APIC'95, El Paso, Extended Abstracts, A Supplement to the international journal of Reliable Computing

Interval Methods in Mobile Robot Control

Kung Chris Wu

***Abstract*— Interval methods are used in the design of a cheap and efficient mobile robot.**

I. Formulation of a Problem

A. Making a robot move is important

The majority of working robots of today are stationary. They are automating the tasks that do not require lots of moving. It would be nice to automate tasks that do include moving. How to make a moving robot? If we want a robot to perform more or less routine operations on a factory floor (e.g., move materials), then we can wire this floor and use these guiding wires to control the robot. For more complicated tasks, especially if they include moving in an unknown environment, we must equip a robot with the ability to make complicated decisions.

B. A cheap robot is necessary

Contemporary state-of-the-art autonomous mobile robots (AMRs) use complex calculational and reasoning schemes for autonomous navigation in alien environments (see, e.g., [1, 2, 4]). Though powerful and versatile, those AMRs are extremely costly and computationally intensive. This makes them inaccessible to many cost-sensitive industrial applications.

In navigating a robot, the main computational power is used to solve the navigation problem. Crudely speaking, this problem is formulated as a mathematical optimization problem, and then solved by using methods of optimal control theory.

The main idea behind intelligent control is as follows: why should we repeat these calculations "from scratch" every time if we can use the experience of experts who are successfully controlling the motion in an unknown environment? So, instead of solving an optimization problem, we ask the expert drivers to formulate these rules, and apply fuzzy control methodology to transform these informal rules into a precise control strategy.

This idea has been successfully implemented (see, e.g., [8, 7, 6]). As a result, we get a small and relatively cheap autonomous robot that is almost as good in navigating as a human driver whose experience it simulates.

C. How our robot works

One of the main limitations of the human driver is that he can control at best one or two control parameters. Therefore, e.g., in a car, although there are many parameters that can potentially be controlled, we have to restrict their set to two: the speed (controlled by accelerator), and the angle (controlled by the steering wheel). Experienced drivers know that in difficult road conditions, it is desirable to control, e.g., each wheel separately, but there is no direct way to do that.

When we make a robot, we are no more bounded by this limitation. An on-board processor can control several independent parameters at a time almost as easily as just one parameter. This makes the motion of a AMR much better. E.g., if a human driver needs to turn a car around, he is limited to steering wheel turns, so his turn requires a lot of space. AMR is not going to operate on a highway system, so this space may not always be available, in which case an AMR that simulates human driving too closely will simply get stuck. The natural way to overcome this difficulty is to have separate controllers on two wheels. It is actually sufficient to have two controllers that control two engines attached to diagonally opposite wheels. By moving these wheels straightforward, we get a straight-line motion. By moving them clock-wise, we can make this robot turn around on the spot, without wasting any space at all.

This idea has been implemented in our robot [7], and we have found out that it successfully navigates through different environments (including a specially designed complicated maze).

The author is with the Department of Mechanical and Industrial Engineering, The University of Texas at El Paso, El Paso, TX 79968. This research was partially funded by the University of Texas at El Paso under the University Research Institute Grants 14-5077-6850.

D. The problem

The algorithms that control our robot come from intelligent control (see, e.g., [3]). They originate from the rules formulate by the experts in terms of natural language, like "if the deviation is large, then ..."; "if the deviation is small, then do ...", etc. In order to formalize these rules, we must describe what "large", "small", etc., mean. If we know that, e.g., possible deviations from the desired orientation are in the interval $[-5^o, 5^o]$, then we can identify deviations ≈ 5 degrees as large, those that are $\ll 5$ as small, etc., and get a reasonable control strategy.

However, we want our robot to operate successfully in *different* environments. Now. suppose that the environment has become smoother, so that the deviations are only from -1^o to 1^o. In this new environment, it makes no big sense to still use the rules that were designed for the entire interval $[-5, 5]$. It is necessary to *scale down* the control, so that the basic interval will now be equal to $[-1, 1]$. Vice versa, if the robot suddenly moves into a rougher terrain, we must *scale up* the domain of our control, because we can now encounter the values that are much larger than 5 degrees.

In short:

- if we see that all the measured deviations are within a small part of the original interval, then we must choose a new, smaller interval domain;
- if we get deviations outside the original interval domain, then we must choose a new, larger interval domain.

The question is: *how to scale intervals?*

II. Main Idea

We want to design a function f that transforms an interval $\mathbf{a} = [a^-, a^+]$ that represent the original interval domain into an interval $f(\mathbf{a})$ that represents a new interval domain.

The values $a^\pm$ are obtained from measurements. The numerical values depend on what starting point we used, and on what unit we have chosen. For example, assume that we are measuring time, and as a result we get 35 ± 5 (i.e., a set $\mathbf{a} = [30, 40]$). If we now change the starting point for measuring time, e.g., take -5 as the new starting point, then the same result will be expressed as $\mathbf{a} + 5 = 40 \pm 5 = [35, 45]$. It is natural to demand that the transformation f should not depend on the choice of the starting point. This means the following:

- If we use the original starting point, then the original interval domain is described by the interval $\mathbf{a} = [a^-, a^+]$. If we apply the function f to this domain, we get $f(\mathbf{a})$.
- If we choose a starting point that is a units before the original one, then the same original interval domain will be described by new numerical values: $\mathbf{a} + a = [a^- + a, a^+ + a]$. If we apply the same transformation to these new values, we get the interval $f(\mathbf{a} + a)$.

We want these two intervals to coincide:

- the interval $f(\mathbf{a})$ in the old units, and
- the interval $f(\mathbf{a} + a)$ in the new units.

To compare these intervals, we must transform both intervals to one and the same starting point. For example, if we transform $f(\mathbf{a})$ to new units, we will get $f(\mathbf{a}) + a$. This new interval must coincide with $f(\mathbf{a} + a)$:

$$f(\mathbf{a} + a) = f(\mathbf{a}) + a. \tag{1}$$

A similar condition can be formulated if formalize the idea that the transformation f should not depend on the choice of a unit. If, instead of the original unit (e.g., sec.), we use a new unit that is λ times smaller (e.g., millisec.), then the numerical values will increase λ times, and instead of $\mathbf{a} = [a^-, a^+]$, we will have $\lambda \cdot \mathbf{a} = [\lambda \cdot a^-, \lambda \cdot a^+]$. In this case, a similar invariance condition can be expressed as

$$f(\lambda \cdot \mathbf{a}) = \lambda \cdot f(\mathbf{a}). \tag{2}$$

Now, we are ready for the formal definition:

III. Definition and the Main Results

Definition 1.

- We say that a function f from intervals to intervals is *translation-invariant* if it satisfies condition (1) for every interval $\mathbf{a}$ and for every real number a.
- We say that a function f from intervals to intervals is *scale-invariant* if it satisfies condition (2) for every interval $\mathbf{a}$ and for every positive real number $\lambda > 0$.

THEOREM. *Every translation-invariant and scale-invariant function from intervals to intervals has the form*

$$f([a^-, a^+]) =$$

$$[\frac{a^- + a^+}{2} + \frac{a^+ - a^-}{2} \cdot b^-, \frac{a^- + a^+}{2} + \frac{a^+ - a^-}{2} \cdot b^+]$$

for some real numbers $b^\pm$.

Comments.

- This result was proven together with S. Starks and V. Kreinovich.
- For a different application of this result, see [5].
- In our case, we are interested in specific intervals, namely, intervals that describe deviation. There intervals are usually symmetric with respect to 0, i.e., they have the form $[-\Delta, \Delta]$ for

some $\Delta > 0$. We would like the transformation intervals to be also of this type.

Definition 2.

- We say that an interval $\mathbf{a} = [a^-, a^+]$ is *symmetric* if $a^- = -a^+$.
- We say that a function f is *symmetry-preserving* if it transforms symmetric intervals into symmetric ones.

PROPOSITION. *If f is a translation-invariant, scale-invariant, and symmetry-preserving function from intervals to intervals, then there exists a real number $C > 0$ such that for every symmetric interval, $f([-a, a]) = [-C \cdot a, C \cdot a]$.*

Comment. So, natural invariance demands lead to a very simple rescaling: we just dilate or shrink the interval. This rescaling was experimentally compared with more complicated ones, and it turned out to perform better. In the actual implementation, we use $C = 1/2$ for shrinking, and $C = 2$ for expanding, because inside the processor, where all the numbers are represented in the binary form, multiplication and division by two are the easiest to implement: they are equivalent to shift by one bit.

IV. Proofs of the Theorem and the Proposition

Let us denote the endpoints of the interval $f([-1, 1])$ by $[b^-, b^+]$. Them, for an arbitrary $c > 0$, due to scale-invariance,

$$f([-c, c]) = cf([-1, 1]) = c \cdot [b^-, b^+] =$$

$$[c \cdot b^-, c \cdot b^+].$$

In particular, if f is symmetry-preserving, then $b^- = -b^+$, so we get the proof of the Proposition (with $C = b^+$).

In the general case, an arbitrary interval $[a^-, a^+]$ can be represented as $m + [-c, c]$, where $m = (a^- + a^+)/2$, and $c = (a^+ - a^-)/2$. Since f is dilation-invariant, we conclude that $f([a^-, a^+]) = m + f([-c, c]) = [m + c \cdot b^-, m + c \cdot b^+]$. Q.E.D.

References

[1] M. Daily et al, "Autonomous cross-country navigation with the ALV", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1988, pp. 377–389.

[2] Y. Goto and A. Stentz, "The CMU system for mobile robot navigation", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1987, pp. 99–105.

[3] A. Kandel and G. Langholtz (Editors), **Fuzzy Control Systems**, CRC Press, Boca Raton, FL, 1994.

[4] F. G. Pin, H. Watanabe, J. R. Simon, and R. S. Pattay, "Autonomous navigation of a mobile robot using custom-designed qualitative reasoning VLSI chips and boards", *Proceedings of the IEEE International Conference on Robotics and Automation*, 1992, pp. 123–128.

[5] S. Starks, "An interval approach to congestion control in computer networks", *These Proceedings*, 1994.

[6] K. Chris Wu, "A robot must be better than a human driver: an application of fuzzy intervals". In: L. Hall, H. Ying, R. Langari, and J. Yen (eds.), *NAFIPS/IFIS/NASA'94, Proceedings of the First International Joint Conference of The North American Fuzzy Information Processing Society Biannual Conference, The Industrial Fuzzy Control and Intelligent Systems Conference, and The NASA Joint Technology Workshop on Neural Networks and Fuzzy Logic, San Antonio, December 18–21, 1994*, IEEE, Piscataway, NJ, pp. 171–174.

[7] K. C. Wu, "Fuzzy logic control of a robot mouse", *Proceedings of the 5th International Symposium on Robotics and Applied Manufacturing*, Maui, Hawaii, August 1994.

[8] K. C. Wu, R. K. Joseph, and N. K. Thupili, "Evaluation of classical and fuzzy logic controllers for wind turbine yaw control", *Proceedings of the 1993 IEEE Regional Conference on Aerospace Control Systems*, Rockwell Science Center, Thousand Oaks, CA, May 25–27, 1993.