Circular Shortest Path in Images

Changming Sun
CSIRO Mathematical and Information Sciences, Locked Bag 17, North Ryde, NSW 1670, Australia.
changming.sun@csiro.au

Stefano Pallottino
Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56125 Pisa, Italy.
pallo@di.unipi.it

***Abstract*— Shortest path algorithms have been used in a number of applications such as crack detection, road or linear feature extraction in images. There are applications where the starting and ending positions of the shortest path need to be constrained. In this paper, we present several new algorithms for the extraction of a circular shortest path in an image such that the starting and ending positions coincide. The new algorithms we developed include multiple search algorithm, image patching algorithm, multiple backtracking algorithm, the combination of image patching and multiple back-tracking algorithm, and approximate algorithm. The typical running time of our circular shortest path extraction algorithm on a 256$\times$256 image is about 0.3 seconds on a rather slow 85MHz Sun SPARC computer. A variety of real images for crack detection in borehole data, object boundary extraction, and panoramic stereo matching have been tested and good results have been obtained.**

***Keywords:* Circular shortest path, Dynamic programming, Multiple search algorithm, Image patching algorithm, Multiple backtracking algorithm, Combination algorithm, Approximate algorithm.**

I. INTRODUCTION

In a weighted graph or network, it is frequently desired to find a shortest path between two nodes. The shortest path is defined as a path from one node to the other such that the sum of the weights of the arcs on the path is minimised. Most algorithms or applications in the graph framework use a labeling approach, in particular the one due to Dijkstra [1], [2].

Buckley and Yang developed a regularised shortest path extraction algorithm for rectangular images [3]. The algorithm uses dynamic programming (DP) techniques for shortest path extraction. They applied their algorithm to borehole data/image for crack detection and also to satellite images for road extraction. The shortest path they are interested in is either the path running from top to bottom or left to right. There is no constraint on the starting and ending positions of the path. A number of authors used dynamic programming technique to obtain a shortest path in a rectangular matrix for stereo disparity measurement [4], [5], [6], [7]. All these applications impose no constraints on the starting and ending positions of the shortest path.

In the applications of inspection of open or equipped boreholes, borehole geophysicists record and analyse measurements of physical properties made in test holes. Probes that measure different properties are lowered into the borehole to collect continuous or point data that is graphically displayed as a geophysical log. Borehole geophysics is also used in groundwater and environmental investigations to obtain information on well construction, rock lithology and fractures, permeability and porosity, and water quality [8], [9]. These applications include fracture identification and orientation, stratigraphic and structural dip analysis. The borehole acoustic televiewer resembles an optical television camera system in producing a full 360° image of the borehole walls. The signals are presented as continuous images. In certain situations, the input images wrap around, e.g. the left and the right edges are actually neighbouring columns. Figure 1(a) shows a full 360° borehole image with cracks in it. As the image is a 360° circular image, the left and the right boundaries of the image are actually neighbouring columns. This image can be shown in a cylindrical format as in Figure 1(b). In the example shown in the figure, a closed or circular shortest path should be extracted. That is the starting and the ending positions of the path should be at neighbouring points.

In some image analysis applications, object boundaries need to be extracted [10]. In these applications, it is necessary to make sure that the boundary extracted are closed contours. Panoramic stereo images are becoming available for 3D applications. In 360° panoramic stereo images, the left and the right columns are connected with each other. Therefore in the stereo matching process, it is necessary to take this constraint into account. This can be achieved by obtaining a circular path in the correlation coefficient matrix.

In this paper we address the issue of obtaining a circular shortest path in an image or a regular grid for a number of applications. The rest of the paper is organised as follows: Section II gives a brief review for ordinary shortest path extraction algorithms. Section III presents five new methods for circular shortest path extraction on images. Section IV shows the experimental results obtained using our circular shortest path extraction methods applied to a variety of applications. Section V gives concluding remarks.

Research partially supported by grants: MURST and INDAM-GNAMPA of Italy.
Corresponding author.

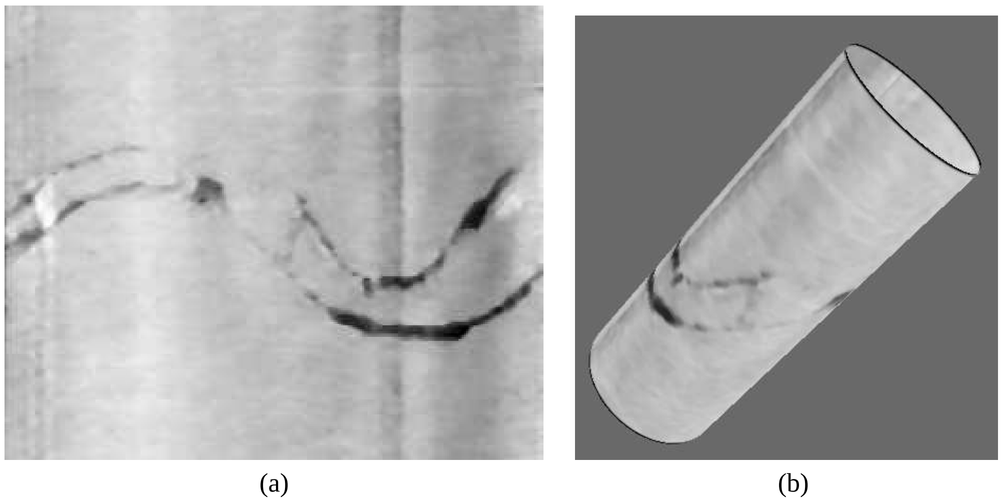


Fig. 1. (a) A 360° borehole image with cracks; (b) Image in (a) shown in a cylindrical format.

II. Ordinary Shortest Path Algorithms: A Brief Review

This section gives a brief review on ordinary shortest path extraction algorithms using labeling algorithms and dynamic programming. The problem is to find a path from the left side to the right side of an image or grid such that the cost of the path is minimum. The cost of the path is the sum of the costs along the path. As an example, Figure 2 illustrates the possible positions that a path from left to right can go from a point to its neighbours on the grid. "B" is a point on the top boundary of the image; and "B2" and "B3" are the possible positions a path can arrive from point "B". The cost of an arc in an image or regular grid is defined as the value of its starting position. For the arc connecting "B" and "B3", the cost of the arc is the image value at position "B". If a point is not on the top or the bottom boundary (e.g. "A"), there will be three possible positions ("A1", "A2", "A3") that the path can go from point "A". If the top and bottom rows are also neighbours, "B1" is also a possible position for a path to reach from point "B". If only the left and the right columns of the grid are neighbours, we can wrap the grid on to a cylindrical surface, so that the left and the right columns are connected. If the top and the bottom rows are also neighbouring rows, we can imagine to bend the cylindrical shape to make the top and bottom touch, so that a toroid shape is formed.

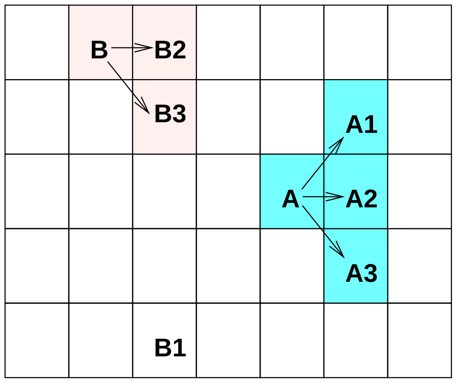


Fig. 2. Neighbour points that a path can go from "A" and "B" to certain points in the next column.

A. Shortest Path Extraction Algorithm

A huge number of scientific papers are devoted to the shortest path problem. We refer to [1], [2] for a general view of the problem and of the most efficient algorithms proposed.

Let $G = (N, A)$ be a directed graph, where N is the *set of nodes* of cardinality n, and A is the *set of arcs* of cardinality m. With each arc $(i, j) \in A$, a cost c_{ij} is associated. Given an *origin*, or *root*, r, the *shortest path tree problem* consists of finding a spanning directed tree T^* rooted at r such that, for each $i \in N$, the path from r to i in T^* is a minimum cost path in G. The tree T^* is a particular collection of shortest paths P_{ri}, for every $i \neq r$. Given the *root*, r, and the destination node s the *single pair shortest path problem* consists of finding a minimum cost path P_{rs} from r to s in G.

The single pair shortest path problem is not easier than the shortest path tree problem, at least for the computational analysis; in fact, in the worst case, to find P_{rs} one has to build the entire tree T^*. Almost all the algorithms have a common scheme: starting from an initial tree T, the algorithms iteratively update it until T^* is found. To each node $i \in N$, a *label* d_i is associated, providing the cost of the path from r to i, belonging either to the current tree or to a previous one; for that, d_i is in general an upper bound of the cost of the path in the current tree T.

The labels allow to compare pair of paths; let (i, j) be an arc and P_{ri} and P_{rj} be two paths of cost d_i and d_j, respectively. The cost of path $P_{ri} \cup \{(i, j)\}$ is $d_i + c_{ij}$; if the two costs are such that

$$d_i + c_{ij} \geq d_j,$$

we say that the *Bellman's condition* for arc (i, j) is satisfied; otherwise $P_{ri} \cup \{(i, j)\}$ results to be better than P_{rj}, and by changing the predecessor node of node j, say $p(j)$, in the tree T, with the node i, we remove from T the arc $(p(j), j)$ and we insert (i, j).

A node i for which it is possible that the Bellman's conditions are not satisfied for all the arcs (i, j) outgoing from it is said a *scan eligible node*. The algorithms handle a *set of scan eligible* nodes Q in such a way that the algorithm iteratively select one of its node to check all the outgoing arcs to try to improve the current tree T. It is proved that, when the Bellman's conditions are satisfied for all the arcs, then the current tree T is the minimum cost one.

A typical iteration is the following:

```
select and remove a node $i$ from $Q$
for each $(i, j)$ outgoing from $i$ such that $d_i + c_{ij} < d_j$ do
    begin
        $d_j := d_i + c_{ij}$;
        $p(j) := i$;
        if $j \notin Q$ then $Q := Q \cup \{j\}$
    end
```

The *Dijkstra-like* approach is obtained when the node selected at the beginning of each iteration is the one with the minimum label among the nodes belonging to Q:

$$i = \text{argmin}\{d_v : v \in Q\}.$$

When the arc costs are non-negative, the Dijkstra's selection ensures that every node will be inserted into, and removed from, Q exactly once; so, the total number of checks of the Bellman's conditions is m. The costly operation is the selection of the minimum label node i which has to be repeated n times. To speed-up these operations, special data structures, such as *heaps* and/or *buckets*, are used to implement Q.

The Dijkstra-like approach is particularly suitable for the single pair shortest path problem; in fact, it is easy to prove that once the destination node is selected from Q, the shortest path P_{rs} has been obtained. Another possible speed-up is to work with path searches in parallel, the first starting from r and the other starting from s and moving "back" along the arcs: when a node has been selected from both the scan eligible sets, with few additional operations the minimum cost path is obtained [11].

A special method is used when the directed graph G is *acyclic*, i.e. when the nodes $i \in N$ can be re-numbered in such a way that for every arc $(i, j) \in A$ it is $i < j$. In this case the shortest path problem is easily solved by examining the nodes according to the natural order $i = 1, \ldots, n$, and for each of them apply the following typical iteration:

```
for each $(i, j)$ outgoing from $i$ such that $d_i + c_{ij} < d_j$ do
    begin
        $d_j := d_i + c_{ij}$;
        $p(j) := i$
    end
```

Regardless of the sign of the arc costs, the number of operations grows linearly with m, i.e. the number of arcs. So, it is the most efficient algorithm for acyclic graphs.

The computation shown above is organised in a "forward" form. That is for a given node i, all the nodes in the next layer connected to i are checked and, possibly, updated. The same computation, following the same nodes order, can be done in "backward" form, by directly applying the *Bellman's equation*:

$$d_j := \min\{d_i + c_{ij} : (i, j) \text{ enters } j\}, \quad j = 2, \ldots, n. \tag{1}$$

The "backward" approach on acyclic graphs leads to the classical dynamic programming algorithm. In later sections of the paper, we will use the "backward" approach, i.e. the dynamic programming algorithm, for shortest path extraction.

B. The Grid Structure

The shortest path problem in the grid is a problem on an acyclic graph. In fact, from an array of u rows and v columns, we can derive a directed graph $G = (N, A)$, where each node $i \in N$ is a pair $[h(i), k(i)]$ where $h(i)$ indicates the row and $k(i)$ indicates the column of the element represented by i. The number of nodes is $n = uv$.

An arc $(i, j) \in A$ exists if $k(j) = k(i) + 1$ and $h(j) = h(i) + \alpha$, where $\alpha = -1, 0, 1$ but the extreme cases in which $h(i) = 1$ or $= u$; the cost of arc (i, j) is set to the entry associated to the pair $[h(i), k(i)]$, i.e. all the arcs leaving node i have the same cost, and it is the value at position $[h(i), k(i)]$. The total number of arcs is $m = (3u - 2)(v - 1) < 3n$ if the top and the bottom rows are not connected. If the top and the bottom rows are connected, $m = 3u(v - 1) < 3n$.

With this transformation the shortest path problem on the grid is mapped into a shortest path problem on a classical graph. By using this transformation, we can see that our problem has another special characteristic: the graph G is a *stable acyclic sequential layered graph*. A graph is *stable* and *layered* if the set of nodes N can be partitioned into subsets (layers) such that there exist arcs between nodes belonging to different layers and do not exist arcs between nodes belonging to the same layer; moreover, it is *sequential* if the layers can be ordered in a sequence $\{L_1, L_2, \ldots, L_v\}$ in such a way the arcs connect two adjacent layers. Finally, it is *acyclic* when the arcs go only from nodes of a layer to nodes belonging to the following layer. Our graph has indeed these properties (and others that we will exploit later), in fact each column of the grid is a layer and the arcs go from one column (layer) to the following one.

Let us suppose that we have to solve the shortest path problem from the nodes in layer L_1 to the nodes in the last layer L_v. The best approach is to exploit the characteristics of the graph by analysing one layer at a time and, for each node j of that layer, by setting the optimal label value d_j by applying the Bellman's equation (1). The resulting algorithm, based on the direct application of the Bellman's equation (1), is

```
Procedure Layer_Grid():
  begin {initialising the labels of nodes of $L_1$}
      for each $j \in L_1$ do
          begin
              $d_j := 0$;
              $p(j) :=$ nil;
          end
      for $h := 2$ to $v$ do
```

```
        for each $j \in L_h$ do
            begin {working on layer $L_h$}
                $d_j := \min\{d_i + c_{ij} : (i, j) \text{ enters } j\}$;
                $p(j) := \text{argmin}\{d_i + c_{ij} : (i, j) \text{ enters } j\}$
            end
    end.
```

III. Circular Shortest Path

The algorithms described in the previous section for ordinary shortest path extraction impose no constraint about the starting and ending position of the path. From now on, we assume that the image or grid is circular, that is the first and the last columns are neighbours. In this section, we will present several new algorithms for circular shortest path (CSP) extraction where the starting and the ending positions of the obtained path are connected. A circular path is a path from the first column to the last column when the starting and ending position are connected. A circular shortest path is a circular path when its cost among all the circular paths are minimum. Figure 3 gives an example showing the different paths obtained using the ordinary shortest path and a circular shortest path extraction algorithm (to be described later). The "*" symbols in the figure indicate the positions of the shortest paths. The cost for the ordinary shortest path is 666, while the cost for the circular shortest path is 702. The path obtained using the ordinary shortest path technique is shown in Figure 3(a). The positions of the paths are: (Ba) $\rightarrow$ (Bb) $\rightarrow$ (Bc) $\rightarrow$ (Cd) $\rightarrow$ (De) $\rightarrow$ (Df) $\rightarrow$ (Dg) $\rightarrow$ (Dh) $\rightarrow$ (Ei) $\rightarrow$ (Fj) $\rightarrow$ (Ek) $\rightarrow$ (Fl). The capital letters indicate row numbers and the lower case letters indicate column numbers. The starting position (Ba) and the ending position (Fl) are not directly connected. The path obtained using the circular shortest path technique is given in Figure 3(b). The positions of the paths are: (Da) $\rightarrow$ (Eb) $\rightarrow$ (Ec) $\rightarrow$ (Fd) $\rightarrow$ (Ee) $\rightarrow$ (Df) $\rightarrow$ (Dg) $\rightarrow$ (Dh) $\rightarrow$ (Ei) $\rightarrow$ (Fj) $\rightarrow$ (Ek) $\rightarrow$ (El). The starting and ending positions (Da) and (El) are actually neighbouring points.

A. Multiple Search Algorithm

To find the required circular shortest path, one can run the ordinary shortest path algorithm for acyclic graphs u times (u is the number of rows in the image or grid), one for each node $[h, 1]$ of the first column as origin. Once computed the shortest paths for all the nodes of the last column, we select as the best circular shortest path the least cost path among the ones terminating at the nodes $[h-1, v]$, $[h, v]$ and $[h+1, v]$ (to satisfy the constraint that the starting and ending positions are neighbours). At the end of the u shortest path computations, we can select the path with the least cost to be our result. This is our multiple search algorithm (MSA).

Figure 4 shows that for any particular given position "C" on the left boundary of the image, the three possible ending positions which are essentially neighbours of "C" are "C1", "C2" and "C3". If the dark positions of the image are assigned to have large values, ordinary dynamic programming technique can be used to find the required shortest path starting from point "C" and ending at one of its "neighbouring" points ("C1", "C2" or "C3"). We also need to change point "C" and "C1", "C2" and "C3" to all the other positions on the left column and right column, and find all the corresponding paths. Each of the path has a cost associated with it. The one which has the minimum cost is the path that we want to extract.

The steps of our MSA algorithm for circular shortest path extraction are:

1) Set the start position "C" from the top row of the image.
2) Assign special values to the dark positions of the left and the right boundaries of the input image as shown in Figure 4.
3) Perform ordinary shortest path extraction using DP on the modified image.
4) Record the cost of the path, and select the current least cost path as the result.
5) Move to the next row of the image and go to Step 2 unless the current row is the last.
6) Display circular shortest path.

This method will guarantee to find the path which satisfies our constraints. The disadvantage of this method is that the ordinary shortest path algorithm for acyclic graphs has to be run u times. Therefore it has a time complexity $O(u^2 v)$.

B. Image Patching Algorithm

In this subsection we will present a fast algorithm for obtaining the required circular shortest path by working with patched images. We call this the image patching algorithm (IPA).

The size of the patches could depend on the type of applications or the content of the images. If an image contains strong circular paths, the starting and ending positions of a shortest path obtained by just using the ordinary dynamic programming technique may not be too far from the constrained circular shortest path. In this situation, the size of the patches needed can be small. Otherwise, if an image contains weak circular paths, the starting and ending positions of a shortest path obtained by just using ordinary dynamic programming technique may be far from a constrained circular shortest path. In this case, a stronger constraint, or larger size of patches may be needed.

Figure 5 shows the image patching process for obtaining the circular shortest path. Patch-1 and Patch-2 are parts of the original image. Copy-of-Patch-1 and Copy-of-Patch-2 are copies of the image regions Patch-1 and Patch-2. These two copies of the local image regions are attached to the original image to build a larger image. Image patching is only performed in the X-direction of the image, as we need to find the circular shortest path from left to right of the image. If a shortest path from top to bottom of the image is needed, the patching can be done at the top and the bottom of the image. Figure 5(a) illustrates the patching process, and Figure 5(b) is an example of a patched image. Dark lines are drawn in Figure 5(b) to show the image boundaries. The left side of the first dark line is the same region for Patch-2. The right side of the second dark line is the same region for Patch-1.

The image patching method does not guarantee to find the required path. However many synthetic and real image tests all produce correct results. If a circular shortest path

	a	b	c	d	e	f	g	h	i	j	k	l
A	252	245	74	13	219	171	193	242	17	203	72	123
B	*160	*83	*23	103	71	214	174	30	31	58	197	117
C	202	117	157	*90	202	102	104	235	104	6	215	99
D	161	143	231	127	*20	*20	*63	*39	180	178	69	108
E	210	63	64	192	28	36	130	170	*29	246	*54	84
F	235	83	129	66	233	237	41	147	190	*31	140	*54
G	176	208	255	0	173	17	252	205	197	212	131	245

(a)

	a	b	c	d	e	f	g	h	i	j	k	l
A	252	245	74	13	219	171	193	242	17	203	72	123
B	160	83	23	103	71	214	174	30	31	58	197	117
C	202	117	157	90	202	102	104	235	104	6	215	99
D	*161	143	231	127	20	*20	*63	*39	180	178	69	108
E	210	*63	*64	192	*28	36	130	170	*29	246	*54	*84
F	235	83	129	*66	233	237	41	147	190	*31	140	54
G	176	208	255	0	173	17	252	205	197	212	131	245

(b)

Fig. 3. Examples showing the different paths obtained using ordinary and circular shortest paths. The values in the table are randomly generated. It is assumed that the column "a" and column "l" are neighbouring columns. (a) Shortest path without constraint. The starting and ending positions (Ba) and (Fl) are not neighbours. (b) Shortest path with constraint, i.e. circular shortest path. The starting and ending positions (Da) and (El) are neighbours.

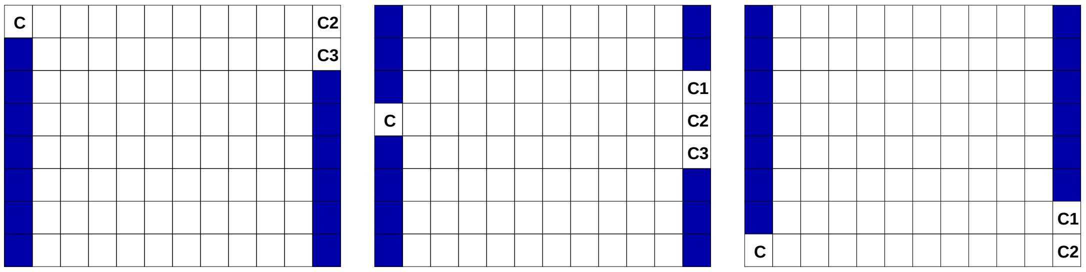


Fig. 4. Constraining the ordinary shortest path search.

is not found, we can iterate the process of finding circular shortest path by using a different size of the patch, or using a multiple back-tracking algorithm (MBTA) to be described in the following subsection. Or if the application is not time critical, the MSA method can be used. The main advantage of the IPA algorithm is its speed, as it only needs one run of the ordinary shortest path extraction algorithm on the patched image. The complexity of the algorithm is $O(u(v+k))$, where k is the width of the added patches.

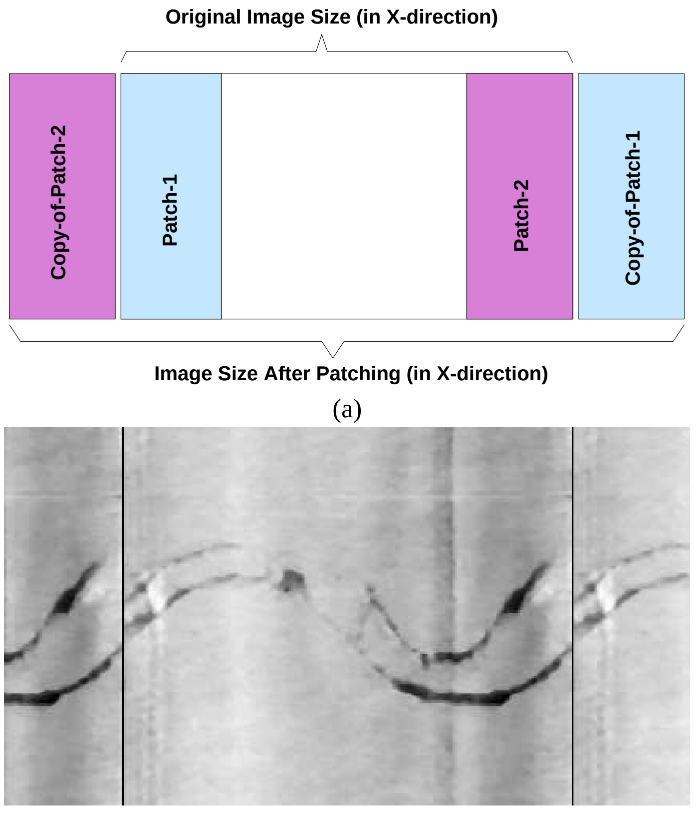
The steps of our image patching algorithm for circular shortest path extraction are:

1) Patch the input image on the left and the right sides with portions of the input image itself (say one-eighth of the image width). The size of the patches depends on the application. If the path is strong or very clear in the image, the size of the patches can be smaller. Otherwise, the width of each patch can be set to half of the width of the input image.
2) Perform ordinary shortest path extraction using DP on the patched image.
3) Extract the shortest path which lies inside the original image.
4) Check if the obtained path satisfy the circular constraint. If so, go to Step 5; otherwise, go to Step 1 with a different patching size, or using MBTA or MSA.
5) Display circular shortest path.

C. Multiple Back-tracking Algorithm

In this subsection we will present another algorithm based on the ordinary shortest path algorithm by performing multiple back-tracking. We call it the multiple back-tracking algorithm (MBTA). When carrying out the ordinary shortest path extraction using dynamic programming, we have in storage the cost value for each node and the corresponding predecessor matrix. From each node on the last column, we can back-track a path from this node to a certain node on the first column. This path has a certain cost. If the starting and the ending positions of this path are neighbours, then we say this path is a possible CSP. We back-track all the nodes on the last column, and we may find several possible CSPs. We can then choose the CSP with the minimum cost as the final result. We have found that on an image or regular grid one can almost always find a circular path although this circular path may not be the circular shortest path.

To speed-up the phase of checking whether a path is a circular one, it is possible to associate to every node $[h, k]$, together with the label $d_{h,k}$ and the predecessor node $p(h, k)$, also the knowledge of the first node $[r', 1]$ of the current path


Original Image Size (in X-direction)
Copy-of-Patch-2
Patch-1
Patch-2
Copy-of-Patch-1
Image Size After Patching (in X-direction)
(a)

(b)

Fig. 5. Image patching for fast circular shortest path extraction. (a) Drawings showing the patching process; (b) Illustration using a real image. Dark lines in this image are artificial. It is used to show the region boundaries.

ending into $[h, k]$. In fact, we introduce, for every node $[h, k]$, the *first node function* $f(h, k)$. To do that, it is enough to initialise $f(h, 1) = h, h = 1, \ldots, u$, and each time the label of $[h, k]$ is improved from a node $[h', k-1]$, it is enough to set, together the predecessor $p(h, k) := h'$, also the first node $f(h, k) := f(h', k-1)$.

Then, at the end, when the minimum label node $[s', v]$ is selected, the first node $[r', 1]$, with $r' = f(s', v)$, is available without moving back along the minimum cost path.

Figure 6 shows the results of the circular shortest path extraction using the multiple backtracking algorithm. Figure 6(a) is the input random image; Figure 6(b) is a matrix (first node function) containing information for each pixel as to which point in the first column it is connected to; and Figure 6(c) shows the CSP obtained overlaid on the input image.

It can be observed from Figure 6(b) that with the increase of the column index of the image, the number of colours decreases. This means that the possible number of circular shortest path is reduced. For a very thin image, the shape of the first column matrix is similar to that of the left part of Figure 6(b). One may have many possible circular shortest path. For a very long image, the number of colours connected to the right edge of the image is smaller. Therefore the possible number of circular shortest path is smaller. The MBTA algorithm guarantees to find a circular path. But this circular path may not be the circular shortest path.

The steps for our MBTA algorithm are:

1) Carry out ordinary dynamic programming to build the cost matrix, and the predecessor matrix.
2) Carry out back-tracking from each node on the last column and record the cost for a circular path.
3) Choose a circular path with the minimum cost as the result of this algorithm.

D. Combination Algorithm

The IPA algorithm provides a fast way of finding "circular" shortest paths. But the path obtained is not always circular. The MBTA algorithm guarantees to find a circular path, although this path may not be the circular shortest path. We can combine these two algorithms as the IP&MBTA algorithm to increase the chance of finding the true circular shortest path in an image or grid. This will involve running each of the IPA and MBTA algorithms once. That is using the IPA algorithm to find a path, if this path is not circular, we use the path obtained by the MBTA algorithm. If the path obtained from running the IPA algorithm is circular, we choose the path with the minimum cost from the IPA and the MBTA algorithms. Many real images tests have shown that the combination algorithm produces all the correct circular shortest path.

E. Approximate Algorithm

In most real cases it is enough to heuristically find a circular path, not necessarily optimum, to correctly process the image. The key point is to guarantee that the best circular path found by the heuristic is not far, in terms of "cost", from the optimal circular path. To ensure that the "sub-optimal path" is "good enough" we would limit the relative error.

More formally, let z^* (> 0) be the (unknown) cost of the optimal circular path and let $z(A)$ the cost of the best path $P(A)$ found by a given approximate algorithm A. Of course $z(A) \geq z^*$. We say that the *relative error* $E(A)$ of path $P(A)$ is:

$$E(A) = \frac{z(A) - z^*}{z^*}.$$

To define a bound on the relative error it is enough to find a so-called "lower bound" of the unknown optimal solution, i.e. a not necessarily feasible solution whose cost z' is such that $z' \leq z^*$. If that solution is a feasible circular path, then it is an optimal solution for the problem. By knowing $z(A)$ and z' we have a "threshold value":

$$B(A) = \frac{z(A) - z'}{z'},$$

which bounds the relative error of path $P(A)$, i.e. $E(A) \leq B(A)$.

To find a solution which is a lower bound for the circular path problem is enough to solve the shortest path problem from any node of the first column to any node of the last column. That path may not be circular, and in this case it cannot be taken as a solution; nevertheless, its cost z' is a lower bound for the optimal solution.

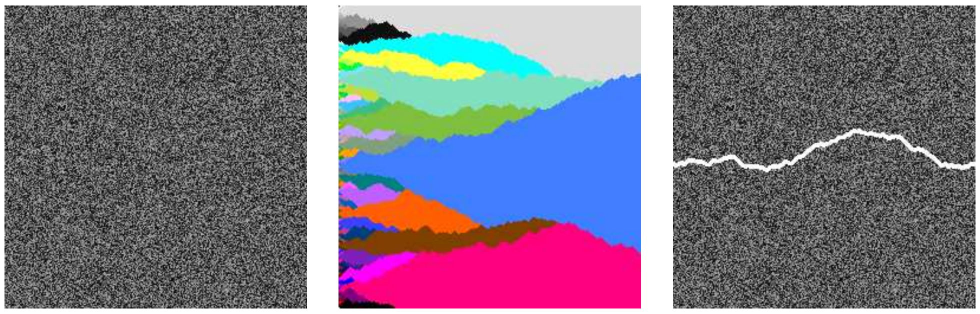


Fig. 6. A random image and its circular shortest path. (a) a random image; (b) A matrix showing each pixel connected to a point on the first column; and (c) CSP overlaid on the random image.

To find such a path it is enough to apply the shortest path algorithm for acyclic graphs by setting to zero the label associated to every node $[h, 1]$ of $L_1, h = 1, \ldots, u$, and to explore all of their outgoing arcs. Once examined all the nodes of all the layers, the *shortest path* can be found by inspection of the last layer L_v: we select the node $[s', v]$ with minimum label:

$$[s', v] = \text{argmin}\{d_{i,v} : i = 1, \ldots, u\}.$$

Then $z' = d_{s',v}$. Moving back, through the predecessor function, from $[s', v]$ to the origin node in the first layer, say $[r', 1]$. If the path from $[r', 1]$ to $[s', v]$ is a circular one, that is, if $|r' - s'| \leq 1$, then the relative path is the minimum cost circular path; so, the optimal solution for our problem has been found. Otherwise, z' results to be a lower bound of the optimal solution value.

To build an approximate circular path we suggest firstly to check among the shortest paths already generated and, if necessary, to solve other shortest path problems to extract a circular path whose cost results are of enough quality to be selected as an approximate solution.

More in detail, as far as the shortest paths used to set nodes s' and $r' = f(s', v)$, since the shortest path from r' to s' is not circular, we repeat for every other node $[h, v]$ the same check $|f(h, v) - h| \leq 1$, that is to check whether a circular path has been created. If there exists at least one circular path, we store the best one, which is not necessarily the optimum one. Let call $P(A)$ that path and denote by $z(A)$ its cost. Now we can evaluate the bound $B(A)$ for the relative error (that bound is ∞ if no circular paths exist).

If that bound fits with the pre-defined accuracy that we require for a circular path to be chosen as an approximate solution for our problem, we will use $P(A)$. Otherwise, we select a given starting node $[\bar{h}, 1]$ and we re-compute the shortest path tree from that node. At the end, we select the minimum label node amongst $[\bar{h} - 1, v], [\bar{h}, v]$ and $[\bar{h} + 1, v]$; this gives the best circular path having $[\bar{h}, 1]$ as starting node. Again, if the path passes the required quality test, it is chosen as $P(A)$, otherwise another starting node $[\bar{h}, 1]$ is chosen and the process is repeated. In the case of t iterations, the time complexity of the approximate algorithm is $O(tuv)$.

The approximate algorithm can be viewed as a special case of MSA or MBTA when the search for the circular shortest path can stop early. The predefined approximation accuracy may not always achieved even all nodes in the last column are checked in the Approximate Algorithm. This is particular true when the gap between the cost of the shortest path and the cost of the circular shortest path is large.

IV. Experimental Results

This section shows some of the results obtained using the methods described in this paper. A variety of images have been tested, including synthetic images and different types of real images.

A. Borehole Data

To compare the different shapes of shortest path obtained using the ordinary and the circular shortest path algorithms, ordinary dynamic programming algorithm is applied to the borehole image shown in Figure 1(a). The ordinary shortest path obtained is shown in Figure 7(a). The result of our circular shortest path obtained by using the combination algorithm is given in Figure 7(c). Notice the position difference of the shortest paths close to the left edges of Figure 7(a) and (c). Figure 7(b,d) show the 360° version of the flat 2D images. It is clear that the path obtained using the circular shortest path method has the same starting and ending position as shown in Figure 7(d) while the path obtained using the ordinary shortest path extraction technique does not join up together at the starting and the ending positions as shown in Figure 7(b).

B. Boundary Detection

du Buf *et al* described their first results on diatom contour extraction in [10]. In a preprocessing step initial contours are extracted using a conventional edge-following algorithm like Canny's. The object contours are extracted by using the best-fitting ellipse and a subsequent contour following in the elliptical polar-transformed image. They applied a depth-first search algorithm which evaluates the grey level changes along the path in the polar-transformed image.

Similar to du Buf *et al*'s algorithm, we obtain some initial positional information about a closed contour. In our case, however, we only need to know the approximate position of the contour. Then the input image is transformed into the

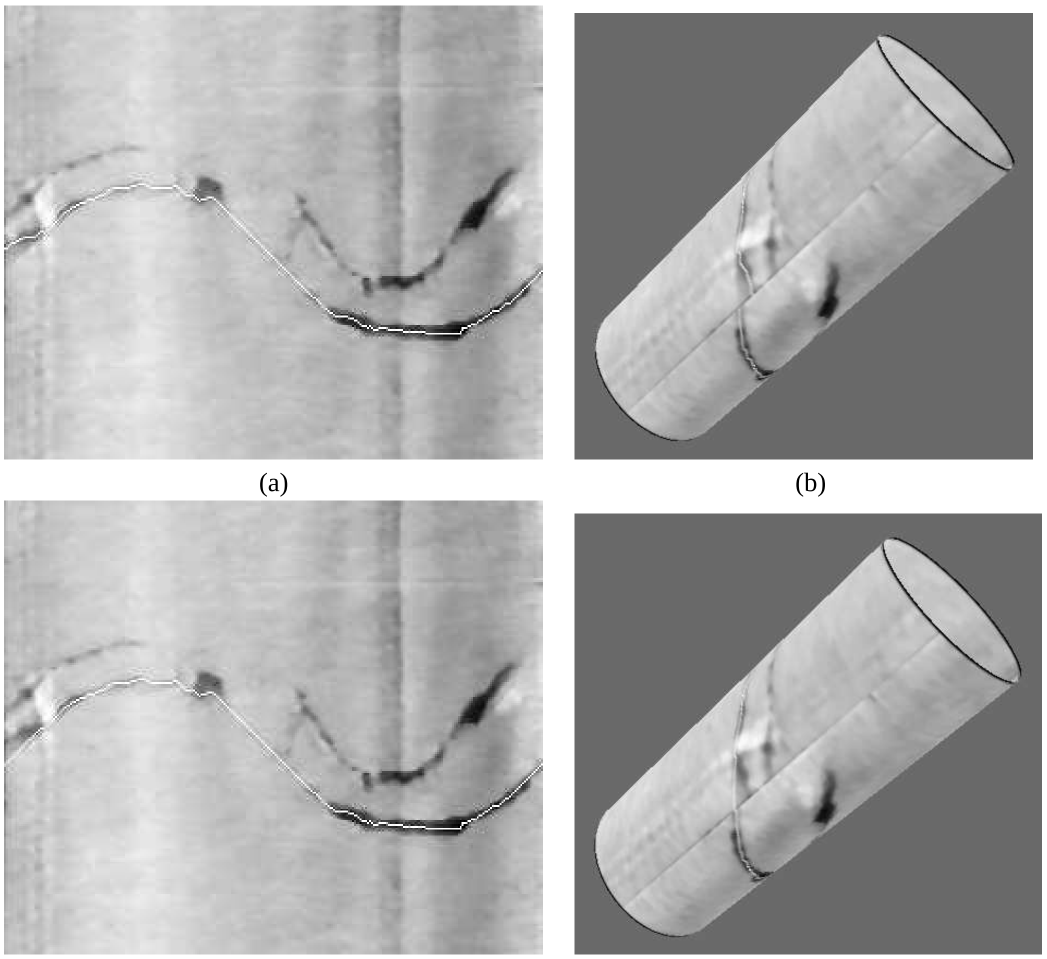


Fig. 7. (a) A borehole image with a crack; The white path is obtained using an ordinary shortest path extraction algorithm without circular constraint. (b) Image in (a) shown in a cylindrical format. The starting and ending positions do not meet. (c) The white path is obtained using the algorithm developed in this paper which has the constraint that the path is circular. (d) Image in (c) shown in a cylindrical format. The starting and the ending points meet each other.

polar coordinate system. Our circular shortest path algorithm is applied to this transformed image and a circular shortest path is extracted. The starting and ending positions of this obtained contour are neighbouring points. If we transform this obtained path from polar coordinate to the original Cartesian coordinate, a closed contour can be guaranteed. Figure 8 shows two examples of finding the boundaries of an object. Figure 8(a,d) are the input images. Figure 8(b,e) are the circular shortest path obtained in the polar coordinates. Figure 8(c,f) show the closed object contours.

C. Panoramic Stereo Matching

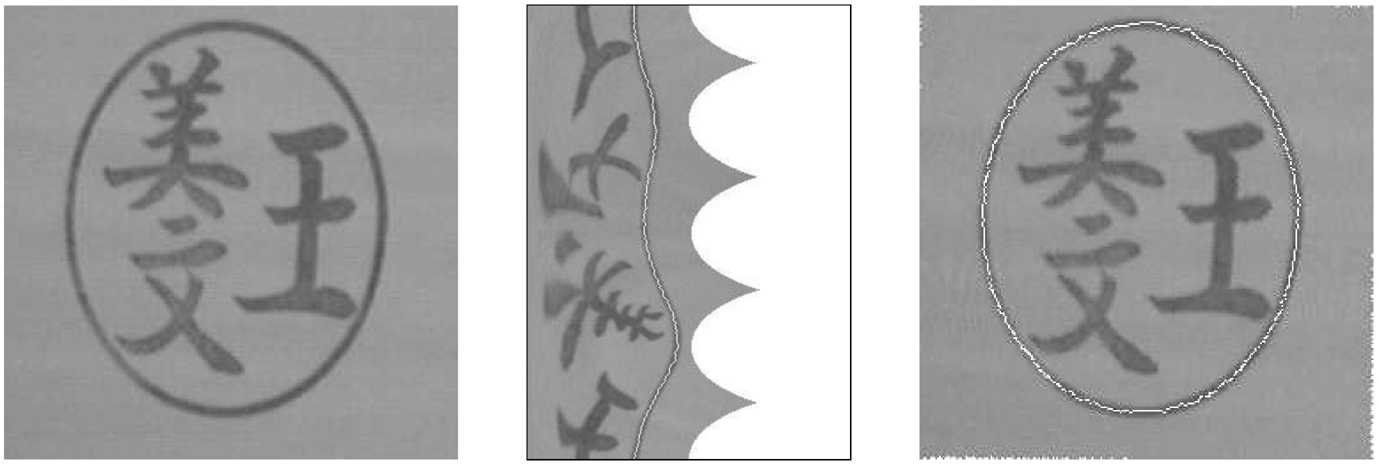
Correlation based methods are very common for stereo matching. Usually a correlation matrix is obtained for each horizontal pair of scanlines from the left and the right stereo images, and a shortest path is obtained in this correlation matrix for disparity estimation. When performing panoramic stereo image matching, it is necessary to take the constraint that the left and the right columns of the stereo images are actually neighbours into account. We can use one of our circular shortest path extraction algorithms to obtain a CSP in a correlation matrix from panoramic stereo images. This obtained CSP will ensure that the starting and the ending position of the path are connected. The first two images in Figure 9 are the left and the right panoramic stereo images; and the third image in Figure 9 is the disparity map obtained using the IP&MBTA algorithm for circular shortest path extraction.

D. Running Times

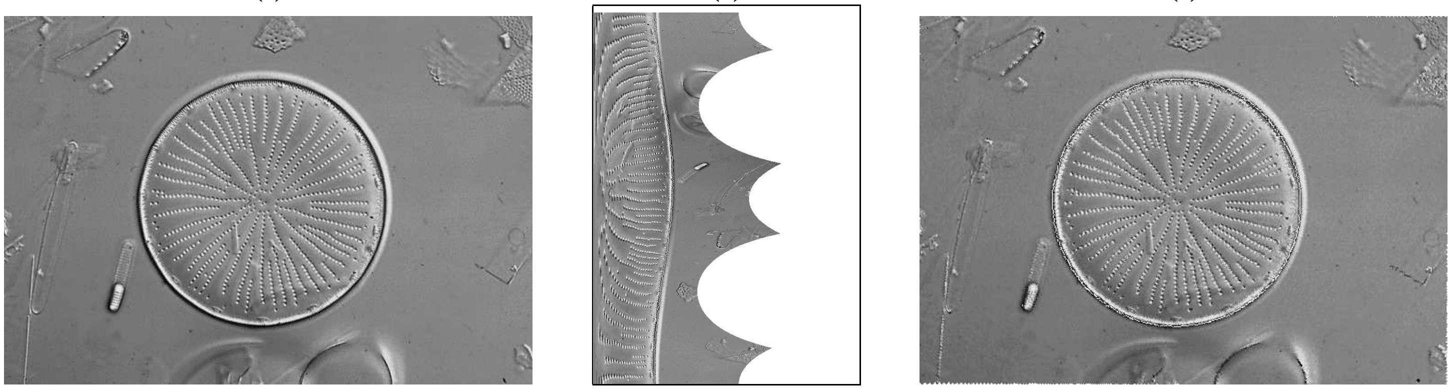
Table I shows the computation time of different algorithms for obtaining circular shortest path on different images. The computer used was a rather slow 85MHz Sun SPARC. All the algorithms except the MSA are very fast and takes in the order of 0.3 seconds. The timing was obtained by running the algorithms on random images several hundreds of times and taking the average.

V. Conclusions

We have developed several new algorithms for finding a circular shortest path in an image. These algorithms have applications in borehole image analysis, object boundary detection, and panoramic stereo matching. The circular shortest path obtained in the image ensures that the starting and ending positions are connected. The five algorithms we developed are



(a) (b) (c)



(d) (e) (f)

Fig. 8. Results of boundary extraction using circular shortest path extraction. (a) An image with a circular contour; (b) Transformed image from Cartesian coordinate to polar coordinate. The white line shows the circular shortest path extracted using the patching method. (Image was rotated by 90 degrees). (c) The recovered image from the polar image. The white line is a closed contour. (d) Image of an Actinocyclus diatom (unicellular algae). (e) Polar transformed image with circular shortest path overlaid. (f) Image shown the closed contour detected.

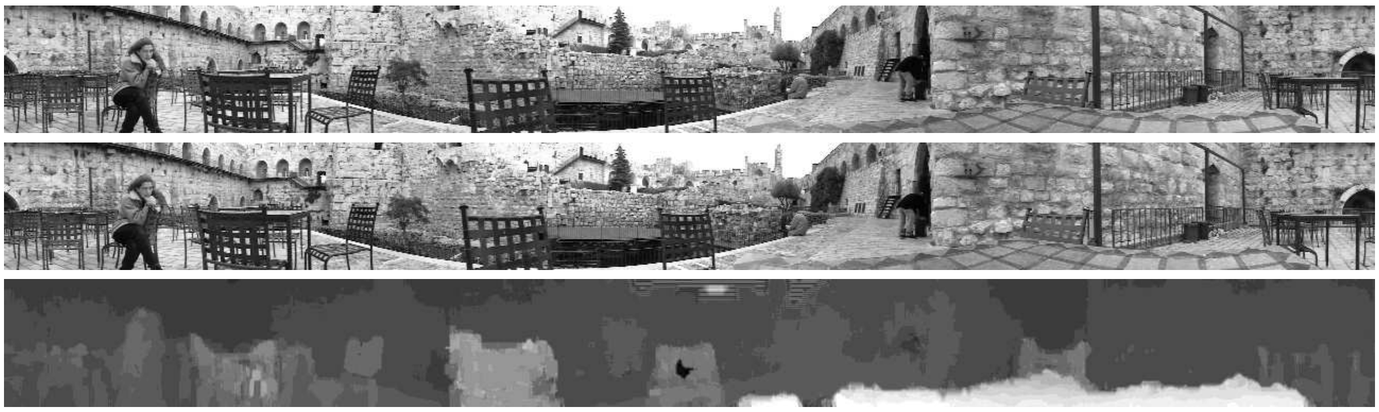


Fig. 9. The first and second images are the left and right input images. The third image gives the matching results using the method described in this paper. (Images courtesy Professor S. Peleg of The Hebrew University of Jerusalem).

TABLE I

RUNNING TIMES OF DIFFERENT ALGORITHMS FOR CIRCULAR SHORTEST PATH EXTRACTION. DYNAMIC PROGRAMMING TECHNIQUES ARE USED AS THE ORDINARY SHORTEST PATH EXTRACTION METHOD ON A 85MHZ SUN SPARC MACHINE.

Image Size	Running Times (s)				
	MSA	IPA	MBTA	IP&MBTA	Approx
256×256	12.284s	0.267s	0.227s	0.312s	0.239s
512×512	389.440s	1.948s	0.989s	2.143s	0.962s

the multiple search algorithm, the image patching algorithm, the multiple back-tracking algorithm, the image patching and multiple back-tracking combined algorithm, and the approximate algorithm. The image patching algorithm is very fast although a solution is not guaranteed. The MBTA is also very fast and it is guaranteed to find a circular path, but may not be the optimal one. The combination of image patching and the multiple back-tracking algorithms achieves a much higher probability and speed in finding the optimum circular shortest path. A typical running time for the image patching algorithm on a 256×256 image is 0.267 seconds on a rather slow 85MHz Sun SPARC computer. The MBTA algorithm takes about 0.227 seconds. The combination of image patching and multiple back-tracking algorithm takes about 0.312 seconds. The algorithm was shown to be fast and reliable in tests on several different types of real images.

Acknowledgement

Figure 8(a) is from the image library of SCIL-Image software distribution. Figure 8(d) is from the ADIAC public data web page:
`http://www.ualg.pt/adiac/pubdat/pubdat.html`.
The first author thank Moshe Sniedovich, Ernesto Martins[1], and Irina Dumitrescu for initial discussions. We are also grateful to Mohan Krishnamoorthy, Michael Buckley and Houyuan Jiang for their suggestions and comments. We thank the anonymous reviewers for their comments.

Biographical Sketch

CHANGMING SUN received his PhD in the area of Computer Vision at Imperial College of Science, Technology and Medicine, London in 1992. Then he joined CSIRO Mathematical and Information Sciences, Australia in December 1992 as Research Scientist, both doing research and working on applied projects. His research interests include computer vision, image analysis, and digital photogrammetry. Dr Sun is a member of the Australian Computer Society and the Australian Pattern Recognition Society.

STEFANO PALLOTTINO is a professor of Operations Research in the Computer Science Department of the University of Pisa, Italy. His research interests include network optimization models and algorithms, path reoptimization algorithms, local search algorithms for hard combinatorial problems, as well as transportation and transit equilibrium models.

References

[1] G. Gallo, S. Pallottino, Shortest path algorithms, Annals of Operations Research 13 (1988) 3–79.
[2] B. V. Cherkassky, A. V. Goldberg, T. Radzik, Shortest paths algorithms: Theory and experimental evaluation, Mathematical Programming 73 (1996) 129–174.
[3] M. Buckley, J. Yang, Regularised shortest-path extraction, Pattern Recognition Letters 18 (7) (1997) 621–629.
[4] C. Sun, Fast stereo matching using rectangular subregioning and 3D maximum-surface techniques, International Journal of Computer Vision 47 (1/2/3) (2002) 99–117.

[1]Unfortunately, Ernesto Queiros Vieira Martins of Universidade de Coimbra (Portugal) suddenly died in November 2000, few months after his useful discussion with the authors; we dedicate this research to his memory.

[5] S. Intille, A. Bobick, Disparity-space images and large occlusion stereo, in: Proceedings of European Conference on Computer Vision, Stockholm, Sweden, 1994, pp. 179–186.
[6] G. L. Gimel'farb, V. M. Krot, M. V. Grigorenko, Experiments with symmetrized intensity-based dynamic programming algorithms for reconstructing digital terrain model, International Journal of Imaging Systems and Technology 4 (1992) 7–21.
[7] S. A. Lloyd, A dynamic programming algorithm for binocular stereo vision, GEC Journal of Research 3 (1) (1985) 18–24.
[8] http://wwwdnyalb.er.usgs.gov/projects/bgag/intro.text.html.
[9] M. A. Lovell, G. Williamson, P. K. H. (eds), Borehole Imaging: applications and case histories, Special Publications no. 159, Geological Society, London, 1999.
[10] H. du Buf, M. Bayer, S. Droop, R. Head, S. Juggins, S. Fisher, H. Bunke, M. Wilkinson, J. Roerdink, J. Pech-Pacheco, G. Cristóbal, H. Shahbazkia, A. Ciobanu, Diatom identifi cation: a double challenge called ADIAC, in: Proc. 10th Int. Conf. on Image Analysis and Processing, Venice, Italy, 1999, pp. 734–739.
[11] R. V. Helgason, J. L. Kennington, B. D. Stewart, Dijkstra's two-tree shortest path algorithm, Tech. rep., Department of Operations Research and Engineering Management, Southern Methodist University, Dallas, TX (1988).