4

SIAM J. COMPUT.
Vol. 13, No. 3, August 1984

© 1984 Society for Industrial and Applied Mathematics

MOVEMENT PROBLEMS FOR 2-DIMENSIONAL LINKAGES*

JOHN HOPCROFT†, DEBORAH JOSEPH‡ AND SUE WHITESIDES§

Abstract. This paper is motivated by questions concerning the planning of motion in robotics. In particular, it is concerned with the motion of planar linkages from the complexity point of view. There are two main results. First, a planar linkage can be constrained to stay inside a bounded region whose boundary consists of straight lines by the addition of a polynomial number of new links. Second, the question of whether a planar linkage in some initial configuration can be moved so that a designated joint reaches a given point in the plane is PSPACE-hard.

Key words. robotics, manipulators, mechanical arms, algorithms, polynomial time, PSPACE-hard

1. Introduction.

This paper is concerned with the motion of linkages from the computational complexity point of view. The research was motivated by earlier work in robotics, particularly that of Lozano-Perez and Wesley [LW-79], Lozano-Perez [L-80], Reif [R-79] and Schwartz and Sharir [S-81], [S-82]. There are two natural ways in which linkage movement problems arise in robotics. First, a linkage can model a robot arm. A frequently encountered model consists of a sequence of links connected together consecutively at movable joints. Second, linkages can also model hinged objects being moved by an arm or other type of manipulator. In both cases, it is essential to plan collision-avoiding paths of motion, as the manipulator and the object it is moving are generally required to lie within regions whose boundaries are determined by walls and the presence of other objects in the work space.

A *linkage* is a collection of rigid rods called *links* (see Fig. 1.1). The endpoints of various links are connected by joints, each joint connecting two or more links. The

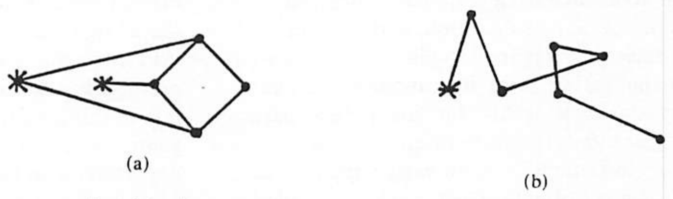

FIG. 1.1. (a) *A planar linkage.* (b) *An arm in the plane.*

links are free to rotate about the joints. In a *planar* linkage, links are allowed to cross over one another, and the linkage may be fastened to the plane so that the locations of certain joints are fixed (the fixed joints are indicated by * in the figures).

In a physical realization of a planar linkage, each link could move in a separate plane parallel to the ground. If links were joined together or to the ground by pins, then a link in one plane might collide with a pin joining links in two other planes. However, it is not difficult to design simple devices that function like pins but that do not interfere with the motions of the linkage. Thus the mathematical model in which

* Received by the editors November 19, 1982, and in revised form June 17, 1983. This research was supported in part by Office of Naval Research under contract N00014-76-C-0018, in part by the National Science Foundation under grant MCS81-01220, and the National Science Foundation under a Postdoctoral Fellowship and in part by a Dartmouth College Junior Faculty Fellowship.

† Department of Computer Science, Cornell University, Ithaca, New York 14853.
‡ Department of Computer Science, University of Wisconsin, Madison, Wisconsin 53706.
§ School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6.

THE IMPACT OF ROBOTICS ON COMPUTER SCIENCE

The rapid development of robotics and the resulting need for computer scientists to be better trained in traditional mathematics necessitate changes in computer science curricula.

JOHN E. HOPCROFT

Computer science began as a discipline about 1965. Its first 20 years were introspective, as it developed areas internal to the science itself. The typical curriculum at a university included data structures, algorithms, the design of programming languages, environments, operating systems, theory of computation, and the like. Having established itself as a science, it is now blossoming and reaching out into application areas including graphics, animation, electronic prototyping, simulation, and robotics. Computer science is destined to play a major role in numerous application areas. It is important to examine the consequences of this changing nature, particularly in regard to curriculum in universities.

Since Marvin Denicoff was an early supporter of robotics, it seems fitting to select robotics as a vehicle for exploring the likely impact of applications on computer science. Research in robotics today consists primarily of work in instrumentation, sensors, control theory, and robot design. These are important engineering problems, and research on them will lead to improvements in the cost of manufacturing and in the quality of finished products. However, the improvements, although significant, will not result in a marked change in our way of life.

There is a more general view of robotics as the study of representing, manipulating, and reasoning about physical objects in a computer [3]. This view is concerned with issues of languages and representation, and it is precisely in these areas, which traditionally come under the auspices of computer science, that significant breakthroughs will occur that will have as fundamental an effect on society as did the industrial revolution. Not only will computer science play a major role in these areas of robotics, but robotics and other application areas will have a major impact on computer science. If indeed computer science departments branch out into such areas, then it is important to examine what the effect of this will be on computer science.

Perhaps the most significant change will be in the prerequisite mathematics required of students, and it is in this area that I will focus. During the past 20 years, computer scientists have urged mathematics departments to teach more discrete mathematics. The needs of computer science have been primarily in logic, graph theory, combinatorics, and number theory—areas dependent on discrete mathematics. The robotics field, however, suggests that future needs will include much more traditional branches of mathematics. In this article, I will illustrate what I view as the computer science side of robotics and the mathematics that will need to be incorporated into the computer science curriculum in order to support research in this area.

Computer science has traditionally dealt with abstractions. The ability to represent abstract concepts symbolically promises to substantially modify previously established ways of thinking about design and assemblies. Traditionally, the engineer has thought of objects as being determined by their size and shape. However, intrinsic to an object being designed is its functionality and not its size or shape. Consequently, the potential to represent objects by

This article is based in part on research supported by the National Science Foundation under grants ECS 83-12096 and DCR 8502568.

© 1986 ACM 0001-0782/86/0600-0486 75¢

approach one another but do not meet, constituting choice points for further discrimination.) What is not clear is how "it should be possible to write an acceptable and fast algorithm for this" (as the authors suggest). One may be forgiven for suspecting some wishful thinking is occurring here. What is still required here is insight and human ingenuity in fitting trees together; ironically enough, in the theory of machine learning that is a limitation. Likewise, the PROLOG program (which the authors report they wrote) is restricted in that "no attention was paid to the heuristic part of constructing the trees so far" (p.297). So, once again, the intervention of human originality is indispensable.

Nevertheless, the authors are to be commended for engaging the issue of automating learning and for contributing an algorithm for reorganizing the program's conceptual hierarchy. Even if the essay is not a breakthrough, it represents "incremental progress" in a breakthrough area, and that is what normal scientific research is all about. A formal demonstration is produced, resembling a proof in model theory, that a solution to the problem of finding the consensus tree exists. There is a model, so the object is possible according to it. This result may in itself open up further avenues of investigation for the authors or others.

The paper is to be recommended to those interested in machine learning and logic programming. The authors should be encouraged to further engage the task of defining the problem of how to deal with the "grey areas" highlighted in reorganizing the conceptual description space of focusing.

—*Louis Agosta*, Chicago, IL

REFERENCES

[1] SIMON, H. A. *The sciences of the artificial*, MIT Press, Cambridge, MA, 1969. See *CR* 11, 1(Jan. 1970), Rev. 18,222.

GENERAL TERMS: ALGORITHMS, DESIGN, EXPERIMENTATION, THEORY

Plan execution, formation, generation
See: 8611-1038 [I.2.1—*SOJA*]

I. 2.9 Robotics
See also: 8611-1018 [F.2.2—*Geometrical problems and computations*]

HOPCROFT, JOHN E. (Cornell Univ., Ithaca, NY) 8611-1042
The impact of robotics on computer science.
Commun. ACM 29, 6 (June 1986), 486–498.

The main thrust of this paper is to point out that various problems arising in robotics require not only the discrete mathematics which are part of current computer science curricula, but also other traditional branches of mathematics, such as classical algebra, algebraic geometry, differential geometry, and algebraic topology, and that maybe these should be on the curricula too.

Indeed, the author identifies a general field concerned with representing and reasoning about physical objects, and calls it "stereo phenomenology." I personally see no need for yet more jargon, and feel that the more usual "solid modeling" or "geometric modeling" is adequate.

I cannot help feeling that the author makes his point in a somewhat heavy-handed manner: He takes an example from several particular areas of robotics or CADCAM, and shows how each mathematical subject he is interested in is of use. However, despite the use of both colored and black-and-white illustrations, he does so at a level which may not be accessible to those without some training in the areas he is talking about, which by implication includes many current computer scientists.

The list of references is somewhat unbalanced in that it does not cover all of the areas mentioned in the paper, and half of them refer to the author's own work. Nevertheless, I completely agree with the point the author is making in the paper, and I certainly hope that computer scientists on curriculum committees will give it the serious consideration it deserves.

—*Ralph Martin*, Cardiff, Wales

GENERAL TERMS: ALGORITHMS, DESIGN, PERFORMANCE, THEORY, VERIFICATION

Propelling mechanisms

RAIBERT, MARC H. (Carnegie-Mellon Univ., Pittsburgh, PA) 8611-1043
Legged robots.
Commun. ACM 29, 6 (June 1986), 499–514.

The paper provides an excellent overview of a research area of increasing interest. Motivations for research in the area of legged robots are clearly presented with a survey of early work in the area. Most of the paper is devoted to a description of the research performed by Marc Raibert at Carnegie-Mellon on hopping machines. Informative pictures are included to show the various generations of legged machines. In conclusion, the paper is informative and easy to follow, even for the reader new to this field. The references are more than adequate. Obviously, for the interested reader, the book published by the same author [1] is the best source of additional information.

—*M. Gini*, Minneapolis, MN

REFERENCES

[1] RAIBERT, M. H. *Legged robots that balance*, MIT Press, Cambridge, MA, 1986.

GENERAL TERMS: DESIGN, EXPERIMENTATION

I. 2.10 Vision and Scene Understanding
Intensity, color, photometry, and thresholding
See: 8611-1049 [I.4.6—*Region growing, partitioning*]

model, Koene's book can be highly recommended.

HORST W. HAMACHER
University of Florida

Numerical Recipes: The Art of Scientific Computing. *By William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling.* Cambridge University Press, Cambridge, UK, 1986. xx + 818 pp. $39.50. ISBN 0-521-30811-9; **Numerical Recipes: Example Book (FORTRAN).** viii + 179 pp. $18.95, paper. ISBN 0-521-31330-9; and **Numerical Recipes: Example Book (PASCAL).** x + 236 pp. $18.95, paper. ISBN 0-521-30956-5. Also available: Numerical Recipes FORTRAN Diskette V1.0, $19.95, ISBN 0-521-30958-1; Numerical Recipes Example Diskette (FORTRAN), $19.95, ISBN 0-521-30957-3; Numerical Recipes Pascal Diskette V1.0, $19.95, ISBN 0-521-30955-7; and Numerical Recipes Example Diskette (Pascal), $19.95, ISBN 0-521-30954-9.

"We call this book Numerical Recipes for several reasons. In one sense, this book is indeed a 'cookbook' on numerical computation. However, there is a distinction between a cookbook and a restaurant menu. The latter presents choices among complete dishes in each of which the individual flavors are blended and disguised. The former—and this book—reveals the individual ingredients and explains how they are prepared and combined." So begins the Preface of this masterpiece.

It is a compendium of the most useful scientific codes, given first in Fortran 77, and presented in seventeen chapters:

Chapter 1. Preliminaries. This chapter outlines how to organize programs; rounding and truncation errors; and stability. It has 3 sections, 4 programs, and 18 pages.

Chapter 2. Solution of linear algebraic equations. It has 12 sections, 10 programs, and 58 pages.

Chapter 3. Interpolation and Extrapolation. It has 7 sections, 13 programs, and 25 pages.

Chapter 4. Integration of functions. It has 7 sections, 13 programs, and 29 pages.

Chapter 5. Evaluation of functions. It has 9 sections, 9 programs, and 24 pages.

Chapter 6. Special functions. It has 8 sections, 30 programs, and 36 pages.

Chapter 7. Random Numbers. It has 7 sections, 15 programs, and 35 pages.

Chapter 8. Sorting. It has 6 sections, 11 programs, and 14 pages.

Chapter 9. Root finding and nonlinear sets of equations. It has 7 sections, 13 programs, and 34 pages.

Chapter 10. Minimization or maximization of functions. It has 10 sections, 16 programs, and 61 pages.

Chapter 11. Eigensystems. It has 8 sections, 7 programs, and 46 pages.

Chapter 12. Fourier transform spectral methods. It has 12 sections, 13 programs, and 73 pages.

Chapter 13. Statistical description of data. It has 10 sections, 21 programs, and 44 pages.

Chapter 14. Modeling of data. This chapter is on fitting data by formulas. It has 7 sections, 12 programs, and 49 pages.

Chapter 15. Integration of ordinary differential equations. It has 7 sections, 8 programs, and 31 pages.

Chapter 16. Two point boundary values problems. It has 7 sections, 8 programs, and 37 pages.

Chapter 17. Partial differential equations. It has 7 sections, 2 programs, and 53 pages.

Each of the sections concludes with references for further reading. The main text is then followed by a five-page list of references. Next, the authors present a 118-page chapter called Numerical Recipes in Pascal. This has an Introduction explaining the differences between FORTRAN 77 and PASCAL and then lists, section by section, a PASCAL version of each of the previous FORTRAN programs. A five-page alphabetical listing is then given of the dependencies among the programs. An 82-page index concludes this monumental work.

These four authors have demonstrated that too many cooks do *not* spoil the book! They have devised an attractive format that uses a smaller but heavier type for the programs. They provide a cogent, well-motivated, and complete treatment of each of the topics. It is possible to open the book at any section and learn without the necessity of starting at the beginning of Chapter 1. In fact, although it was not their original purpose, this book could serve as the text for a well-rounded numerical analysis course in the senior undergraduate or first graduate year.

I find that the authors achieved their goal of presenting and explaining the most useful, carefully documented, scientific programs. They have prepared diskettes that contain all of the programs in several machine readable formats. In particular, PC format is available. To aid the user, they have also prepared example booklets along with example diskettes. We have used the diskettes and run some programs on a PC/AT. In order to avoid the difficulty of preparing separate codes for the many different computing environments, the authors explain that they have not included interactive or graphics codes (except for one rough plotting routine). We highly recommend the book and the refreshing spirit that it brings to the reader and computer user.

EUGENE ISAACSON
*Courant Institute
of Mathematical Sciences
New York University*

SIAM REVIEW 1988