Implementation of the Spline Method for Mobile Robot Path Control

Halit Eren, Chun Che Fung and Jeromy Evans
Curtin University of Technology, Perth, Australia
School of Electrical and Computer Engineering
Kent Street, Bentley, WA, 6102, AUSTRALIA
Tel: 61-8-9266 7903, Fax: 61-8-9266 2584, e-mail: terenh@cc.curtin.edu.au

Abstract

This paper presents an implementation of a Spline Curve Algorithm in a mobile robot path-planning domain. This method is based on a continuous path mapped between two points. The path can easily be updated and modified as the robot moves. This feature makes the algorithm to be suitable for dynamic control as in the case of multiple mobile robots operating in the same environment in cooperative or competitive manner. The algorithm can be integrated with other planning strategies to improve system performance.

1. Introduction

There are various techniques for the implementation of mobile robot path planning and control [1,2,3]. Some of these methods include Artificial Neural Networks and Fuzzy logic [4,5], learned maps [6], dead reckoning and landmarks [7], sonar methods etc. Path planning involves the calculation of a suitable path between two points in space by taking into consideration of the obstructions on the path as well as the tasks to be achieved. Motion planning methods and their implementation for multiple moving objects have been studied by various investigators [8,9,10,11]. In this paper, the path-planning algorithm incorporates the spline method [12,13,14,15] that generates a continuous path within the operational environment. The algorithm accounts for the robot's initial and final orientation, as well as any obstructions on the path. The algorithm can be translated into motor velocity control signals easily.

Currently, the available methods within the spline techniques vary in complexity depending on the application environment. They may be implemented in the form of line-segmented (discontinuous) or smooth and continuous to address the issues such as initial and final orientation of robots. The Line segmented approach requires a greater transit time compared to continuous paths. The continuous techniques typically require increased calculation time, but they integrate the initial and final orientations into the path.

In this paper, first, the spline method algorithm will be discussed briefly with particular emphasis on the practical implementation. Then, a typical optimal path calculation will be introduced supporting the software architecture of the entire system. The comparison of the algorithm with other techniques with emphasis to the advantages and disadvantages will also be given, based on the experience obtained during the implementation process.

2. The Spline Method

A spline interpolation is an algorithm used in CAD and other graphical applications [13,14,15]. Mathematically, it is described by a piecewise cubic (or higher order) polynomial functions constrained by certain continuity conditions across various curve sections [15]. It is defined entirely by a small series of control points that indicate only the general shape of the curve as illustrated in Figure 1. The entire path is formed by control points and the boundaries between the adjacent sections.

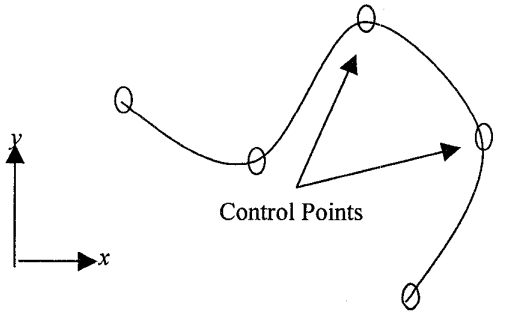

Figure 1. A series of control points defining the path

0-7803-5276-9/99/$10.00 © 1999 IEEE

In a common cubic spline method, each section of the curve may be described by parametric equations, as:

$$\begin{aligned} x(u) &= a_x u^3 + b_x u^2 + c_x u + d_x \\ y(u) &= a_y u^3 + b_y u^2 + c_y u + d_y \\ &\text{where } (0 \le u \le 1) \end{aligned} \quad (1)$$

Representing the functions in a matrix equivalent form, a position $\boldsymbol{P}(u)$ in the curve may be described as:

$$\boldsymbol{P}(u) = [u^3\ u^2\ u\ 1] \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \quad (2)$$

Hence, the actual function is defined by the coefficients a, b, c and d, which need to be determined for each section. The numerical values for these four unknowns can be calculated using the boundary conditions, $\boldsymbol{P}(0)$ and $\boldsymbol{P}(1)$, and the first (or higher) derivatives, $\boldsymbol{P}'(0)$, and $\boldsymbol{P}'(1)$ of the function.

This implementation is based on the Cardinal Spline algorithm, which uses four consecutive control points to set the boundary conditions for each segment. For instance, in the section in Figure 2, the assumption is that the gradient at the control points p_k and p_{k+1} is proportional to the vectors p_{k-1} p_{k+1}(A) and p_k p_{k+2}(B) respectively.

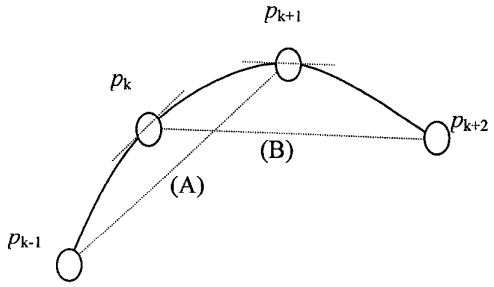

Figure 2. Four control points and the boundary conditions

The boundary conditions become:

$$\begin{aligned} \boldsymbol{P}(0) &= \boldsymbol{p}_k \\ \boldsymbol{P}(1) &= \boldsymbol{p}_{k+1} \\ \boldsymbol{P}'(0) &= \tfrac{1}{2}(1 - t)(\boldsymbol{p}_{k+1} - \boldsymbol{p}_{k-1}) \\ \boldsymbol{P}'(1) &= \tfrac{1}{2}(1 - t)(\boldsymbol{p}_{k+2} - \boldsymbol{p}_k) \end{aligned} \quad (3)$$

These boundary conditions can be substituted into the spline curve matrix to yield the form

$$P(u) = [u^3\ \ u^2\ \ u^1\ \ 1] \cdot M_c \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix} \quad (4)$$

where $\boldsymbol{M}_c$ is the Cardinal Matrix:

$$M_c = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \quad (5)$$

where $s = \tfrac{1}{2}(1-t)$ (6)

The value t is the tension parameter, which controls how tight or loose the Cardinal spline fits the control points.

Expanding back into the polynomial form gives

$$\begin{aligned} \boldsymbol{P}(u) &= \boldsymbol{p}_{k-1}(-su^3 + 2su^2 - su) + \\ &\quad \boldsymbol{p}_k[(2-s)u^3 + (s-3)u^2 + 1] + \\ &\quad \boldsymbol{p}_{k+1}[(s-2)u^3 + (3-2s)u^2 + su] + \\ &\quad \boldsymbol{p}_{k+2}(su^3 - su^2) \\ &= \boldsymbol{p}_{k-1}\mathrm{CAR}_0(u) + \boldsymbol{p}_k\mathrm{CAR}_1(u) + \\ &\quad \boldsymbol{p}_{k+1}\mathrm{CAR}_2(u) + \boldsymbol{p}_{k+2}\mathrm{CAR}_3(u) \end{aligned} \quad (7)$$

In the above expression, CAR_k (k:[0,3]) are the Cardinal blending functions – they blend the boundary constraints to obtain each coordinate position along the curve. For discrete values of u and constant s, these blending functions may be represented in look-up tables.

In the case of zero tension ($t = 0$, $s = \tfrac{1}{2}$), the Cardinal blending functions reduce to:

$$\begin{aligned} \mathrm{CAR}_0 &= -\tfrac{1}{2}u^3 + u^2 - \tfrac{1}{2}u \\ \mathrm{CAR}_1 &= 1\tfrac{1}{2}u^3 - 2\tfrac{1}{2}u^2 + 1 \\ \mathrm{CAR}_2 &= -1\tfrac{1}{2}u^3 + 2u^2 + \tfrac{1}{2}u \\ \mathrm{CAR}_3 &= \tfrac{1}{2}u^3 - \tfrac{1}{2}u^2 \end{aligned} \quad \text{for } 0 \le u \le 1 \quad (8)$$

These four functions can be represented in look-up tables for known values of u (or alternatively, using a fixed-point representation of u). In the simplest, but slowest implementation, the spline calculation for each interior point involves four multiplications and four additions.

$$\begin{aligned} p(u) &= \boldsymbol{p}_{k-1}\mathrm{CAR}_0(u) + \boldsymbol{p}_k\mathrm{CAR}_1(u) + \\ &\quad \boldsymbol{p}_{k+1}\mathrm{CAR}_2(u) + \boldsymbol{p}_{k+2}\mathrm{CAR}_3(u) \end{aligned} \quad (9)$$

An alternative and faster implementation is possible using incremental calculations for each successive point.

3. Implementation and Path Planning

To define a basic path from one location to another, accounting for the initial and final orientation, seven control points are required as shown in Figure 3. These points serve different purposes in the path definition, as:

Control points (2) and (6) account for the starting and ending locations A and B respectively.

Control points (1), (3) and (5), (7) are required to define the initial and final orientation.

Control point (4) is optional and is used to control the curvature of the path.

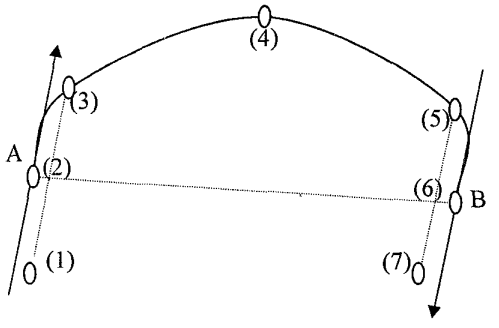

Figure 3 Control points defining the curved path

By experimentation, it was found that offsetting points (1), (3) and (5), (7) from the desired initial and final vectors (depending on the angle ∠AB) ensures the path is straight immediately after A and before B. Offsetting the points by the same degree maintains the initial and final orientation. That is because the gradient at any control point k is proportional to the vector formed by the succeeding and preceding control points, $\boldsymbol{p}_{k-1}\boldsymbol{p}_{k+1}$.

In calculating the path, all interior points in the segments (2) → (3), (3) → (4), (4) → (5) and (5) → (6) must be determined. The segment (1) → (2) and (6) → (7) do not form part of the actual path. They are required only in the calculation of their adjacent segments. The distances from A and B to the control points are chosen to be proportional to the initial and the final velocity of the robot. Hence, the location of control point (4) is proportional to the vector sum of (2)→(3), (6)→(5) and ½(AB).

Although the implementation of the spline curve can considerably be complex, in this simple case, only a few vector additions are required in determining the location of the control points. Furthermore, for the complex cases involving obstacle avoidance or collision detection, any additional required control points are simply inserted into the path at the appropriate location.

In a typical implementation, an optimal path calculation, which maps actual real-space scene into a configuration space can be made. Effectively, the robot is moved around each obstacle and the locus traced for a single point within its area, e.g., the centroid as in Figure 4. If no overlap of the objects occur in configuration space, it is possible for robot to maneuver between the obstacles.

The optimal (shortest) path between the obstacles is found by building a visibility-based weighted graph that connects starting point A to the end point B. This can be done by ray-casting a line from each vertex to every other and adding it to the graph and by meeting the following conditions:

- The segment cannot be added to the existing graph.
- The segment must not intersect with any edge in configuration space (unless it is an edge itself).

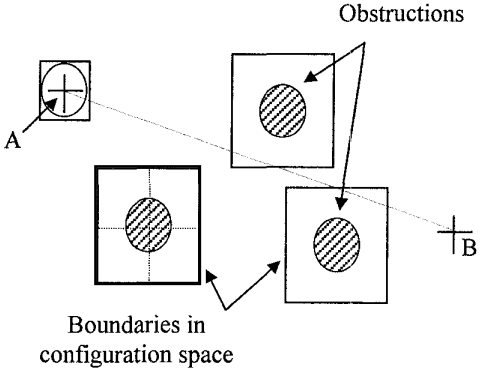

Figure 4. Mapping of objects to configuration space

The result of the graph building process yields a graph as shown in Figure 5. An efficient implementation of this algorithm is achieved by assuming all objects in configuration space are rectangular, hence all edges are either vertical or horizontal. This assumption simplifies the calculation considerably.

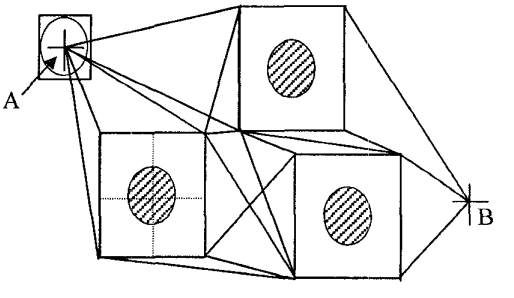

Figure 5. Connecting starting and ending locations

A connected-graph-searching algorithm can then be used to determine the shortest path. This effectively yields a list of vertices through which robot must pass. These

positions are mapped directly into control points for the spline path as in Figure 6.

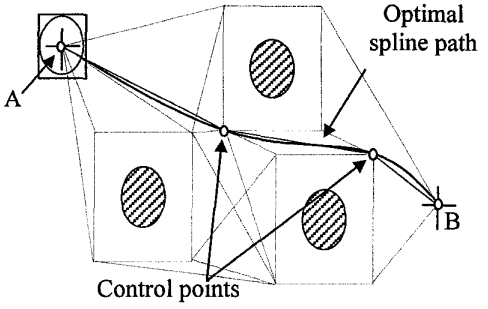

Figure 6. Optimal path mapping vertices to control points

Functionality for the initial and final orientation within the optimal spline path is included by the four additional control points at the start and end of the path, points (1), (3) and points (5), (7) respectively, provided that they do not cause an intersection in configuration space. This is advantageous since the spline algorithm can be incorporated with the optimal path calculation by simply mapping appropriate coordinates in the control points.

4. Implementation on Mobile Robots

Experimentation of the spline algorithm is implemented on a set of six miniature mobile robots operating on a 2m×3m field, as shown in Figure 7. The implemented algorithm was run on a Pentium computer and the control signals were down loaded via a radio communication link.

In addition to providing basic, optimal continuous path control, the spline algorithm permits a number of elementary player activities to be implemented. This is achieved simply by the definition of appropriate control points around the target areas. Some typical activities are listed below:

- Follow a path in a predefined form.
- Go To Location – move to a certain location with certain orientation and stop.
- Shoot – charge towards an object.
- Disposition an object by pushing in desired direction
- Intercept a moving object by predicting the trajectory and estimating the interception point.
- Block the move of other robots.

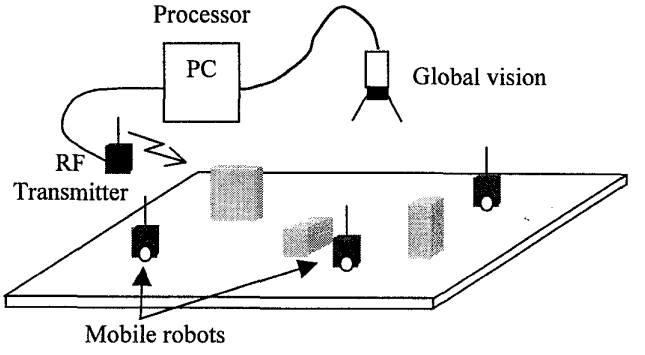

Figure 7. Multiple robot system for implementation

During execution, the spline path algorithm calculates the path motion which in turn is translated into real-time motor velocity control signals. The actual path is represented as a list of (x, y) locations, interpolated between the control points. By considering each pair of successive points in the path, any basic motor control strategy can be employed.

Figure 8 illustrates a small part of a spline segment. Each successive pair of points yield a displacement (d) and angle (θ) error in the path.

To maintain a near constant motor velocity, the excitation of the two driving motors may be calculated as:

$$V_L = V_c + k_\theta \cdot \theta \qquad (10)$$
$$V_R = V_c - k_\theta \cdot \theta$$

Where θ is the angle error, k_θ is a constant coefficient, V_c is a constant voltage setting the general velocity, and V_L, V_R are the velocity control levels for the two motors. This basic control algorithm gives reasonably good performance and permits the transit time to be easily approximated. The transit time is an essential factor for a trajectory interception when the path is calculated.

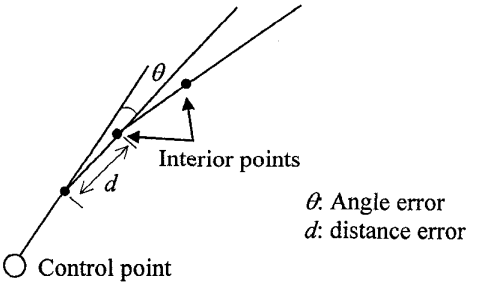

Figure 8. Interior points yield angles and distance errors

An alternative control scheme may use the displacement error instead of the constant V_c component [1]. A property of the spline path is that it generally features more points around the sharper parts of the curve. Such an implementation results in the robot slowing for the turns and speeding up for the straighter parts of the path. Unlike path motion planning based on straight line segments which may result in overshooting when the end is approached. In this manner, the spline algorithm technique is certainly advantageous in many applications.

Provided that a sufficient number of interior positions are calculated between each control point, the velocity control signals permit the robots to follow a smooth path. However, too few interior points result in 'jerky' motion and too many interior points will demand additional computation by the processor. An alternative approach implemented has used a small number of interior points, converted to velocity control signals, which are then low-pass filtered to generate a smooth velocity signal.

5. The Software and Discussions of Results

The task architecture of the system is presented in Figure 9 to demonstrate the implementation of the spline algorithm. The vision system periodically updates the perceived position information, which is then used to determine the location of the path's control points. Another periodic task calculates the interior points and velocity control signals output to the robot motors. Whenever changes in the control points are made, this task is notified and makes its own copy of the new locations of the points. This approach permits the output task to generally operate at a frequency completely independent of the other tasks.

A significant advantage of the spline algorithm in path planning has been its flexibility of being easily incorporated into other approaches. To do it, all that is required is a list of control points (coordinates), which can be automatically translated to a continuous path. Other advantages include:

- Simplicity in the incorporation of constraints on the path. For instance, if the path must pass through a particular location, this can be done by insertion of an appropriate control point.
- The implementation for dynamic robot paths and completely predefined paths are essentially identical.
- It gives continuity conditions in the motion for high order derivatives. For instance, the 2^{nd} or 3^{rd} derivatives avoid sudden accelerations or jerk.
- The path can be calculated dynamically by simply reassessing the location of the control points. This requires a small number of calculations for each frame. The majority of calculation can be performed in a background periodic task that uses a current copy of the control points.
- It is possible to use the control points to define basic robot activities.

Nevertheless, the method has disadvantages such as requiring more calculation time when, for example, compared to a dynamic reactive path control method. Also, certain threshold parameters such as the relationship between velocity and control point locations may need to

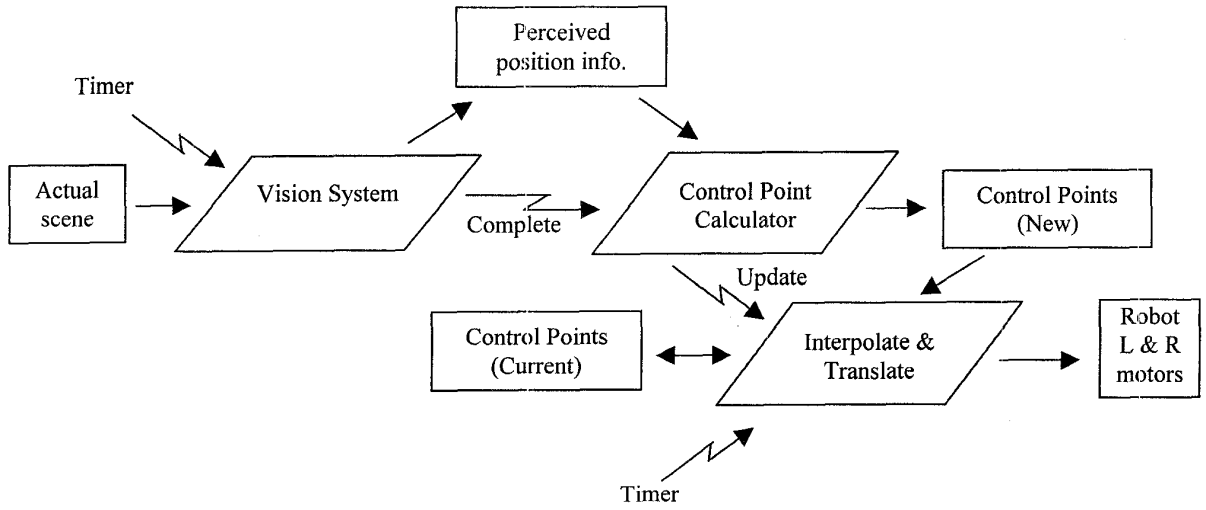

Figure 9. Task architecture of the robot path control system

be measured by experimentation to ensure the robot follows the defined path. It has been found that the actual number of interior points to calculate is also application dependent, and a path without sufficient curvature can temporarily overshoot the destination. In certain situations, by selection of the appropriate parameters, these disadvantages can be outweighed by the flexibility offered by the utilization of control points in defining the entire path.

6. Conclusions

Present path planning techniques vary in performance and ways of implementation. The spline algorithm has been implemented on a multiple mobile robot system. It has provided a highly flexible solution for path planning and control. It can be integrated with other techniques to improve performance by ensuring a continuous path to always be maintained. This algorithm may be suitable to other applications where modification of the path is necessary during the operations. In this study, it is found that addition of control points guarantees continuity in the path without significant deviations.

References

[1] Kim, J., (1998), "Multi-Robot Cooperative System Development," Dept. of Electrical Engineering, KAIST, Korea.

[2] Veloso, M. *et al.*, (1998), "The CMUnited-97 small robot team", Robocup-97: The First Robot World Cup Soccer Games and Conferences, Springer Verlag, Berlin.

[3] Kakikura, M., (1988), "Path finding problems of autonomous robots," Researches of the Electrotechnical Laboratory, no. 85, pp. 1-104.

[4] Han, M.W.; Kolejka, T., (1994), "Artificial neural networks for control of autonomous mobile robots," Intelligent Manufacturing Systems (IMS`94), IFAC Workshop, Pergamon, Oxford, UK; pp. 157-62.

[5] Woong, G. H, Seung, M. B., Tae, Y. K., (1997), "Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots," IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, IEEE, New York, NY, vol. 3, pp. 2747-51.

[6] Kampmann, P.; Schmidt, G., (1991), "Indoor navigation of mobile robots by use of learned maps," Information Processing in Autonomous Mobile Robots. Proceedings of the International Workshop. Springer-Verlag, Berlin, Germany; pp. 151-69

[7] Makela, H.; Koskinen, K., (1991), "Navigation of outdoor mobile robots using dead reckoning and visually detected landmarks," Fifth International Conference on Advanced Robotics. Robots in Unstructured Environments, IEEE, New York, NY, vol. 2, pp. 1051-6.

[8] Hwang, Y. K., (1995), "Motion planning for multiple moving objects," Proceedings. IEEE International Symposium on Assembly and Task Planning, IEEE Comput. Soc. Press, Los Alamitos, CA, pp. 400-5.

[9] Tsai, Y. C.; Szu, W. K.; Hsu, J. Y. J., (1994), "A two-phase navigation system for mobile robots in dynamic environments," Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. Advanced Robotic Systems and the Real World, IEEE, New York, NY, vol. 1, pp. 306-13.

[10] Alvarez, J. C.; Lopez, H.; Sirgo, J. A.; Sanchez, L., (1996), "A dynamic path planner for autonomous vehicles," Intelligent Components for Autonomous and Semi-Autonomous Vehicles. (ICASAV'95). IFAC Workshop. Pergamon, Oxford, UK;. pp. 51-6.

[11] Bourbakis, N. G., (1997), "A traffic priority language for collision-free navigation of autonomous mobile robots in dynamic environments," IEEE Transactions on Systems, Man and Cybernetics, Part B, vol.27, no.4; pp. 573-87.

[12] de Figueiredo, R. J. P.; Markandey, V., (1988), "Spline-based algorithms for shape from shading (for robot vision application)," Proceedings of the SPIE, The International Society for Optical Engineering, vol. 850; pp. 152-5.

[13] Boehm, W., (1986), "Multivariate spline algorithms," Mathematics of Surfaces. Proceedings of a Conference. Clarendon Press, Oxford, UK; pp. 197-215.

[14] Parry, W. J., Hanson, J. V., (1984), "A real-time digital interpolator," International Communications and Energy Conference, IEEE, New York, NY, pp. 188-91.

[15] Spath, H., (1995), "One Dimensional Spline Interpolation Algorithms," A. K. Peters, Wellesley, Massachusetts.