Available online at www.sciencedirect.com

ScienceDirect

Publishing Services by Elsevier

International Journal of Naval Architecture and Ocean Engineering 9 (2017) 492–498

http://www.journals.elsevier.com/international-journal-of-naval-architecture-and-ocean-engineering/

Optimisation of pipeline route in the presence of obstacles based on a least cost path algorithm and laplacian smoothing

Ju Young Kang [a,*], Byung Suk Lee [b,1]

[a] School of Mechanical Engineering, Pusan National University, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan, 46241, South Korea

[b] Naval Architecture, Ocean and Marine Engineering, University of Strathclyde, Henry Dyer Building, 100 Montrose Street, Glasgow, G4 0LZ, UK

Received 28 December 2016; revised 2 February 2017; accepted 6 February 2017

Available online 24 February 2017

Abstract

Subsea pipeline route design is a crucial task for the offshore oil and gas industry, and the route selected can significantly affect the success or failure of an offshore project. Thus, it is essential to design pipeline routes to be eco-friendly, economical and safe. Obstacle avoidance is one of the main problems that affect pipeline route selection. In this study, we propose a technique for designing an automatic obstacle avoidance. The Laplacian smoothing algorithm was used to make automatically generated pipeline routes fairer. The algorithms were fast and the method was shown to be effective and easy to use in a simple set of case studies.

Keywords: Dijkstra's algorithm; Geographic information systems; Laplacian smoothing algorithm; Obstacle avoidance; Subsea pipeline

1. Introduction

Pipelines are essential for oil and gas projects, and represent a significant portion of infrastructure investment. It is essential, therefore, that their design should be cost-effective whilst meeting their operational requirements. There are many design parameters, including route selection, pipe sizing, material, coating, wall thickness, free span and cathodic protection (Chakrabarti, 2005). The design is affected by many factors too, such as flow rate, fluid properties, seabed terrain, on-bottom stability and thermal expansion. Several organisations provide design codes including the American Petroleum Institute, American Society of Mechanical Engineers, British Standards, Det Norske Veritas (DNV) and the International Organization for Standardization. Using these design codes, such as the DNV Offshore Standard and Recommended Practice (DNV, 2012, 2010a; 2011a, 2010;b 2006, 2011b; 2010c, 2010d), subsea pipeline designers can determine the pipe diameter and wall thickness, as well as numerically analysing the expansion, fatigue, on-bottom stability and span. Most of these tasks can be computerized, but pipeline route selection still requires much human intervention due to difficulties in automation. This work therefore requires skilled designers with good experience in the art of pipeline route selection. Consequently the process is time-consuming, and a prolonged pipeline route design process can create a bottleneck for an oil and gas project.

Many factors were considered by various authors when selecting the optimum route, including environmental, physical, societal, political, regulatory, technical and economic issues (e.g., Carpenter and Callen (1984), Ryder (1987), Feldman et al. (1995), Montemurro et al. (1998) and Feizlmayr and McKinnon (1999)). Recently, the rapid increases in speed and capability of computers have allowed engineers and researchers to use three-dimensional (3D) Geographic Information Systems (GIS) during the route selection process. Thus, a

* Corresponding author. Fax: +82 51 582 9164.

E-mail addresses: jyoung@pusan.ac.kr (J.Y. Kang), b.s.lee@strath.ac.uk (B.S. Lee).

Peer review under responsibility of Society of Naval Architects of Korea.

[1] Fax: +82 51 582 9164.

http://dx.doi.org/10.1016/j.ijnaoe.2017.02.001

GIS-based system was used in a pipeline expansion project by Montemurro et al. (1998), while Feizlmayr and McKinnon (1999) performed various projects such as pipeline location selection and emergency response plans using a GIS-based pipeline information system. Matori and Lee (2009) prepared a GIS-generated subsea pipeline route based on the commercial program ArcGIS 3.2. These studies have contributed to technical advances in pipeline route design, but there is still no effective method for automatic pipeline route design with efficient obstacle avoidance algorithms.

In this study, we first investigate the use of the Least Cost Path (LCP) algorithm for pipeline route design, and then discuss smoothing algorithms to make the generated route fairer and show why the Laplacian smoothing algorithm is a better choice for this purpose. A computer program was written in C++ based on the algorithms and a series of case studies were conducted to show how the developed techniques can be used for pipeline route design in 3D seabed terrains with various obstacles.

2. LCP algorithm

Selecting an optimum pipeline route is the first major step during marine pipeline design and construction, and the criteria or conditions (terrain, obstacle, politics, etc.) that apply to the pipeline design are never the same in different offshore pipeline projects. The use of GIS to support the complex task of pipeline route selection process has been discussed extensively and some of the pertinent documents are open to the public. However, most are brief articles in magazines rather than detailed academic studies, and thus they do not provide sufficiently detailed technical information to be of practical use. In any case the industry practice still largely relies on manual process which requires much skill and experience.

In automating this crucial task, some form of optimisation scheme must be used and a most promising approach in this application is the LCP algorithm. The LCP algorithm is an algorithm for finding the least cost paths between two nodes. The least cost path would be the shortest path, the least time path, or minimum construction cost path depending on the user. This algorithm has been employed widely in many industries. For example, Berglund et al. (2003) and Yu et al. (2003) employed it for road planning, whereas Fraichard and Ahuactzin (2001) and Connors and Elkaim (2007a, 2007b) applied this algorithm to self-driving autonomous vehicles. Recently, the LCP algorithm has been used widely in video games. One of the classic LCP algorithms is Dijkstra's Algorithm (1959). Cui and Shi (2011) reviewed three frequently used techniques for path finding, where they focused on the A* path finding algorithm introduced by Hart et al. (1968) and is an extended form of Dijkstra's Algorithm.

In this study we investigated the use of the LCP algorithm based on Dijkstra's Algorithm in determining the optimum pipeline route. The technique was demonstrated by using a number of case studies in diverse obstacle situations based on a 3D seabed terrain.

Dijkstra's Algorithm and the A* Algorithm are classic graph search algorithms, which can find the LCP between any two nodes on a graph. The graph is a mathematical structure where the vertices or edges are associated with geometric objects, as shown in Fig. 1.

Before the development of A* Algorithm, Dijkstra's Algorithm was the only choice for path finding. Unlike Dijkstra's Algorithm where the entire graph must be searched, the A* uses a heuristic method to reduce the search area. The heuristics employed by the algorithm estimates the cost of the shortest (cheapest) path from the last node (n) on the path to the goal. By reducing the search range, the algorithm greatly reduces the time required to find LCPs. The acceleration of path finding is undoubtedly attractive for applications in video games and autonomous vehicles, and thus the A* Algorithm has become one of the most widely used algorithms for finding LCPs. However, this algorithm cannot guarantee finding the shortest route because it does not cover all of the possible nodes that the path can visit. Unlike video games or autonomous vehicles, real-time search is not important in pipeline route design, while optimum solutions are the key requirement. Therefore, in our current application Dijkstra's Algorithm was considered to be a better method. In fact, Dijkstra's algorithm can also obtain the solution almost immediately due to the speed and capacity of modern computers.

To find a shortest path (there may be more than one shortest paths), Dijkstra's Algorithm repeatedly examines the unvisited nodes neighbouring the current one and compares the previously assigned tentative values with the values that can be achieved by using the current node. For each node the tentative value (shortest distance to the node from the starting point so far determined) are then updated by assigning the lesser of the two values to the node. The node with the lowest value of the neighbouring set is then made the current node and the process is repeated until the target node is reached. If a cul-de-sac is reached, the nodes are retraced until progress can be made. Fig. 1 shows a simple graph problem where v1 is the starting node while v5 is the target node. The number on each edge denotes the cost (distance) between two nodes of the edge. It can be demonstrated that the LCP is (v1 → v2 → v4 → v3 → v5) and the cost of this route is seven (7).

Most digital display systems project images using fixed picture-element (pixel) arrays, and the pixels are normally

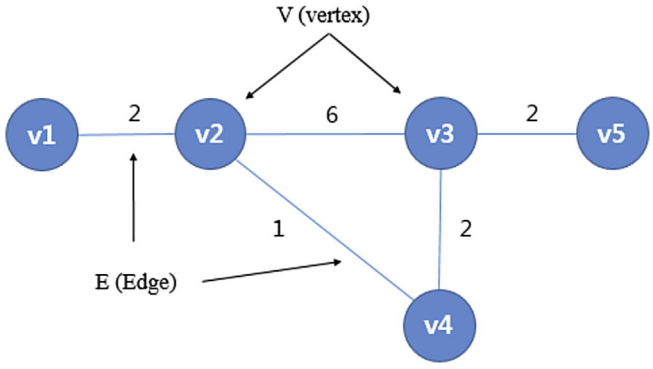

Fig. 1. An example graph.

arranged in a regular 2D grid for convenience. GIS relies on digital displays, and therefore grid-based path finding algorithms will often be more advantageous for offshore pipeline route design than the graph-based algorithm. However, the graph structure can easily be extended to a grid type by considering each pixel as a node. The whole terrain can therefore be discretized as an array of nodes. An example of such a grid is shown in Fig. 2.

Fig. 2 shows an example grid for applying Dijkstra's Algorithm with the starting node *S* and the target node *n21*. The numbers in brackets refer to the least movement cost from the start node *S* to the node in question, and the names of the nodes are marked above the movement costs. The movement cost between two immediate orthogonal neighbouring nodes is assumed to be 10, while the cost of moving from any node to its diagonal neighbour is 14 (an approximation of $10\sqrt{2}$).

There are nine paths from *S* to *n21*, excluding zigzag paths. Among these, five have orthogonal movements only and the total cost from the start to the target is 50. Each of the other four routes contains one diagonal movement and three orthogonal movements, and the total cost is equally 44. The four paths with the same cost are: ($S \rightarrow n3 \rightarrow n13 \rightarrow n17 \rightarrow n21$), ($S \rightarrow n4 \rightarrow n13 \rightarrow n17 \rightarrow n21$), ($S \rightarrow n4 \rightarrow n14 \rightarrow n17 \rightarrow n21$) and ($S \rightarrow n4 \rightarrow n14 \rightarrow n18 \rightarrow n21$).

n9 (24)	*n10* (20)	*n11* (24)	*n12* (28)	*n16* (38)	*n20* (48)	*n24* (58)
n1 (14)	*n2* (10)	*n3* (14)	*n13* (24)	*n17* (34)	*n21* (44)	*n25* (54)
n8 (10)	*S* (0)	*n4* (10)	*n14* (20)	*n18* (30)	*n22* (40)	*n26* (50)
n7 (14)	*n6* (10)	*n5* (14)	*n15* (24)	*n19* (34)	*n23* (44)	*n27* (54)

Fig. 2. An example grid of nodes showing the calculation of the shortest path using Dijkstra's Algorithm.

The example shown in Fig. 2 is for an idealised 2D plane grid rather than a 3D terrain. To apply the LCP algorithm to a 3D terrain, the vertical levels of the nodes should be taken into consideration in determining the movement cost. The movement cost from the start node to a node *n* can be expressed by

$$f(n) = f(n-1) + g(n) + e(n) \quad (1)$$

$$e(n) = w_e \cdot dz, \quad (2)$$

where $f(n)$ is the least cost value from node *S* to node *n* in a 3D terrain, $g(n)$ is the cost value from node *n-1* to node *n* on a 2D plain, $e(n)$ is an additional cost for 3D terrain, w_e is the weighting value for elevation and dz is the difference between the altitude of the node *n-1* and node *n*.

If, for example, all the nodes other than *n13* and *n14* are on the same level, then the cost of paths passing through *n13* or *n14* will be increased as shown in Eq. (2). Nevertheless, if the increased cost is still less than a more circuitous route on the same level, the LCP will include one or both of the two nodes. The cost of changing the altitude (w_e in Eq. (2)) must be set very carefully, because vertical slopes in a pipeline can adversely affect its smooth operation. On an upslope, gravity slows down the flow encouraging accumulation of sludges, whereas gravity accelerates the flow on a downslope making flow control more difficult. For instance, when the pumps driving the flow at the upstream end are stopped, the flow at the downstream end will continue for some time. In addition, pipelines are inflexible and will not be able to hug the contours of the seabed perfectly, causing free spans which can lead to structural problems in the pipeline.

Furthermore, it is inevitable that there will be obstacles on the seabed around the proposed route. There are three ways to avoid them in the LCP algorithm: removing nodes; removing edges; and imposing a very high weighting to the appropriate edges. These methods are all very similar but subtly differ in implementation. The last method avoids the nodes with the

n9 (24)	*n10* (20)	*n11* (24)	*n12* (28)	*n16* (38)	*n20* (48)	*n24* (58)
n1 (14)	*n2* (10)	*n3* (14)	*n13*	*n17* (42)	*n21* (52)	*n25* (62)
n8 (10)	*S* (0)	*n4* (10)	*n14*	*n18* (52)	*n22* (56)	*n26* (66)
n7 (14)	*n6* (10)	*n5* (14)	*n15*	*n19* (62)	*n23* (66)	*n27* (70)

Fig. 3. An example of a grid-based terrain with obstacles.

obstacles by giving a very high weight on the edges leading to obstacles, making the path including such edges extremely expensive. An extreme variation of this is the second method which has no possible path to the obstacles. This approach can easily be applied to a simple graph structure, but is inefficient for a grid-based structure. The first method removes nodes with the obstacles from the nodes array. In this approach the nodes assigned as obstacles are not considered at all, thereby reducing the calculation time.

For this reason, the method of node removal was used for obstacle avoidance in this study. Fig. 3 gives an example of the LCP algorithm for obstacle avoidance. The obstacles are placed on *n13*, *n14*, and *n15*. There are two optimal paths from *S* to *n21*: (*S* → *n3* → *n12* → *n16* → *n21*) and (*S* → *n3* → *n12* → *n17* → *n21*), and the actual cost of these paths is 52.

3. Smoothing algorithm

In a graph structure, Dijkstra's Algorithm has been found to be a very effective LCP technique and the solution found can be regarded as the optimum path with a minimum cost. A good example of such problems is finding the shortest or fastest route from one location to another in a complex road network. In this case towns (or road junctions) and distances (costs) can be assigned as nodes and edges in the graph, and the algorithm places towns and junctions to be visited in the order of an optimum route. However, determining a pipeline route in a grid-based graph presents a slightly different problem, as any sharp bends in such routes are not acceptable. One of the most crucial requirements in marine pipeline is the minimum allowed bend radius, because the bend radius of the pipeline must not be less than this value throughout its entire length to ensure that the stresses in the pipe wall do not exceed the allowable limits usually specified by the design codes. Obviously this requirement calls for a not only 'smooth' but also 'fair' pipeline path.

Indeed, the initial paths generated by Dijkstra's Algorithm usually do not satisfy this requirement. To address this problem, many studies have used spline techniques to smooth out the initial 'raw' optimal path.

Cubic splines are essentially piecewise polynomials of degree 3, and the resulting curves pass through all the given data points. This makes them an eminently successful mathematical form for interpolating between known data points, and hence their attractiveness as a route-smoothing algorithm. Although this technique tends to produce curves with much less undesired points of inflection than other interpolation techniques, such as Lagrangian, there are perhaps too much oscillation for pipeline routes. B-splines, on the other hand, can be made 'fairer', but the control vertices must be found for the entire curve, which can be quite complex.

The most important limitations of the spline techniques in addition to the problems discussed above, in the authors' opinion, is the fact that it is very difficult to ensure that the thus 'smoothed' curve does not touch the obstacles. To address these problems, we propose the use of a Laplacian smoothing algorithm to smooth the optimal route initially determined.

The Laplacian smoothing is an iterative algorithm, which can be used to smooth polylines or polygonal meshes. This algorithm replaces the original vertex on the initial path with a new one determined by a weighted average of neighbouring points in an iterative manner. The smoothing equation is:

$$P_{smooth} = P_{original} + \lambda \bullet L, \tag{3}$$

where λ is a scale factor ($0 < \lambda < 1$) and L is the Laplacian operator. For a vertex v_i, its Laplacian operator can be linearly approximated by the neighbouring vertices v_j as follows:

$$L(v_i) = \sum_{j \in i^*} w_{ij}(v_j - v_i), \tag{4}$$

where i^* is the index set of vertices neighbouring vertex v_i, and w_{ij} is the weighting factor for the edge (i, j) corresponding to vertex v_i. The sum of the weighting factors should be 1.

Unlike spline techniques, the Laplacian smoothing algorithm can be controlled directly and the curvature of the pipeline can always be checked to ensure that the path is 'fair'. The curvature κ at a vertex v_i is defined by the limit of the ratio of change in the slope angle $d\theta$ and the arc length ds, while the radius of the curvature R at vertex v_i is the inverse of the curvature. Assuming that the edge between two successive vertices on the route is sufficiently short, then the radius of the curvature R at the ith vertex v_i can be represented as follows:

$$R(v_i) = \frac{\Delta s}{\Delta \theta}, \tag{5}$$

where Δs is the length from the $(i\text{-}1)$th vertex v_{i-1} to the $(i+1)$th vertex v_{i+1} through v_i and $\Delta\theta$ is the change in the slope angle of the tangent line. An example of a curvature calculation for a polyline is shown in Fig. 4.

The ease of obstacle avoidance is another benefit of the Laplacian smoothing algorithm. The initial pipeline path generated through LCP algorithm avoids obstacles, but avoidance is not guaranteed when splines are used. However, the Laplacian smoothing algorithm changes the position of each vertex in turn in a curve, and therefore it is possible to remedy the situation if the curve is seen to be touching any obstacles during the iterative process.

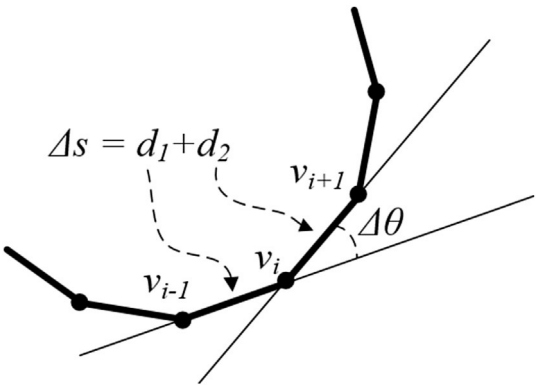

Fig. 4. Curvature calculation for a polyline.

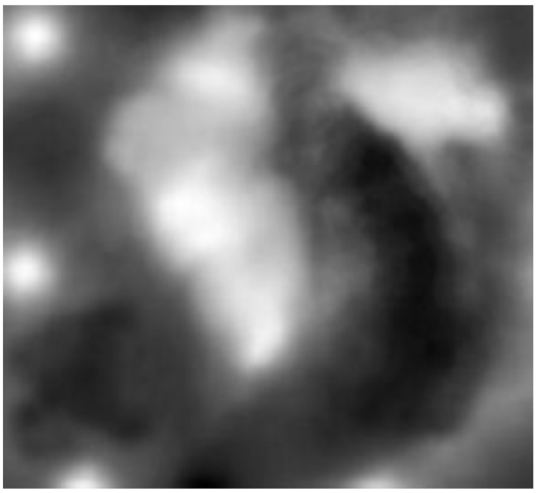

Fig. 5. A height map of the sample seabed terrain.

4. Simple case studies

A computer program was written in C++ for automatic subsea pipeline route design based on the foregoing discussion. 3D graphics were rendered with the Open Graphics Library (OpenGL) application programming interface. As has been discussed earlier, the 3D terrain information is among the most important information in offshore pipeline route design, and the program uses greyscale bitmap images representing the 3D terrain. The vertical level of each grid location is represented by the brightness of a pixel. Dark regions in the image represent valleys and lighter regions represent peaks. The data size of such image files is usually quite small and requires relatively small amount of memory by modern standards. In addition, the images can easily be converted into 3D models for visualisation. Fig. 5 shows a height map of an example seabed terrain used in this case study. The bitmap image comprises 448 (width) × 409 (height) pixels, and its size is only 536 kilobytes (KB). The distance between orthogonally adjacent pixels is 1 km in this example, so the whole bitmap image represents a 3D geographic area of 448 km × 409 km.

In this case study, the start and target points for the pipeline route were taken to be at (50, 120) and (400, 120), respectively, and the optimal pipeline routes were generated for a number of scenarios. In the first place three maximum curvature conditions were investigated without any obstacles. The pipe bend radius must be sufficiently large to prevent stress in

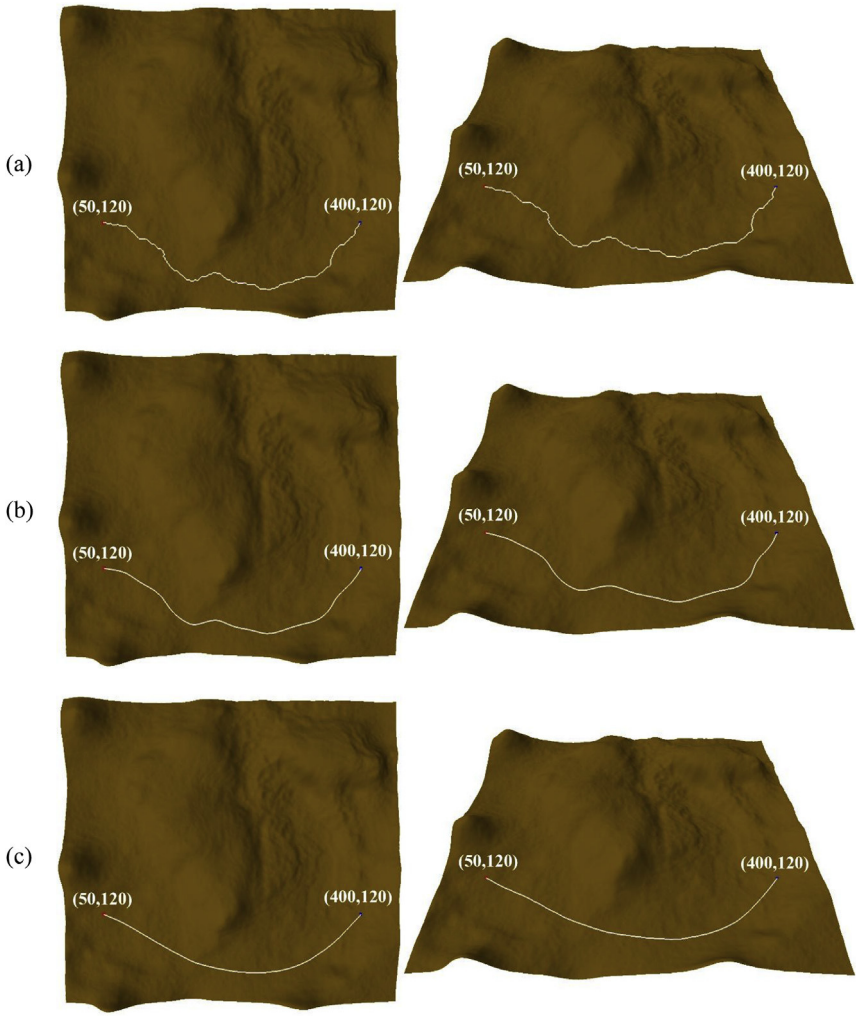

Fig. 6. (a) Raw unsmoothed LCP (1.025 km); (b) smoothed to 30 km bend radius; and (c) smoothed to 90 km bend radius.

Table 1
Number of iterations required to achieve given minimum bend radius using a variety of scale factors.

Minimum radius of curvature, R (km)	Scale factor, λ	Number of iterations, t
1.025 (unsmoothed)	–	0
30	0.1	1344
30	0.5	243
30	0.9	116
90	0.1	28491
90	0.5	5435
90	0.9	2787

the pipe wall, but the radius should not be too large. One further consideration is the need to make the route as straight as possible, because a zigzagging route will not only increase the total length of the pipeline longer but will also require the pipelaying vessel to manoeuvre excessively. Fig. 6 illustrates the path generated for three minimum bend radii. The first path shown in Fig. 6(a) is the initial LCP before being smoothed, and the minimum bend radius of this route was found to be 1.025 km. Fig. 6(b) and (c) show the raw LCP of 6(a) after being smoothed to meet the minimum required bend radii of 30 km and 90 km.

The Laplacian smoothing algorithm is an iterative algorithm, and the minimum bend radius can be controlled by the number of iterations and the scale factor λ in Eq. (3). A greater number of iterations produce 'smoother' curves, and therefore a greater minimum bend radius requires more iterations. In addition, larger scale factors can reduce the number of iterations required, as shown in Table 1. It is interesting to note that different values of scale factors used produce very similar end results, although the number of iterations to achieve them was

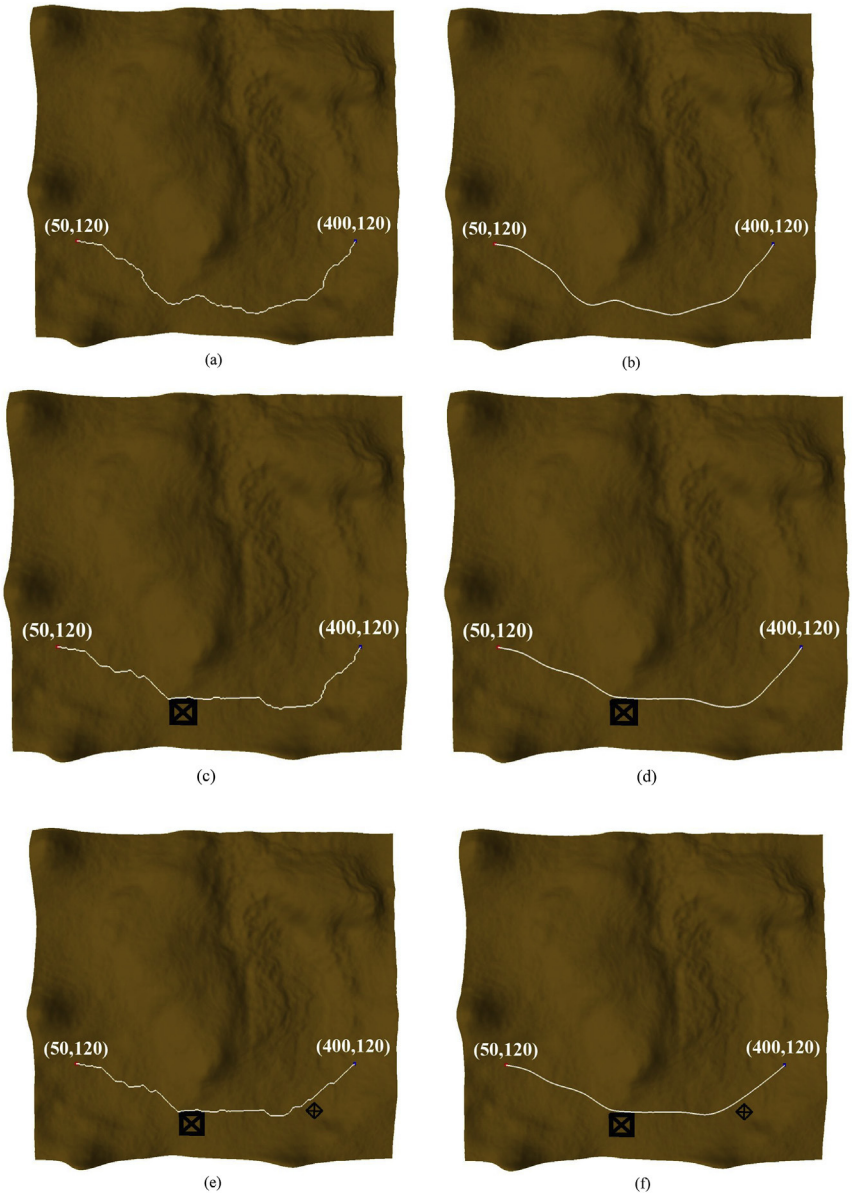

Fig. 7. LCPs obtained in a variety of scenarios: (a) no obstacle, unsmoothed; (b) no obstacle, minimum bend radius = 30 km; (c) one obstacle, unsmoothed; (d) one obstacle, 30 km; (e) two obstacles, unsmoothed; (f) two obstacles, 30 km.

different. For example, the smoothed routes with scale factors of 0.1 and 0.9 differed little, but the number of iterations required was significantly different, as shown in Table 1. Thus, for lower curvature, the use of a higher scale factor is recommended.

Possible obstacles in the way of pipeline routing includes existing subsea structures, coral reefs, fish farms, wildlife preservation areas and wind farms. Some of these require pipelines to be laid at a specified distance from them. In the algorithm used in our study a virtual obstacle was used for each real obstacle, defined with its boundaries at the specified clearance from the real one. Fig. 7 compares the LCP pipeline routes generated for the same start and end points but in different scenarios. The route shown in Fig. 7(a) is the initial LCP without any obstacles and Fig. 7(b) shows this route smoothed to a minimum bend radius of 30 km. The paths shown in Fig. 7(c) and (e) are the raw LCPs with one and two obstacles, respectively, and Fig. 7(d) and (f) are their smoothed versions with a minimum bend radius of 30 km.

All of the three smoothed paths ((b), (d) and (f) in Fig. 7) use the same scale factor of 0.9, but the number of iterations was 116, 235 and 235 respectively. The raw paths of each group (Fig. 7(a), (c) and (e)) passed through 386, 371 and 351 path nodes respectively, and the total length of the smoothed paths (Fig. 7(b), (d) and (f)) were 417.533 km, 395.778 km and 385.441 km respectively. For the situations studied the Dijkstra's algorithm supplemented by Laplacian smoothing was seen to produce routes avoiding obstacles as well as satisfying the desirable minimum bend radius. As a bonus, the developed algorithm found the optimum path almost instantaneously.

5. Concluding remarks

In this study we developed a method of offshore pipeline route design in the presence of obstacles based on the Dijkstra's LCP algorithm and Laplacian smoothing algorithm. The performance of the integrated technique was demonstrated using models of 3D seabed terrain with various obstacles. The developed algorithm can find the least cost path almost instantaneously on demand. Using a PC with Intel Core i3-3220 CPU 3.30 GHz and 16.0 GB, it takes less than 1 s for each case. The speed is undoubtedly much faster than manual manipulation, and this method can definitely minimise the human error during the pipeline route selection process. The Laplacian smoothing algorithm appears to be highly effective and it is simple to implement. Unlike B-spline, this algorithm does not need to find control points, and the generated 'smoothed' curve from the Laplacian algorithm can easily avoid obstacles with only one if statement in C++ code. It has been found that this integrated algorithm can be easily extended to various situations by the device of additional costs, and the system can cover any number of obstacles within reason without any complex manipulation. Furthermore, it gives flexibility to control the fairness of the route designed by continuing the iteration until a desired result is obtained. It was shown that the proposed technique can make obstacle avoiding pipeline route planning much more cost-effective and save time by obtaining the near-optimal solutions in an automatic manner.

References

Berglund T, Jonsson H, Soderkvist I. 2003. An obstacle-avoiding minimum variation B-spline problem. In: Proc. Int. Conf. on Geometric Modeling and Graphics (2003).

Carpenter, G., Callen, R., 1984. Improved procedures for natural gas pipeline routing in Michigan. Environ. Prof. 6, 26–31.

Chakrabarti, S., 2005. Handbook of Offshore Engineering. Offshore Structure Analysis, Inc, Illinois, USA.

Connors J, Elkaim G. 2007a. Analysis of a spline based, obstacle avoiding path planning algorithm. Vehicular Technology Conference, VTC2007-Spring. IEEE 65th, p. 2565–2569.

Connors, J., Elkaim, G., 2007b. Manipulating B-Spline based paths for obstacle avoidance in autonomous ground vehicles. In: Proceedings of the ION National Technical Meeting of the Institute of Navigation, vol. 5, pp. 1081–1088.

Cui, X., Shi, H., 2011. Direction oriented pathfinding in video games. Int. J. Artif. Intell. Appl. 2 (4), 1–11.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271.

DNV, 2012. Offshore Standard, DNV-OS-F101, Submarine Pipeline Systems.

DNV, 2010a. Recommended Practice, DNV-RP-F101, Corroded Pipelines.

DNV, 2011a. Recommended Practice, DNV-RP-F102, Pipeline Field Joint Coating and Field Repair of Linepipe Coating.

DNV, 2010b. Recommended Practice, DNV-RP-F103, Cathodic protection of submarine pipelines by galvanic anodes.

DNV, 2006. Recommended Practice, DNV-RP-F105, Free Spanning Pipelines.

DNV, 2011b. Recommended Practice, DNV-RP-F106, Factory Applied External Pipeline Coatings for Corrosion Control.

DNV, 2010c. Recommended Practice, DNV-RP-F107, Risk Assessment of Pipeline Protection.

DNV, 2010d. Recommended Practice, DNV-RP-F109, On-Bottom Stability Design of Submarine Pipelines.

Feizlmayr, A.H., McKinnon, C., 1999. Lower costs, environmental protection drive future pipeline technologies. Oil Gas J. 97, 83–89.

Feldman, S., Ramona, P., Edwalser, C., Douglas, A., 1995. A prototype for pipeline routing using remotely sensed data and GIS analysis. Remote Sens. Environ. 123–131.

Fraichard T, Ahuactzin J. 2001. Smooth path planning for cars. Conference: robotics and automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on, Volume: 4.

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107.

Matori, A.N., Lee, H.Y., 2009. Suitability Analysis of Subsea Pipeline Route Using GIS. Map Malaysia. GIS Development.

Montemurro, D., Barnett, S., Gale, T., 1998. GIS-based process helps TransCanada select best route for expansion line. Oil Gas J. 96 (25), 63–69.

Ryder, A., 1987. Pipeline routing-experiences from northern Scotland. Pipes Pipelines Int. 27 (39), 5–14.

Yu, C., Lee, J., Munro-Stasiuk, M., 2003. Extensions to least-cost path algorithms for roadway planning. Int. J. Geogr. Inf. Sci. 17 (4), 361–376.