ELSEVIER

Pattern Recognition Letters 18 (1997) 621–629

Pattern Recognition Letters

Regularised shortest-path extraction

Michael Buckley *, Jean Yang

CSIRO Mathematical and Information Sciences, Macquarie University Campus, Locked Bag 17, North Ryde NSW 2113, Australia

Received 6 November 1996; revised 17 June 1997

Abstract

Regularization of shortest-paths and active contours has been considered and attempted by a number of workers. However, it was not until the development of the "time-delayed dynamic programming" algorithm of Amini et al. (1990) in the active contours context that a method was found which was able to apply a simple and intuitive smoothness constraint with an efficient computational scheme. We show that, when applied to the shortest-path problem, this technique gives rise to a simple and efficient algorithm. However, we find that the method is not practically useful in some situations because of discretization effects. A modification using pixel subdivision is proposed which to a large extent overcomes this problem. The modified method is illustrated using two examples: fracture detection in borehole images and road detection in satellite images. © 1997 Elsevier Science B.V.

Keywords: Dynamic programming; Regularization; Road detection; Shortest-path extraction

1. Introduction

Dynamic programming methods for finding shortest-paths between two sets of pixels have been very successful in finding long, nearly linear features in images. One major area of application has been finding roads in satellite images. Dynamic programming algorithms are simple to program and very efficient computationally, and are guaranteed to find a globally optimal path. Amini et al. (1990) provide a number of references to applications of dynamic programming in vision. The general principles of dynamic programming are introduced in (Bellman, 1957).

However, simple shortest-path extraction via dynamic programming imposes no smoothness constraint on the path. It is therefore often the case that the required smooth path is not found, as there exists a shorter non-smooth path joining the two sets. Several authors have attempted to regularise the path extraction process by incorporating a penalty for roughness. Fischler et al. (1981) added a constant to each pixel value, effectively penalising the number of pixels in a path. In a slightly different context – namely active contours – Amini et al. (1990) developed a more sophisticated method which applies a penalty proportional to the "energy" of the contour and finds the optimal contour using "time-delayed" dynamic programming. A similar idea is used in a different context in (Gruen and Li, 1995) and (Merlet and Zerubia, 1996). Martelli (1976) uses a "smoothness" measure in regularising edge and contour detection.

In this paper we show how the "time-delayed" dynamic programming algorithm for contour extraction can be adapted to produce a very simple and effective algorithm for finding smooth shortest-paths across an

* Corresponding author.

PII S0167-8655(97)00076-7

image. We also show that a simple pre-processing step involving pixel subdivision is required, and that when this extra step is included the result is a very effective algorithm for extraction of smooth paths.

2. Ordinary shortest-path extraction

First we describe the well-known dynamic programming algorithm for ordinary shortest-path extraction; cf. (Amini et al., 1990). We consider the case where a top-to-bottom shortest path is required through a rectangular image. This procedure is symmetric in the sense that a bottom-to-top version yields the same shortest-path, except where the shortest-path is not unique. In this case there is some arbitrariness in the selection of one of the equally-shortest paths and this is usually not symmetric. A left-to-right shortest-path algorithm is obtainable by exchanging the roles of the image subscripts i and j.

For $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$ let $X(i,j)$ be the grey value of the (i,j)th pixel in the $m \times n$ input image X. A pth-order path P from any point (i,j) in X to the top of the image is a set of i pixels

$$P = \{(1,j_1),\ldots,(i,j_i)\},$$

with $1 \leqslant j_r \leqslant n$ for $r = 1,\ldots i$;

$$|j_{r+1} - j_r| \leqslant p \tag{1}$$

for $r = 1,\ldots i-1$; and $j_i = j$. That is, P is a set of pixels, one in each of the first i rows of X, which ends at the point (i,j) and has the pth-order connectivity property (1). Typically $p = 1$, in which case the path is an 8-connected set, although larger values of p are used below after pixel subdivision.

Such a path is monotonic – it always proceeds downwards. Iteration of the ordinary shortest-path algorithm allows paths which fold back on themselves, but we do not consider these here.

In ordinary shortest-path extraction the length of a path P is defined simply as the sum of the grey values of the pixels in the path:

$$\sum_{r=1}^{i} X(r,j_r). \tag{2}$$

We seek to find the shortest path joining the top and bottom rows of X; that is, the path $P = \{(1,j_1),\ldots,(m,j_m)\}$ which minimises

$$\sum_{r=1}^{m} X(r,j_r).$$

To do this we first define the distance $Y(i,j)$ from each (i,j) to the top of the image as the minimum over all paths P ending at (i,j) of the path length (2). For the trivial case $i = 1$ there is only one path ending at $(i,j) = (1,j)$, namely the single-pixel path $P = \{(1,j)\}$, so

$$Y(1,j) = X(1,j). \tag{3}$$

This holds for all $j = 1,\ldots,n$.

The key recursion upon which the algorithm is based is obtained using the continuity property (1). Because of this property any path from the top of the image to a point (i,j) must include one of its $2p+1$ "neighbours" above – i.e. $(i-1,j+k)$ for $-p \leqslant k \leqslant p$. This is true for all $i \geqslant 2$. Therefore, the minimum distance to the top from (i,j) is the minimum over k between $-p$ and p inclusive of the distance to the top via $(i-1,j+k)$. This gives the basic recursion

$$Y(i,j) = X(i,j) + \min_{k:|k|\leqslant p} Y(i-1,j+k), \tag{4}$$

which, together with the initial condition (3), enables quick recursive calculation of $Y(i,j)$ for all i and j.

The value of k which achieves the minimum in (4) is needed in the later back-tracking phase of the algorithm, so it is efficient to store this in an array $k(i,j)$:

$$k(i,j) = \operatorname*{argmin}_{k:|k|\leqslant p} Y(i-1,j+k).$$

The length of the top-to-bottom shortest path can now be calculated as

$$\min_{1\leqslant j\leqslant n} Y(m,j). \tag{5}$$

Furthermore, the last point of the shortest-path is the point $(m,\hat{j}_m)$ which achieves the minimum in (5).

The second-last point, $(m-1,\hat{j}_{m-1})$, satisfies $|\hat{j}_m - \hat{j}_{m-1}| \leqslant p$. It is easily seen that, in fact, $\hat{j}_{m-1} = \hat{j}_m + k(m,\hat{j}_m)$. Similarly, $\hat{j}_{m-2} = \hat{j}_{m-1} + k(m-1,\hat{j}_{m-1})$, and so on. This process can be repeated, "back-tracking", as it were, through the image, to find $\hat{j}_{m-3},\hat{j}_{m-4},\ldots,\hat{j}_2,\hat{j}_1$.

Thus we have the following simple and fast algorithm for ordinary shortest-path extraction:

```
/* Forward pass... */
for j = 1 ... n
  Y(1, j) = X(1, j)
for i = 2 ... m
  for j = 1 ... n
    {
    Y(i, j) = X(i, j) + min_{k:|k|≤p} Y(i − 1, j + k)
    k(i, j) = ⟨the value of k which achieves
      this minimum ⟩
    }
/* Back-tracking... */
ĵ_m = argmin_{1≤j≤n} Y(m, j)
for i = m − 1 ... 1
  ĵ_i = ĵ_{i+1} + k(i + 1, ĵ_{i+1})
```

Note that $Y(i,j)$ is not defined for $j < 1$ or $j > n$. This simplified code implicitly assumes that these values are some constant value which is greater than all other values of $Y(i,j)$.

Fig. 1 contains two sample images, an image of the inside of a borehole (left) and a filtered version of an aerial photograph of the La Rochelle area in France. The aim is to extract the fractures in the borehole image and in the La Rochelle image to extract the main road which crosses the top of the image near the left side and the bottom of the image a little to the right of the centre. The filter which has been applied to the original La Rochelle image to produce the right-hand image in Fig. 1 may not be optimal. However, this is not so important as the purpose here is to assess the effectiveness of path-finding algorithms for a given image on which they operate.

Fig. 2 shows the result of this algorithm applied to the images in Fig. 1. In the borehole image (left) the path follows the stronger of the two fractures. In the La Rochelle image (right) the algorithm fails to find all of the main road, bending sharply to the right in the town near the top of the picture whereas one would like it to continue along the main road towards the top-left corner of the picture. Being unconstrained with respect to smoothness the path also misses some other sections of the road in the lower half of the image. Note that in both examples the path jumps across areas of high grey value towards areas of low grey value. This is a powerful feature of shortest-path methods.

The storage requirement of this algorithm is $O(2mn)$ for the arrays Y and k. The computational expense is $O(Qmn)$ for the forward pass where $Q = 2p + 1$ is the number of possible steps at each point, and $O(m)$ for the backward pass. The overall computational expense is therefore $O(Qmn)$.

3. Regularised shortest-path extraction

In this section we describe an algorithm for regularised shortest-path extraction and point out some of its practical shortcomings. The basic algorithm seems to have been first discovered by Amini et al. (1990) in the context of active contours. These authors use the phrase "time-delayed dynamic programming" to describe the method. Another version of the algorithm is used in (Merlet and Zerubia, 1996) for detection of roads.

In regularised shortest-path extraction we apply a penalty to each path which is proportional to a simple measure of its roughness. As in ordinary shortest-path extraction the minimum-length top-to-bottom path is found, but now the length of each path includes a roughness term. The roughness of a path $P = \{(1, j_1), \ldots, (i, j_i)\}$ is defined as (Amini et al., 1990; Gruen and Li, 1995; Merlet and Zerubia, 1996)

$$\sum_{r=2}^{i-1} (j_{r-1} - 2j_r + j_{r+1})^2. \tag{6}$$

The length of path P is therefore defined as

$$\sum_{r=1}^{i} X(r, j_r) + \lambda \sum_{r=2}^{i-1} (j_{r-1} - 2j_r + j_{r+1})^2,$$

where λ is the regularisation constant and the computational task is to find the top-to-bottom path P for which this is a minimum.

The regularisation constant, λ, governs a trade-off between the two measures which we would like to minimise: path brightness, as measured by $\sum_{r=1}^{i} X(r, j_r)$, and path roughness as measured by (6). If λ is very large, the optimum path will be very smooth but may not be very dark. On the other hand if λ is very small, the roughness penalty is negligible and the optimum path will be as dark as possible but may not be smooth. An appropriate value of λ must be selected by the user.

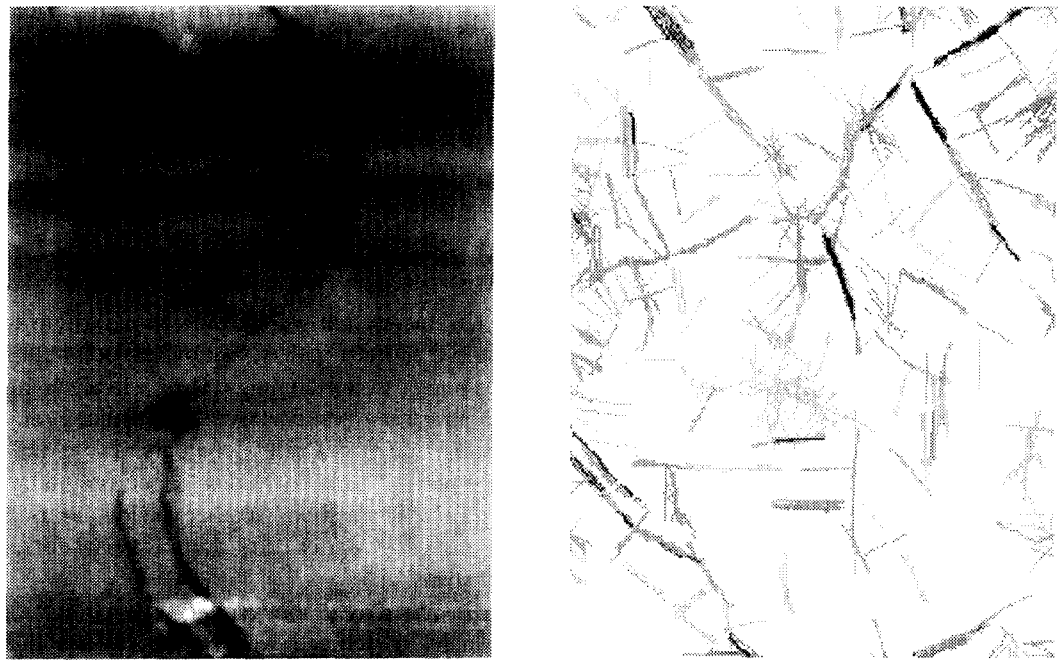

Fig. 1. Borehole image and La Rochelle image (©CNES (92) SPOT IMAGE).

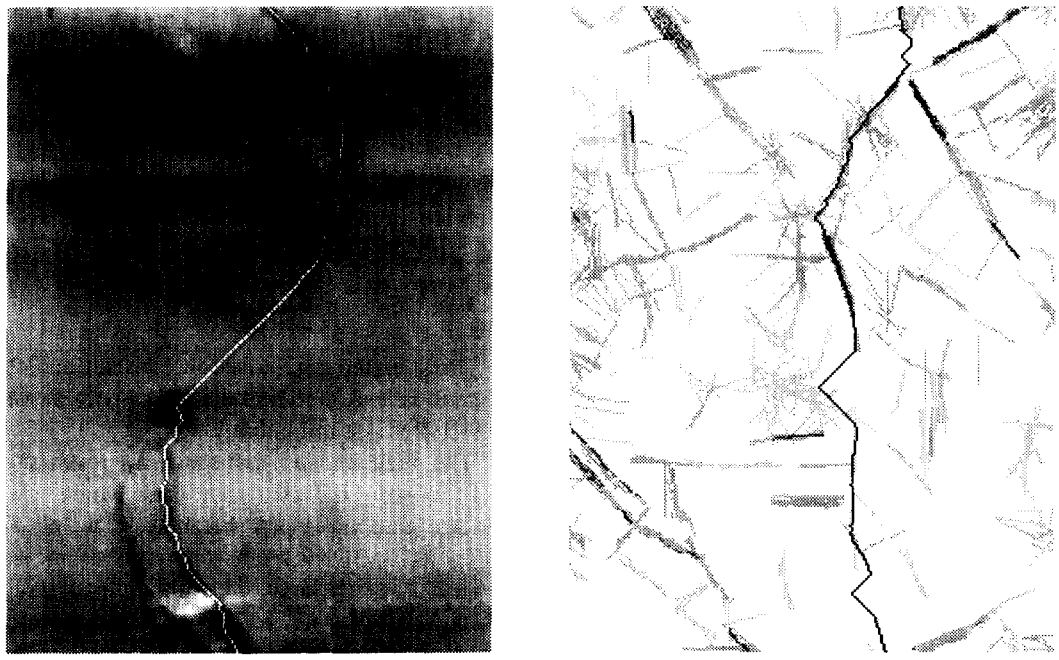

Fig. 2. Ordinary shortest paths, $p = 1$. Paths are overlayed in white in borehole image (left) and in black in filtered La Rochelle image (right).

In ordinary shortest-path extraction we define $Y(i,j)$ as the distance (length of shortest path) from each point (i,j) to the top of the image. Here we need to define $Y(i,j,k)$ as the distance (length of shortest path) from each point (i,j) to the top of the image via $(i-1, j+k)$, for $k = -p, \ldots, p$, so Y represents a three-way array with dimensions $m \times n \times P$.

The initial condition corresponding to (3) is

$$Y(2,j,k) = X(2,j) + X(1,j+k),$$

which holds for $j = 1, \ldots, n$ and $k = -p, \ldots, p$. If $j + k < 1$ or $j + k > n$, $X(1, j+k)$ is taken to be infinite.

To derive a recursion for $Y(i,j,k)$ we note that any path from (i,j) via $(i-1, j+k)$ must pass through $(i-2, j+k+l)$ for some $|l| \leqslant p$. The contribution of this triple of points to the roughness of the path as defined by (6) is $\lambda(l-k)^2$. This leads to the equation

$$Y(i,j,k) = X(i,j) + \min_{l:|l|\leqslant p} \{Y(i-1, j+k, l) + \lambda(l-k)^2\} \tag{7}$$

through which all values of $Y(i,j,k)$ may be calculated recursively. The value of l achieving the minimum in (7) is stored for use in back-tracking:

$$l(i,j,k) = \operatorname*{argmin}_{l:|l|\leqslant p} \{Y(i-1, j+k, l) + \lambda(l-k)^2\}.$$

The back-tracking phase of the algorithm begins by finding

$$(\hat{j}_m, \hat{k}_m) = \operatorname*{argmin}_{(j,k)} Y(m,j,k).$$

The last point on the optimal path is now known: $(m, \hat{j}_m)$. The second-last point is known also; it is $(m-1, \hat{j}_{m-1})$ with $\hat{j}_{m-1} = \hat{j}_m + \hat{k}_m$. The position $\hat{j}_{m-2}$ of the third-last point can now be calculated as $\hat{j}_{m-2} = \hat{j}_{m-1} + \hat{k}_{m-1}$ with

$$\hat{k}_{m-1} = l(m, \hat{j}_m, \hat{k}_m).$$

This process can now be repeated to find all points $(r, \hat{j}_r)$ on the regularised shortest path.

Thus we have the following algorithm for regularised shortest path extraction.

```
/* Forward pass... */
for j = 1...n
  for k = -p...p
    Y(2,j,k) = X(2,j) + X(1,j+k)
for i = 3...m
  for j = 1...n
    for k = -p...p
      {
      Y(i,j,k) = X(i,j)
        + min_{l:|l|≤p}{Y(i-1,j+k,l) + λ(l-k)^2}
      l(i,j,k) = ⟨the value of l which achieves
        this minimum⟩
      }
/* Back-tracking... */
(ĵ_m, k̂_m) = argmin_(j,k) Y(m,j,k)
ĵ_{m-1} = ĵ_m + k̂_m
for i = m-1...2
  {
    k̂_i = l(i+1, ĵ_{i+1}, k̂_{i+1})
    ĵ_{i-1} = ĵ_i + k̂_i
  }
```

Again this is simplified code which assumes that $X(i,j)$ and $Y(i,j,k)$ are both infinite for $j < 1$ or $j > n$.

Fig. 3 shows the results of the regularised shortest path algorithm applied to the images in Fig. 1 with large values of λ. The apparent roughness of the selected paths is small, as expected. The optimal path through the borehole image, for example, has 9 bends. At each of these points the second difference of position is 1 or -1. At all other points this second difference is zero. Hence the roughness of the path is 9λ.

Although the criterion (6) has been successfully minimised so that the paths in Fig. 3 are "smooth" according to a mathematical definition of roughness and smoothness, these paths are not really "smooth" in the usual sense of the word. They contain infinitely smooth (i.e. straight) sections separated by sharp bends. A gradual change of orientation would look smoother, but a more significant problem than visual appearance is the inability of such an algorithm to extract paths at arbitrary slopes. For example, the method fails to find all of the road in the La Rochelle image. This effect is caused by discretization in the range of possible orientations of very smooth (straight) paths or path fragments. A simpler artificial example of this problem is shown in Fig. 4(a). In this image the regularised shortest path algorithm would prefer the path beginning at the top-right of

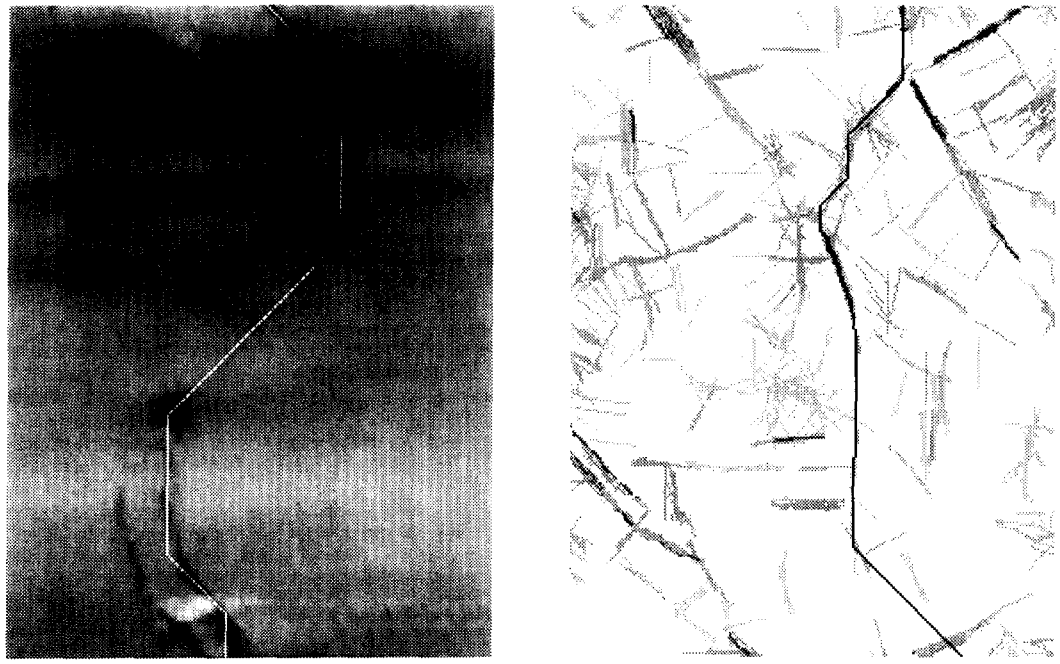

Fig. 3. Regularised shortest paths, $p = 1$. Paths are overlayed in white in borehole image (left) and in black in filtered La Rochelle image (right). Regularisation parameters: $\lambda = 500$ for borehole image; $\lambda = 200$ for La Rochelle image.

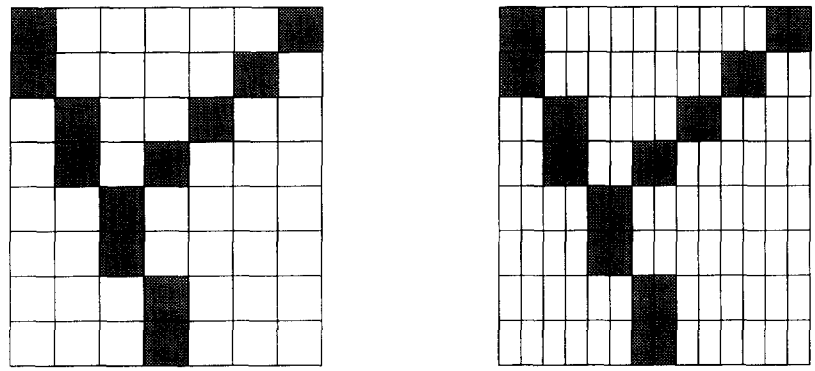

Fig. 4. (a) Original image. (b) Image subdivided by factor $q = 2$.

the image as its roughness is 3λ while the path from the top-left of the image which ideally should be preferred has roughness 6λ. The next section addresses, and to a large extent solves, this problem.

It is important to note that the scale of the regularisation constant λ depends on the scale of the grey values of the image. If the image values are scaled by a factor k then λ should also be scaled by k to obtain the same shortest path.

The storage requirement of the regularised shortest-path algorithm is $O(2mn)$ for the arrays Y and l, where $Q = 2p + 1$. The computational expense is $O(Q^2mn)$

for the forward pass and $O(m)$ for the backward pass, so the overall computational expense is $O(Q^2mn)$.

The factor Q is invariant to changes in resolution. For example, if the minimum allowable path angle is 45°, then $p = 1$ and $Q = 3$, regardless of resolution. Therefore if the resolution is multiplied by R, then m and n increase by a factor R, but Q is unchanged. Both storage requirement and computational cost are therefore proportional to the square of resolution.

4. Pixel subdivision

As pointed out above the regularised shortest paths shown in Fig. 3 are not as "smooth" as we would hope. The essential problem here is exemplified in Fig. 5(a). The path shown in this figure is as "smooth" as it can be given its orientation. One would hope that the measure of the roughness of such a path would be zero. However, by the measure (6) the roughness of this path is 4λ as each of the four internal triples of pixels contains a bend. Furthermore it is clear that any roughness measure based on successive triples of pixels will give a non-zero roughness for such a path. As there seems no way to implement roughness measures other than such local ones, this problem is fundamental.

The issue is one of discretization. Clearly there exist vertical and diagonal paths (or path fragments) with zero roughness. More generally, if $1 \leqslant a \leqslant p$ paths such as

$$P = \{(1, a+b), (2, 2a+b), (3, 3a+b), \ldots\}$$

have zero roughness and slope $1/a$, so for any given p the possible slopes of zero-roughness paths are

$$\infty, \pm 1, \pm 1/2, \pm 1/3, \ldots, \pm 1/p.$$

However there are no other zero-roughness paths or path fragments. In particular there are, apart from vertical paths, no zero-roughness paths with slope m greater than 1 or less than -1. Fig. 5(a) is an example of this with $m = -2$.

Figs. 5(b)–(e) show how a simple pixel subdivision procedure can overcome this problem by permitting zero-roughness paths at a greater range of angles. In Fig. 5(b) each pixel is divided into two – that is replicated once. Fig. 5(c) shows a path through the new subdivided pixels which has zero roughness and the required slope, $m = -2$. This can occur because the roughness of the path is measured in a new space where its slope is -1. In Fig. 5(d), the path in Fig. 5(c) is replicated once and Fig. 5(e) is obtained from Fig. 5(d) by sampling every second column. Notice that the path in Fig. 5(e) is identical to that in Fig. 5(a) and yet its roughness is zero because it is derived from the path in Fig. 5(c) and the roughness of this path is measured in the space of subdivided pixels.

The modified procedure for regularised path extraction with subdivision factor q is carried out as follows:

- Divide each pixel horizontally into q sub-pixels; that is, replicate horizontally $q - 1$ times.
- Find the regularised shortest path using maximum step p' defined below.
- Replicate the path q times.
- Subsample every qth column.

If in a given context a minimum slope of M is assumed, then $p \approx 1/M$ would be used in the original (non-subdivided) space. Now if a subdivision by a factor q is used to allow more angles, a modified value of maximum step p given by $p' = qp$, must be used in the subdivided space so that sufficiently small slopes can be obtained. For example, if $p = 2$ in the original space and a subdivision factor $q = 2$ is used then $p' = 4$ must be used in the subdivided space to permit slopes as small as $1/2$.

In simple cases the smallest possible slope is taken to be 1, so $p = 1$. If a subdivision factor q is used in such a case the maximum step value $p' = q$ would be used in the space of subdivided pixels. This is the case for the examples used in this paper.

Fig. 4 illustrates the effect of this procedure. As discussed in the previous section, the regularised shortest path algorithm would prefer the path beginning at the top-right of the image while we would prefer the path from the top-left of the image to be found. However, in the space of subdivided pixels shown in Fig. 4(b), the path from the top-left is preferred as its roughness is zero while the best path through the grey pixels from the top-right has roughness 3λ.

Fig. 6 shows regularised shortest paths through the borehole and La Rochelle images with $p = 1$ and subdivision factor $q = 5$. Compared to Fig. 3 these are paths are visually smooth. Some sharp bends can be seen but these are less pronounced than in Fig. 3. More importantly though, the regularised shortest path

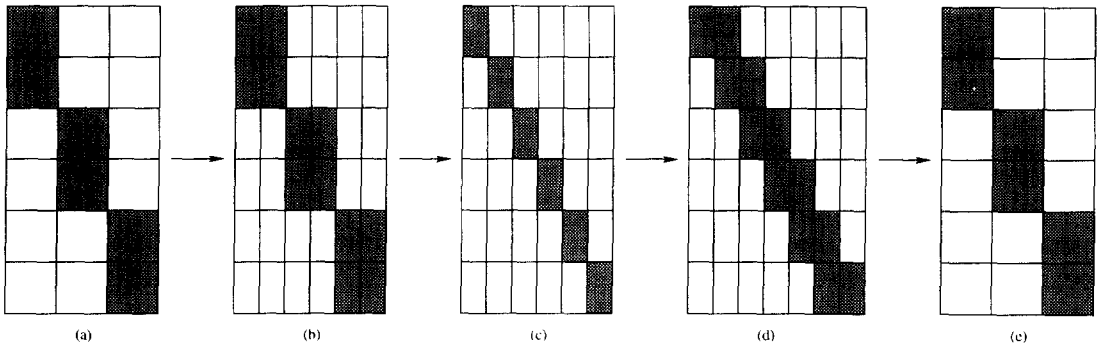

Fig. 5. (a) Original image. (b) Image subdivided by factor $q = 2$. (c) Regularised shortest path. (d) Replicated shortest path. (e) Subsample: every second column.

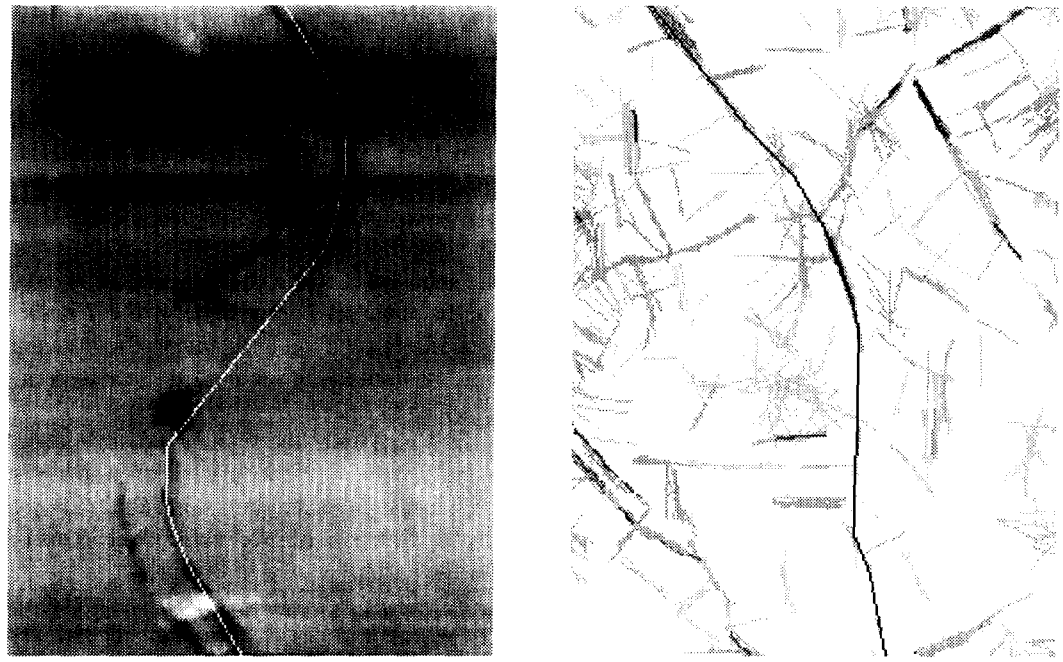

Fig. 6. Regularised shortest paths with $p = 1$ and subdivision factor $q = 5$. Paths are overlayed in white in borehole image (left) and in black in filtered La Rochelle image (right). Regularisation parameters: $\lambda = 500$ for borehole image; $\lambda = 200$ for La Rochelle image.

through the La Rochelle image in Fig. 6 is successful in finding the section of road on the top-left quarter of the image. This is now possible because of the wider range of possible slopes of zero-roughness path segments. In Fig. 2 the top section of the road was not found because no smoothness was imposed. Smoothness was imposed in Fig. 3 but this part of the road could still not be found for any choice of λ, because its slope (approx. 1.25) is such that a path running through the road in this area is *locally* rough despite being *globally* smooth. This problem is overcome in Fig. 6 using pixel subdivision.

Pixel subdivision by a factor q increases the computational cost, Q^2mn, by a factor of approximately q^3, as n is multiplied by q and Q increases from $Q = 2p + 1$ to $Q' = 2p' + 1 = 2pq + 1 \approx qQ$. For ex-

ample, if $p = 1$ and $q = 2$ then n is doubled and Q increases from 3 to 5, so Q^2mn increases by a factor $50/9 = 5.56$.

A reduction in computational cost may be achieved if resolution can be reduced by means of an alternative approach in which, instead of dividing pixels q times horizontally, we merge pixels q at a time vertically. The same aspect ratio of pixels is achieved this way and the number of possible path angles is increased in the same way by setting $p' = qp$. Whereas in pixel subdivision m is unchanged and n increases by a factor q, in pixel merging, n is unchanged and m is *reduced* by a factor q. Computational cost is therefore increased by a factor approximately q only. In the example above with $p = 1$ and $q = 2$ the computational cost increased by a factor 5.56 with pixel subdivision, but with pixel merging the corresponding factor is $25/18 = 1.39$.

5. Conclusion

We have shown that time-delayed dynamic programming, when applied to the shortest-path problem, results in a very simple algorithm for extraction of smooth paths. We have also shown that the algorithm encounters problems because of discretization, but that these can be overcome by subdividing pixels. Examples demonstrate the effectiveness of this procedure.

Acknowledgements

The authors acknowledge Hal Gurgenci, Centre for Mining Technology and Equipment for granting permission to use the borehole image and CNES for permission to use the La Rochelle image.

References

Amini, A.A., Weymouth, T.E., Jain, R.C., 1990. Using dynamic programming for solving variational problems in vision. IEEE Trans. Pattern Anal. Machine Intell. 12 (9), 855–867.

Bellman, R., 1957. Dynamic Programming. Princeton University Press, Princeton, NJ.

Fischler, M.A., Tenenbaum, J.M., Wolf, H.C., 1981. Detection of roads and linear structures in low resolution aerial imagery using a multisource knowledge integration technique. Comput. Graphics Image Process. 15, 201–223.

Gruen, A., Li, H., 1995. Road extraction from aerial and satellite images by dynamic programming. IRPRS J. Photogrammetry and Remote Sensing 50 (4), 11–20.

Martelli, A., 1976. An application of heuristic search methods to edge and contour detection. Comm. ACM 19 (2), 73–83.

Merlet, N., Zerubia, J., 1996. New prospects for line detection by dynamic programming. IEEE Trans. Pattern Anal. Machine Intell. 18 (4), 426–431.