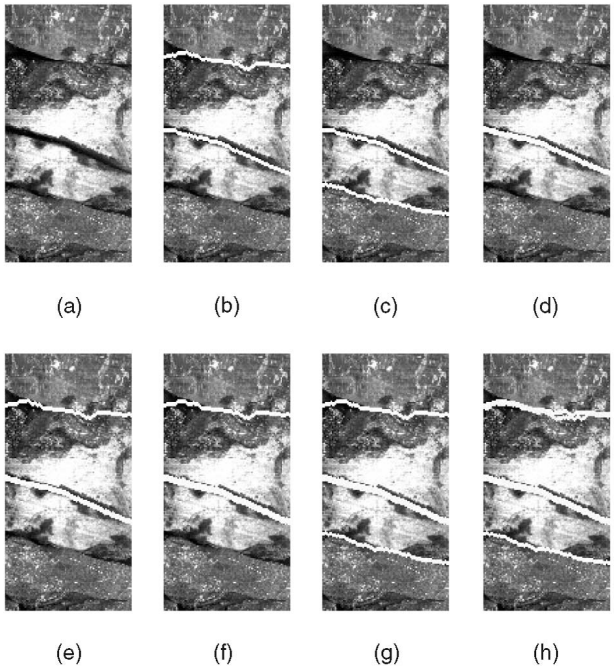

Fig. 11. Multiple paths obtained for borehole images. (a) Input image. (b) and (c) Results with our CET method. (b) CET, $K = 2$ $\delta = 20$, $\Delta = 80$. (c) CET, $K = 2$ $\delta = 20$. $\Delta = 50$. (d), (e), (f), (g), and (h) Results via the MCNF approach (paths next to each other). (d) MCNF, $K = 2$. (e) MCNF, $K = 3$. (f) MCNF, $K = 4$. (g) MCNF, $K = 5$. (h) MCNF, $K = 6$.

operations are carried out (the gradient of the MR image was also equalized). The origin of the polar coordinates must be inside the object boundaries. For multiple objects, such as two eyes in a single image, two separate runs of the CET algorithm are necessary. Results shown in Figs. 12, 13, and 14 are examples of objects with closed boundaries.

Fig. 15 shows some example results for feature detection in an image taken underwater near a coral. Fig. 15a is the input image taken underwater containing a rope and coral, Fig. 15b is the gradient image of Fig. 15a (stretched to the 0-255 range for display purposes), Fig. 15c is a single brightest path obtained from the original image Fig. 15a, Fig. 15d shows two paths obtained using the gradient image Fig. 15b, Fig. 15e shows three paths obtained using the gradient image Fig. 15b, and Fig. 15f shows three paths obtained using the gradient image Fig. 15b with signs of 1, -1, and 1 attached to the vertex cost function (the gradient image is inverted when obtaining the results shown in Figs. 15d, 15e,and 15f). These signs allow us to locate a central dark path along with one bright path on each side. It can be seen that the three paths found in Fig. 15f best align with the object of interest, the rope.

Fig. 16 shows another example for object boundary detection using our CET algorithm. Fig. 16a is an image of a cell with a fuzzy boundary; Fig. 16b shows the cell boundary obtained by finding two paths with signs -1 and 1 attached to the vertex cost function. This corresponds to one dark path a distance of 6 or 7 pixels inside a bright path (i.e., $\delta = 6$ and $\Delta = 7$). Although we obtained two paths for the cell boundary, the two paths are very similar. For this example, a single shortest path extraction can also be used on the inverted gradient image for cell boundary extraction. In this case, the image gradient may be obtained using a scale of about 7 pixels. Both Figs. 15 and 16 give examples for the use of the sign function in our CET algorithm.

6.1 Running Times

On a 2.4GHz Pentium 4 running Linux, our CET algorithm takes about 1.9 s on a 256×256 image to locate the two best

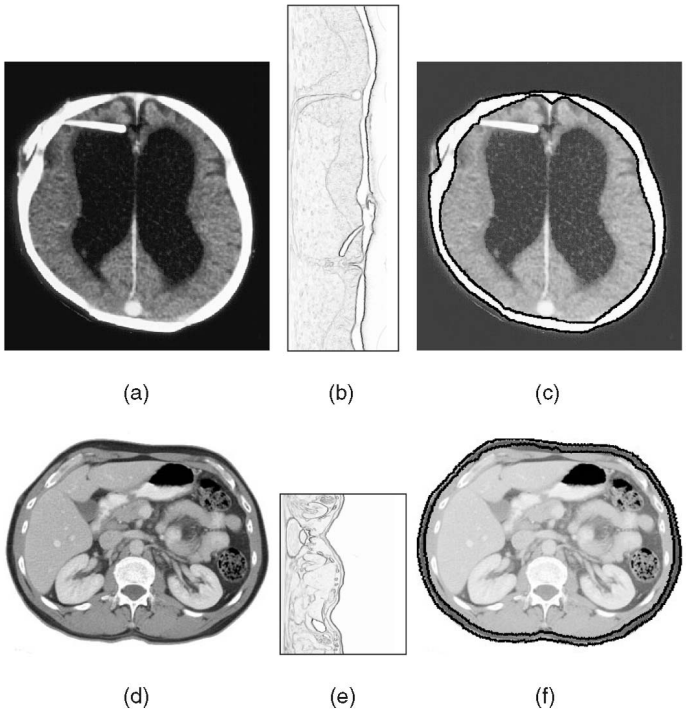

Fig. 12. Extracting features in medical images. (a) and (d) Input images. (b) and (e) Inverted gradient image of the polar transformed images. (c) and (f) Object boundaries extracted (black lines). In both the images, $K = 2$ for the inside and outside boundaries. The images in (c) and (f) are also Gamma (= 2) transformed for display purpose. (c) $\delta = 5$, $\Delta = 30$. (f) $\delta = 5$, $\Delta = 8$.

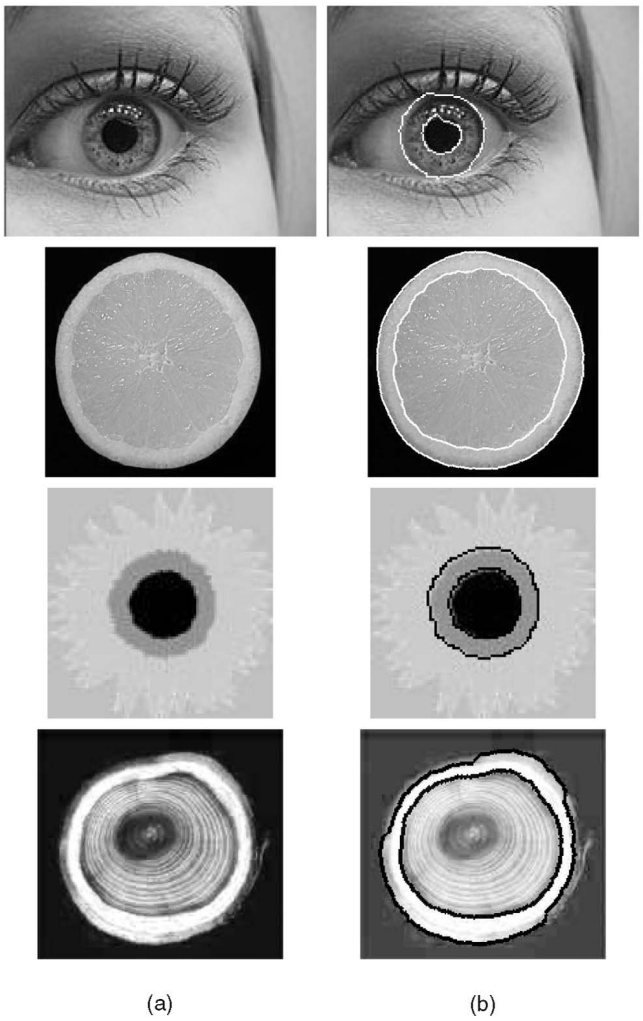

Fig. 13. Multiple paths obtained for some images (eye, cut orange, flower, and wood rings). (a) Input images. (b) Object boundaries obtained, $K = 2$.

paths with $\delta = 30$, $\Delta = 40$, and retaining only the lowest 60 percent of vertices. Table 2 shows the running times of our algorithms with different percentages of vertices that are kept in the expanded trellis. Clearly, a lower percentage of vertices in the trellis leads to significantly reduced computing times.

Table 3 gives comparative timings for using our efficient method and the whole column search method on images with random noise. The parameter settings when running the algorithm are given in the table. It can be seen from the table that our efficient method with the use of a trie is much faster than searching the entire column, especially for large image sizes or for a large number of paths.

6.2 Discussions

Here, we summarize all the parameters that we can use for the multiple paths extraction algorithm.

K: The number of paths sought.

δ, Δ: Minimum and maximum spacings between neighboring pair of paths.

k: Connectivity of vertices between neighboring columns of the initial trellis.

percent: Percentage of vertices in the constrained expanded trellis to retain for further processing.

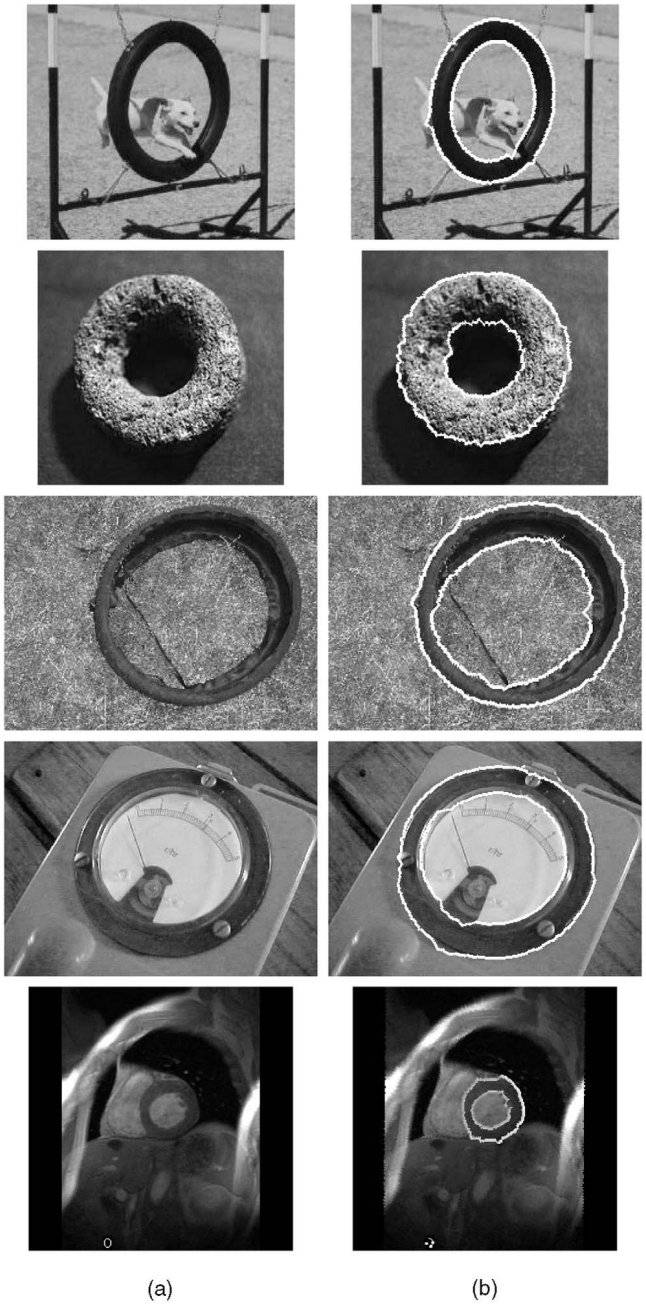

Fig. 14. Multiple paths obtained for other images (dog in circle, donut, broken tire, meter, and a cardiac MR image). (a) Input images. (b) Object boundaries obtained, $K = 2$.

sign: Sign function which may be used for extracting a combination of dark and bright paths.

crossing: Whether crossing is allowed in the algorithm.

The first three parameters in the above list, K, δ, and Δ need to be specified for the algorithm. The two important ones are the minimum and maximum spacing constraints: δ and Δ. The smaller the difference between δ and Δ, the bigger the correlation of shapes of multiple paths that we may obtain. If the multiple paths present in an image have little geometric relationship, then δ may need to be small and Δ may need to be large. Therefore, the algorithm may be slow.

In this paper, the parameter k, which is related to the edge cost function, has always been set to 1 with the exception of

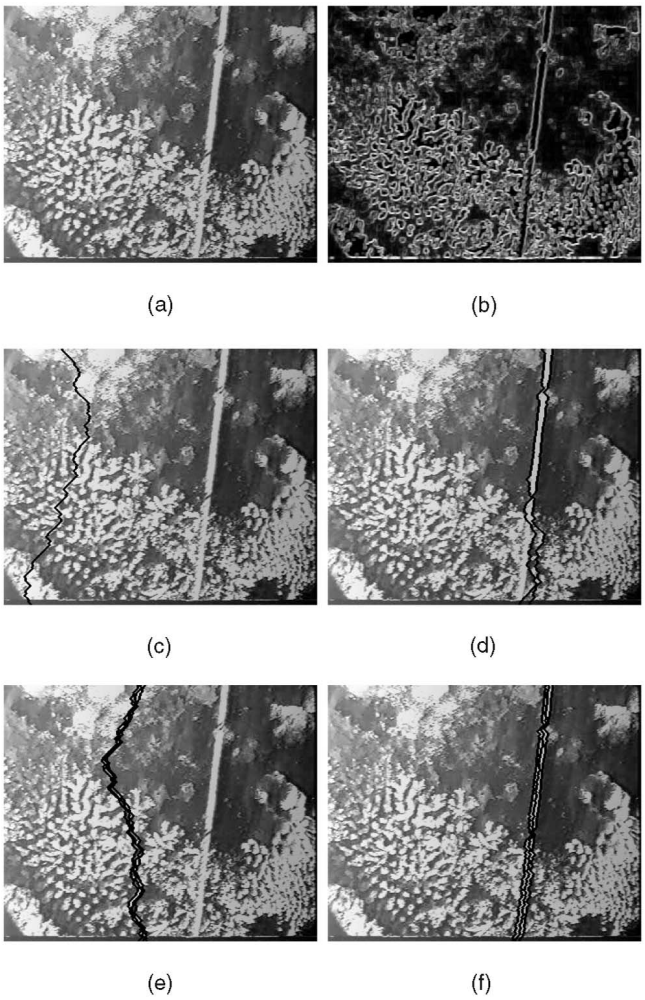

Fig. 15. A rope detection example with the image taken underwater near a coral. (a) Input image taken underwater containing a rope and coral. (b) Gradient image of (a) (stretched to the 0-255 range for display purpose). (c) A single brightest path obtained from the original image (a), $K = 1$. (d) Two paths obtained using the gradient image (b), $K = 2$. (e) Three paths obtained using the gradient image (b), $K = 3$. (f) Three paths obtained using the gradient image (b) with signs of 1, -1, and 1 attached to the vertex cost function, $K = 3$.

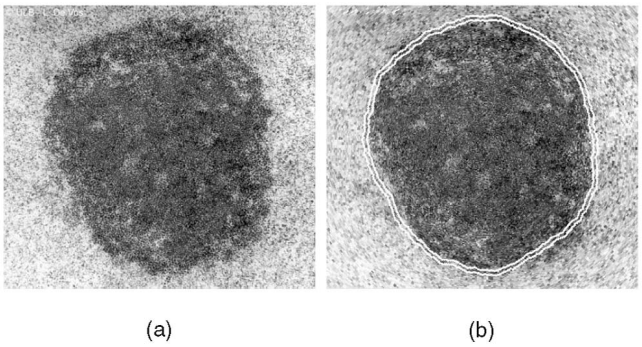

Fig. 16. A cell boundary detection example where the cell boundary is poorly defined. (a) Input image of a cell with a fuzzy boundary. (b) Cell boundary obtained using $K = 2$ with signs = -1, 1, $\delta = 6$, and $\Delta = 7$.

TABLE 2
Running Times of Our CET Algorithm on Images with Various Percentage Values

Percentage	Running Times (s)
100	8.85
90	8.21
80	7.76
70	6.92
60	6.15
50	5.41
40	4.47
30	3.52
20	2.61

$\delta = 2$, $\Delta = 5$, $K = 3$, *and image size* 256×256.

TABLE 3
Running Times of the Algorithm with Whole Column Search or with the Use of Tries on Different Images with $\delta = 2$, $\Delta = 5$, and Keeping 80 Percent of the Vertices

K	Image size	Running Times (s)	
		Column search	Use of trie
2	64×64	0.15	0.07
	128×128	1.07	0.34
	256×256	8.23	1.29
3	64×64	1.83	0.45
	128×128	14.90	1.90
	256×256	120.60	7.76

Fig. 7b, where it is set to 2. For $k = 1$, each vertex on the initial trellis has $2k + 1 = 3$ edges connected to vertices in the left column, whereas, for $k = 2$, each vertex has five connections. The vertex cost function in (2) can be just the image intensity values or image gradients. One can also select a percentage threshold so that only this percentage of vertices in the constrained expanded trellis are used for processing. The sign(j) function, with values either 1 or -1 for each index j ($j = 1, \cdots, K$), enables us to find a combination of dark or bright paths in images. Although, in all our experiments, we did not allow paths to cross, it is possible to allow paths to cross. However, the search cost will increase by a factor of $K!$.

The algorithm developed in this paper can be used in a number of applications where multiple paths are sought. These paths can be horizontal paths, vertical paths, or circular paths. In most parts of this paper, we are talking about multiple paths from the left of the image to the right of the image, i.e., horizontal paths. If vertical paths are to be obtained, a rotation of the input (and output) image is needed so that the horizontal paths extraction algorithm can be applied. For circular paths, a Cartesian to polar coordinate (and a reverse for the result) transformation is necessary.

The objective of our algorithm is to obtain multiple paths where the sum of pixel values on these paths is minimal. In some applications where multiple paths are clearly defined and well separated, a single shortest path algorithm may be used by applying it several times on the image. But, this simple approach will be error prone for paths which are less well-defined. Figs. 10b and 15c show examples where successive application of a single shortest path algorithm does not work, whereas our CET algorithm produces the desired results.

For closed object boundary extraction, the input image is first transformed from Cartesian to polar coordinate. Then, multiple horizontal paths are extracted using our CET algorithm. If the path features are clearly present in the image, direct application of the CET algorithm may produce closed boundaries. If the features are not well-defined, there are several ways to ensure the endpoints of the multiple paths meet, as mentioned in Section 2.2 for single shortest path. We can use similar techniques to obtain multiple closed object

boundaries. The simplest and the most efficient technique to use is the Image Patching Algorithm or a combination of the Image Patching Algorithm with the Multiple Back-Tracking Algorithm developed in [7]. In the Image Patching Algorithm, the input image (in our case, the polar transformed image) is patched on the left and the right sides with portions of the input image itself, say with one-fourth of the image width. Then, the CET algorithm can be applied to the patched image for obtaining paths that should be closed. For a detailed description of the Image Patching Algorithm and related algorithms, please see [7].

7 Conclusions

Shortest path or K-best paths algorithms have found many applications in several disciplines. We have proposed a new algorithm for finding multiple paths in images using a constrained expanded trellis (CET). With our method, we are able to easily incorporate several constraints into our algorithm. These constraints help to regulate the shape of the paths and reduce the computational cost significantly. A list of possible multiple paths can also be obtained. We give several application examples for feature extraction and object segmentation in images. Our CET algorithm could be extended to 3D images for linear features although the resulting search space may be large.

Acknowledgments

The authors are grateful to the anonymous reviewers for their very constructive comments. They also thank Dr. Mark Berman and Ms. Leanne Bischof of CSIRO Mathematical and Information Sciences for their comments on this paper. The cardiac MR image in Fig. 14 (last row) was from Jens C. Nilsson and Bjørn A. Grønning, Danish Research Centre for Magnetic Resonance (DRCMR) [25].

References

[1] C. Sun and S. Pallottino, "Circular Shortest Path on Regular Grids," *Proc. Asian Conf. Computer Vision,* pp. 852-857, Jan. 2002.
[2] M. Barzohar and D.B. Cooper, "Automatic Finding of Main Roads in Aerial Images by Using Geometric-Stochastic Models and Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 7, pp. 707-721, July 1996.
[3] S. Lloyd, "Stereo Matching Using Intra- and Inter-Row Dynamic Programming," *Pattern Recognition Letters,* vol. 4, pp. 273-277, Sept. 1986.
[4] Y. Ohta and T. Kanade, "Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 7, no. 2, pp. 139-154, Mar. 1985.
[5] C. Sun, "Fast Stereo Matching Using Rectangular Subregioning and 3D Maximum-Surface Techniques," *Int'l J. Computer Vision,* vol. 47, nos. 1/2/3, pp. 99-117, Apr.-June 2002.
[6] C. Sun, "Fast Optical Flow Using 3D Shortest Path Techniques," *Image and Vision Computing,* vol. 20, no. 13/14, pp. 981-991, Dec. 2002.
[7] C. Sun and S. Pallottino, "Circular Shortest Path in Images," *Pattern Recognition,* vol. 36, no. 3, pp. 711-721, Mar. 2003.
[8] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Englewood Cliffs, N.J.: Prentice Hall, 1993.
[9] Q.-P. Gu and S. Peng, "An Efficient Algorithm for the k-Pairwise Disjoint Paths Problem in Hypercubes," *J. Parallel and Distributed Computing,* vol. 60, no. 6, pp. 764-774, June 2000.
[10] R. Perry, A. Vaddiraju, and K. Buckley, "Trellis Structure Approach to Multitarget Tracking," *Proc. Sixth Ann. Workshop Adaptive Sensor Array Processing,* Mar. 1999.
[11] W.-T. Chan and F.Y. Chin, "Efficient Algorithms for Finding the Maximum Number of Disjoint Paths in Grids," *J. Algorithms,* vol. 34, no. 2, pp. 337-369, Feb. 2000.
[12] S.-W. Lee and C.-S. Wu, "A K-Best Paths Algorithm for Highly Reliable Communication Networks," *IEICE Trans. Comm.,* vol. E82-B, no. 4, pp. 586-590, Apr. 1999.
[13] D.A. Castañon, "Efficient Algorithms for Finding the K Best Paths through a Trellis," *IEEE Trans. Aerospace and Electronic Systems,* vol. 26, no. 2, pp. 405-410, Mar. 1990.
[14] J.K. Wolf, A.M. Viterbi, and G.S. Dixon, "Finding the Best Set of K Paths through a Trellis with Application to Multitarget Tracking," *IEEE Trans. Aerospace and Electronic Systems,* vol. 25, no. 2, pp. 287-296, Mar. 1989.
[15] S.D. Nikolopoulos and G. Samaras, "Sub-Optimal Solutions to Track Detection Problem Using Graph Theoretic Concepts," *J. Systems Architecture* vol. 42, no. 9/10, pp. 743-760, Feb. 1997.
[16] M. Buckley and J. Yang, "Regularised Shortest-Path Extraction," *Pattern Recognition Letters,* vol. 18, no. 7, pp. 621-629, July 1997.
[17] P. Bamford and B. Lovell, "Unsupervised Cell Nucleus Segmentation with Active Contours," *Signal Processing,* special issue on deformable models and techniques for image and signal processing, vol. 71, no. 2, pp. 203-213, Dec. 1998.
[18] L.D. Cohen and R. Kimmel, "Global Minimum for Active Contour Models: A Minimal Path Approach," *Int'l J. Computer Vision,* vol. 24, no. 1, pp. 57-78, Aug. 1997.
[19] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic Active Contours," *Int'l J. Computer Vision,* vol. 22, no. 1, pp. 61-79, Feb./Mar. 1997.
[20] C. Leung, B. Appleton, and C. Sun, "Fast Stereo Matching by Iterated Dynamic Programming and Quadtree Subregioning," *Proc. British Machine Vision Conf.,* A. Hoppe, S. Barman, and T. Ellis, eds., vol. 1, pp. 97-106, Sept. 2004.
[21] Y. Boykov and V. Kolmogorov, "Computing Geodesics and Minimal Surfaces via Graph Cuts," *Proc. Int'l Conf. Computer Vision,* vol. I, pp. 26-33, Oct. 2003.
[22] B. Appleton and C. Sun, "Circular Shortest Paths by Branch and Bound," *Pattern Recognition,* vol. 36, no. 11, pp. 2513-2520, Nov. 2003.
[23] B. Appleton and H. Talbot, "Globally Optimal Geodesic Active Contours," *J. Math. Imaging and Vision,* vol. 23, no. 1, pp. 67-86, July 2005.
[24] R. Sedgewick, *Algorithms in C++,* third ed., nos. 1-4, Addison-Wesley, 1998.
[25] M.B. Stegmann, R. Fisker, and B.K. Ersboll, "Extending and Applying Active Appearance Models for Automated, High Precision Segmentation in Different Image Modalities," *Proc. 12th Scandinavian Conf. Image Analysis,* vol. 1, pp. 90-97, June 2001.

Changming Sun received the PhD degree in the area of computer vision from Imperial College of Science, Technology, and Medicine, London in 1992. Then, he joined CSIRO Mathematical and Information Sciences, Australia, where he is currently a principal research scientist carrying out research and working on applied projects. His research interests include computer vision and photogrammetry, image analysis, pattern recognition, and bioinformatics. He has served on the program/organizing committees of various international conferences. Dr. Sun is a member of the Australian Pattern Recognition Society.

Ben Appleton received the PhD degree in the area of image analysis from the University of Queensland (UQ), Australia, in 2005. He received degrees in engineering and in science from the University of Queensland in 2001 and was awarded a university medal. He is currently a research fellow in the Electromagnetics and Imaging Research Group of UQ. He has contributed 16 research papers to international journals and conferences and was awarded the prize for Best Student Paper at Digital Image Computing: Techniques and Applications (DICTA) 2003. His research interests include image segmentation, stereo vision, and cardiac modeling.