Step 4: The user's query is translated into

SELECT	ENAME
FROM	EMP
WHERE	AGE ≥ 0 AND AGE ≤ 23 OR
	AGE ≥ 70 AND AGE ≤ 100.

VI. Conclusions

We have presented a new method for fuzzy query translation based on the α-cuts operations of fuzzy numbers for relational database systems. The proposed method emphasizes friendliness and flexibility for the inexperienced users. We have described the method of fuzzy query translation in details, where we allow the retrieval conditions of the user's SQL queries to be represented by fuzzy terms described by fuzzy numbers. Based on the proposed method, we have implemented a fuzzy query translator for relational database systems by using Turbo C version 3.0 on a PC/AT for translating user's fuzzy queries into precise queries. Because the user can construct his queries intuitively and can choose different retrieval threshold values for fuzzy information retrieval, the existing database systems will be more friendly and more flexible to the users.

References

[1] B. P. Buckles, "Information-theoretical characterization of fuzzy relational databases," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, pp. 74–77, Jan./Feb. 1983.

[2] S. M. Chen, J. S. Ke, and J. F. Chang, "Techniques of fuzzy query translation for database systems," in *Proc. 1986 Int. Computer Symp.*, Tainan, Taiwan, R.O.C., Dec. 1986, pp. 1281–1290.

[3] S. M. Chen and J. Y. Wang, "Document retrieval using knowledge-based fuzzy information retrieval techniques," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 793–803, May 1995.

[4] S. M. Chen, "A fuzzy reasoning technique based on the α-cuts operations of fuzzy numbers," in *Proc. 2nd Int. Conf. Automation Technology*, Taipei, Taiwan, R.O.C., July 1992, pp. 147–154.

[5] C. J. Date, *An Introduction to Database Systems*, 3rd ed. Reading, MA: Addison-Wesley, 1981.

[6] J. Giarratano and G. Riley, *Expert Systems: Principles and Programming*. Boston, MA: PWS-Kent, 1989.

[7] G. T. Her and J. S. Ke, "A fuzzy information retrieval system model," in *Proc. 1983 National Computer Symp.*, Taipei, Taiwan, R.O.C., Dec. 1983, pp. 147–155.

[8] W. T. Jong and S. M. Chen, "Fuzzy query translation techniques for relational database systems," in *Proc. Int. Joint Conf. CFSA/IFIS/SOFT'95 Fuzzy Theory Applications*, Taipei, Taiwan, R.O.C., Dec. 1995, pp. 95–100.

[9] M. Kamel, B. Hadfield, and M. Ismail, "Fuzzy query processing using clustering techniques," *Inf. Process. Manage.*, vol. 26, no. 2, pp. 279–293, 1990.

[10] A. Kaufmann and M. M. Gupta, *Fuzzy Mathematical Models in Engineering and Management Science*. Amsterdam, The Netherlands: North-Holland, 1988.

[11] S. K. Chang and J. S. Ke, "Database skeleton and its application to fuzzy query translation," *IEEE Trans. Software Eng.*, vol. SE-4, pp. 31–43, Jan. 1978.

[12] ——, "Translation of fuzzy queries for relational database systems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, pp. 281–294, July 1979.

[13] D. Lucarella and R. Morara, "FIRST: Fuzzy information retrieval system," *J. Inf. Sci.*, vol. 17, no. 1, pp. 81–91, 1991.

[14] Y. Takahashi, "A fuzzy query language for relational databases," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, pp. 1576–1579, Nov./Dec. 1991.

[15] R. R. Yager and D. P. Filev, *Essentials of Fuzzy Modeling and Control*. New York: Wiley, 1994.

[16] M. S. Yeh and S. M. Chen, "A new method for fuzzy query processing using automatic clustering techniques," *J. Comput.*, vol. 6, pp. 1–10, Spring 1994.

[17] L. A. Zadeh, "Fuzzzy sets," *Inf. Control*, vol. 8, pp. 338–353, 1965.

[18] ——, "The concept of a linguistic variable and its application to approximate reasoning," *Inf. Sci.*, vol. 8, no. 3, pp. 199–249, 1975 (part 1); vol. 8, no. 4, pp. 301–357, 1975 (part 2); vol. 9, no. 1, pp. 43–80, 1976 (part 3).

[19] M. Zemankova, "FIIS: A fuzzy intelligent information system," *Data Eng.*, vol. 12, no. 2, pp. 11–20, June 1989.

[20] M. Zemankova and A. Kandel, "Implementing imprecision in information systems," *Inf. Sci.*, vol. 37, pp. 107–141, Dec. 1985.

[21] H. J. Zimmermann, *Fuzzy Set Theory and its Applications*, 2nd ed. Boston, MA: Kluwer, 1991.

A Counterexample to the Alexopoulos–Griffin Path Planning Algorithm

Robert A. Conn, Javier Elenes, and Moshe Kam

***Abstract*— The planar stationary-obstacle path-planning problem for polygonal obstacles has been correctly and completely solved by Lozano-Perez and Wesley [3], i.e., a global, optimal algorithm was provided which requires $O(\mu^2 \log \mu)$ computation time, where μ is the number of obstacle-faces in the scene. That algorithm is known as the *VGRAPH* algorithm. Two variants of *VGRAPH* have been developed to solve the same problem in $O(\mu^2)$ computation time [2], [5]. Our paper discusses a recent algorithm proposed by Alexopoulos and Griffin [1], called *V*GRAPH*, which also claims to provide an optimal solution. We demonstrate by counter-example that *V*GRAPH* is neither global nor optimal.**

I. Review of V^*GRAPH Notation [1]

- s is the initial robot position.
- t is the goal position.
- Obstacles are assumed to be simple polygons.
- Numerals refer to vertices of obstacles. The vertices of each obstacle are numbered consecutively in increasing order along a tour of the polygon. Without loss of generality, we shall assume numbering in the counter-clockwise direction.
- A *node* is either a vertex of an obstacle, the starting point s, or the goal point t.
- A *w-visible vertex* is a vertex which is visible from the node w.
- A *w-visible path* is a subset of the set of w-visible vertices. To be a w-visible path, this subset must contain a maximal consecutive sequence of vertices from the same obstacle. For example, the maximal consecutive sequences of $\{1, 2, 4, 6, 7, 8, 9\}$ are $\{1, 2\}, \{4\}$, and $\{6, 7, 8, 9\}$.
- An *extreme vertex* of a w-visible path is either a minimum or a maximum of the elements of the path.

Manuscript received August 12, 1995; revised May 11, 1996. This work was supported by the National Science Foundation through PYI award ECS 9057578 and Research Grant ECS 9216588.

The authors are with the Data Fusion Laboratory, Electrical and Computer Engineering Department, Drexel University, Philadelphia, PA 19104 USA (e-mail: kam@lorelei.ece.drexel.edu).

Publisher Item Identifier S 1083-4419(97)03888-0.

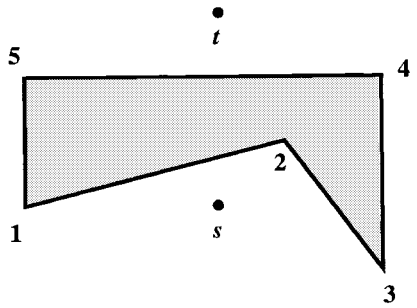

Fig. 1. Counter-example: starting point (s), obstacle (defined by vertices 1–5), and goal point (t).

- An *obtuse vertex* of an obstacle is a vertex with an interior angle greater than π radians.

II. The Algorithm in [1][1]

```
1  begin
2      P := {s};
3      Open := {s};
4      g(s) := 0;
5      f(s) := 0;
6      repeat
7          w := {i ∈ Open: f(i) ≤ f(j), for all
                j ∈ Open};
8          remove w from Open
9          put w in P
10         VV := {w-visible vertices not in Open};
11         EV := {extreme vertices of the w-visible
                       paths in VV};
12         AV := EV −{obtuse vertices in EV};
13         for each vertex i in AV do
14             h(i) := Euclidean distance d(i,t) from
                    i to t;
15             g(i) := g(w) + d(w,i);
16             f(i) := g(i) + h(i);
17             put i in Open
18     until ( (Open = ∅) or (w = t) );
19     if (Open = ∅) then
20         exit with failure;
21     if (w = t) then
22         exit with P;
23 end.
```

III. Counter-Example

Fig. 1 shows the scenario for the counter-example. The coordinates of the robot, the goal, and the vertices are

$$
\begin{aligned}
s &= (3,1)\\
t &= (3,4)\\
\text{“1”} &= (0,1)\\
\text{“2”} &= (4,2)\\
\text{“3”} &= (5.5,0)\\
\text{“4”} &= (5.5,3)\\
\text{“5”} &= (0,3).
\end{aligned}
$$

[1]Taken directly from [1, p. 320]. Minor notational changes have been made in lines 13, 19, and 21.

Using the above scenario as the input to the algorithm of [1] (Section II in this correspondence), we compute the output. The numbers to the left of each assignment indicate the line number of the algorithm which requires that assignment. Comments appear to the right.

2	$P := \{s\}$;	Initialization
3	Open $:= \{s\}$;	
4	$g(s) := 0$;	
5	$f(s) := 0$;	
7	$w := s$;	s is the only element in $Open$
8	Open $:= \phi$;	Remove s from $Open$
9	$P := \{s\}$;	s is already in P
10	VV $:= \{1,2,3\}$;	The vertices visible from s
11	EV $:= \{1,3\}$;	Extreme vertices of sequence “1,” “2,” “3”
12	AV $:= \{1,3\}$;	Both “1” and “3” are nonobtuse vertices
13	$i :=$ “1”;	First element in AV
14	$h(\text{“1”}) := 4.24$;	$d(\text{“1,”}\ t)$
15	$g(\text{“1”}) := 3$;	$g(s) + d(s, \text{“1”})$
16	$f(\text{“1”}) := 7.24$;	$g(\text{“1”}) + h(\text{“1”})$
17	Open $= \{1\}$;	Add i to $Open$
13	$i :=$ “3”;	Second element in AV
14	$h(\text{“3”}) := 4.72$;	$d(\text{“3,”}\ t)$
15	$g(\text{“3”}) := 2.69$;	$g(s) + d(s, \text{“3”})$
16	$f(\text{“3”}) := 7.41$;	$g(\text{“3”}) + h(\text{“3”})$
17	Open $:= \{1,3\}$;	Add i to $Open$
7	$w :=$ “1”;	$f(\text{“1”})$ is smaller than $f(\text{“3”})$
8	Open $:= \{3\}$;	Remove “1” from $Open$
9	$P := \{s,1\}$;	Add “1” to P
10	VV $:= \{2,5\}$;	Vertices “5,” “2,” and “3” are visible from “1;” however, vertex “3” is in $Open$
11	EV $:= \{2,5\}$;	“5” and “2” are both extreme vertices in sequences of length one
12	AV $:= \{5\}$;	The interior angle of vertex “2” is greater than π radians
13	$i :=$ “5”;	First element in AV
14	$h(\text{“5”}) := 3.16$;	$d(\text{“5,”}\ t)$
15	$g(\text{“5”}) := 5.0$;	$g(\text{“1”}) + d(\text{“1,”}\ \text{“5”})$
16	$f(\text{“5”}) := 8.16$;	$g(\text{“5”}) + h(\text{“5”})$
17	Open $:= \{3,5\}$;	Add “5” to $Open$
7	$w :=$ “3”;	$f(\text{“3”})$ is less than $f(\text{“5”})$
8	Open $:= \{5\}$;	Remove “3” from $Open$
9	$P := \{s,1,3\}$;	Add “3” to P
10	VV $:= \{1,2,4\}$;	Vertices “1,” “2,” and “4” are visible from vertex “3”
11	EV $:= \{1,2,4\}$;	Vertices “1,” “2,” and “4” are all extreme vertices
12	AV $:= \{1,4\}$;	Vertex “2” is obtuse
13	$i :=$ “1”;	First element of AV
14	$h(\text{“1”}) := 4.24$;	$d(\text{“1,”}\ t)$
15	$g(\text{“1”}) := 8.28$;	$g(\text{“3”}) + d(\text{“3,”}\ \text{“1”})$
16	$f(\text{“1”}) := 12.52$;	$g(\text{“1”}) + h(\text{“1”})$
17	Open $:= \{1,5\}$;	Add “1” to $Open$
13	$i :=$ “4”;	Second element of AV
14	$h(\text{“4”}) := 2.69$;	$d(\text{“4,”}\ t)$
15	$g(\text{“4”}) := 5.69$;	$g(\text{“3”}) + d(\text{“3,”}\ \text{“4”})$
16	$f(\text{“4”}) := 8.38$;	$g(\text{“4”}) + h(\text{“4”})$

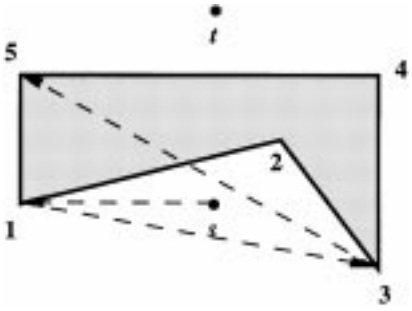

Fig. 2. Output of the [AG] algorithm.

17	Open := $\{1, 4, 5\}$;	Add "4" to $Open$
7	$w :=$ "5";	f("5") is less than f("1") and f("4")
8	Open := $\{1, 4\}$;	Remove "5" from $Open$
9	$P := \{s, 1, 3, 5\}$;	Add "5" to P.

At this point, we stop tracing the algorithm, since the output path has intersected the obstacle. Note that the algorithm provides no possibility of removing a vertex from P. Therefore, whatever the final output of the algorithm for our scenario, the path will always begin with the four-vertex sequence $\{s, 1, 3, 5\}$. Fig. 2 shows the path which follows this sequence P.

The algorithm of [1] is not global—no valid solution is found for our scenario (though a solution definitely exists, namely $\{s, 1, 5, t\}$). Furthermore, the algorithm is not optimal—an optimal solution sequence will be composed of optimal subsequences, while the output of the algorithm contains the nonoptimal subsequence $\{1, 3, 5\}$.

IV. Failure of the Algorithm

It is claimed in [1] that the algorithm of Section II employs the A^* algorithm in [4]. Unfortunately this claim is incorrect. One difficulty stems from line 9 of the algorithm in Section II. *Every* vertex that we expand becomes part of our final path. The authors of [1] have missed a crucial step of the A^* algorithm—the maintenance of pointers between the nodes of P, and the updating of those pointers at each iteration to reflect the current best path [4, Step 7].

V. Conclusion

We have shown (by counter-example) that the V^*GRAPH algorithm of [1] is neither global nor optimal. The reason lies in the graph search procedure used in [1]. This procedure, contrary to the claim of the authors, does not employ the A^* algorithm in [4].

VI. Epilogue—Performance Time

The authors of [1] have computed the worst-case performance of their algorithm to be $O(\mu^2 \log \mu)$. This makes V^*GRAPH more expensive computationally (in the worst case) than the modified $VGRAPH$ algorithms of [2] and [5] which have a worst-case performance of $O(\mu^2)$. The claim is made in [1] that the advantage of V^*GRAPH lies in its "average case performance." We note in passing that this claim is not meaningful without a precise definition of "average case." In order to make a meaningful definition of "average case performance," we shall need answers to difficult questions such as *What is an average polygon?* and *What is an average arrangement of m polygons?*

References

[1] C. Alexopoulos and P. M. Griffin, "Path planning for a mobile robot," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 318–322, Mar. 1992.
[2] T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility-polygon search and Euclidean shortest paths," in *Proc. 26th IEEE Symp. Foundations Computer Science*, Portland, OR, 1985, pp. 155–164.
[3] T. Lozano-Perez and M. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, 1979.
[4] N. J. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980, sec. 2.2.
[5] E. Welzl, "Constructing the visibility graph for n line segments in $o(n^2)$ time," *Inf. Process. Lett.*, vol. 20, pp. 167–171, 1985.

A Note on "Solving the Find-Path Problem by Good Representation of Free Space"

Yongji Wang

***Abstract*— The conditions of a pure translation and a pure rotation for car-like robots and dual-drive robots are derived. Based on these conditions the suitability of the path planning algorithm developed in [1] for each of the two kinds of mobile robots is discussed.**

I. Introduction

In the development of a controller for an intelligent wheeled mobile robotic system such as a car-like robot or a basic dual-motor four wheeled robot, the first step is path planning which can be described as follows: Given a robot with an initial location and orientation, a goal location and orientation, and a set of obstacles in space, find a continuous path which avoids collection with obstacles along the way.

Various methods for dealing with the find-path problem have been developed in the past [2]. Compared with other algorithms available, the algorithm developed in [1] is well known for its advantages. It is quite fast and finds good paths which generously avoid obstacles rather than barely avoiding them. In this correspondence, we derive the conditions for two basic motions that are required in the implementation of a controller for following the path generated by the algorithm in [1], i.e., a pure translation and a pure rotation. In particular, we consider the most widely used two kinds of robots, i.e., car-like robots and dual-drive robots. Comparing these conditions with the assumption of the reference point defined in [1], we will show that in most cases the pure rotation required in [1] is not realistic.

II. Main Results

The motion of a robot moving on a plane can be described by the position of a reference point $P(x_p, y_p)$ in a given coordinate frame and the orientation angle θ of the robot (see Fig. 1). The general constraint equation describing the dependence of the three parameters for car-like robots and dual drive robots is [3], [4]

$$a\frac{d\theta}{dt} = \frac{dy_p}{dt}\cos\theta - \frac{dx_p}{dt}\sin\theta$$

or

$$a\frac{d\theta}{dx} = f'(x_p)\cos\theta - \sin\theta \tag{1}$$

Manuscript received September 4, 1995; revised May 11, 1996.

The author is with the Department of Mechanical Engineering, Edinburgh University, Edinburgh EH9 3JL, U.K.

Publisher Item Identifier S 1083-4419(97)03889-2.