AEROSPACE APPLICATIONS OF SOFT COMPUTING AND INTERVAL COMPUTATIONS (WITH AN EMPHASIS ON SIMULATION AND MODELING)

SCOTT A. STARKS and VLADIK KREINOVICH
NASA Pan-American Center for Earth and Environmental Studies
University of Texas at El Paso
El Paso, TX 79968, USA
emails sstarks@utep.edu, vladik@cs.utep.edu

ABSTRACT

This paper presents a brief overview of our research in applications of soft computing and interval computations to aerospace problems, with a special emphasis on simulation and modeling.

KEYWORDS: Soft computing, interval computations, aerospace applications

INTRODUCTION: DATA PROCESSING AND INTERVAL COMPUTATIONS

Data processing

In many real-life problems, we are interested in the value y of a physical quantity which is *difficult* or *impossible* to measure directly. For example, we cannot directly measure the distance to a star, or the amount of oil in a given area. To measure this quantity y, we:

- measure some other quantities $x_1, \ldots, x_n$ which are related to y by a known dependence $y = f(x_1, \ldots, x_n)$, and then
- compute the estimate $\widetilde{y}$ for the desired quantity y by applying the algorithm f to the results $\widetilde{x}_i$ of measuring the quantities x_i: $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

This two-stage process is called *indirect measurement*, and computing f is called *data processing*.

For example, to estimate the amount of oil in a given area, we may use geophysical data plus satellite images of this area.

Error estimation of the results of data processing: mathematical statistics and interval computations

Values $\widetilde{x}_i$ come from measurements, and measurements are never 100% accurate; therefore, $\widetilde{x}_i \neq x_i$. Due to the inaccuracies $\Delta x_i = \widetilde{x}_i - x_i$ of direct measurements, the result $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is, in general, different from the desired value $y = f(x_1, \ldots, x_n)$: $\Delta y = \widetilde{y} - y \neq 0$. In practical applications, it is extremely important to know what are the possible values of the difference Δy.

For example, if our estimate for amount of oil in a given area is $\approx$100 mln. ton, then whether we start exploiting this oil or not depends on the accuracy of this estimate:

- If the measurement error Δy does not exceed 10 mln. ton, then the actual value can be anywhere from 90 to 100, and we should recommend exploitation.
- On the other hand, if the measurement error Δy can be as large as 100 mln. ton, then this means that the actual value y can actually be equal to 0 (meaning that there may be no oil at all). In this case, further, more accurate measurements are needed because we can make a decision.

To estimate Δy, we must have some information about the errors Δx_i of direct measurements. What type of information can we have?

- The manufacturer of the measuring instrument gives us a *guaranteed* error Δ_i, i.e., a value for which $|\Delta x_i| \leq \Delta_i$. (Without such a guarantee, a measurement result does not restrict possible values of x_i and thus, it is not a measurement.)
- In some cases, in addition to the upper bounds Δ_i, we know *probabilities* of different values of Δx_i.

If we know probabilities, then we have a typical problem of *mathematical statistics*: given probability distributions for $\Delta x_i = \widetilde{x}_i - \Delta x_i$, find the probability distribution for $y = f(x_1, \ldots, x_n)$. To get the probabilities of Δx_i, we *calibrate* the measuring instrument, i.e., we compare its results with the results of a better (standard) measuring instrument. An application of statistical methods to environmentally-oriented multi-spectral satellite image processing is given in [23].

However, there are two important situations when we do not know these probabilities:

- In *fundamental physics*, we perform measurements on the *cutting edge*, so no better instrument is possible at all.
- In *manufacturing*, calibration of all sensors is potentially possible, but, in practice, too expensive.

When we do not know the probabilities, we only know that $|\widetilde{x}_i - x_i| \leq \Delta_i$, i.e., the only information about x_i is that x_i belongs to the *interval* $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$. For example, if the measured value of the current is $\widetilde{x} = 1$ A, and the manufacturer

guarantees the measurement error to be within ± 0.1 A, then the actual value of x can be any number from the interval $[0.9, 1.1]$.

In this case, the problem of estimating the error of indirect measurement can be reformulated as follows:

- we know n intervals $\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$,
- we know an algorithm f which transforms n real numbers $x_1, \ldots, x_n$ into a real number y, and
- we want to compute the interval

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \{f(x_1, \ldots, x_n) \,|\, x_i \in \mathbf{x}_i\}.$$

This problem is called the basic problem of *interval computations*.

Linearization is not always possible

If a function f is smooth, and the errors Δx_i are small, then we can neglect quadratic terms in f, and get explicit formulas for $\mathbf{y}$. Due to our approximation, we get *approximate* endpoints of the interval $\mathbf{y}$: the actual values y can be, therefore, slightly outside this approximate interval.

In many applications, it is OK, but in some real-life situations, the consequences of a possible error are so serious that we need to *guarantee* that y is contained in the resulting interval $\mathbf{y}$. An example of this problem is planning a mission to the Moon. To get guaranteed estimates for this problem, Ramon E. Moore, then Stanford's Ph.D. student working on 1959 NASA-oriented project, designed new techniques called *interval computations*.

INTERVAL COMPUTATIONS IN AEROSPACE APPLICATIONS: WHY

Let us enumerate the reasons why methods of interval computations are needed in aerospace applications:

- First, we want to *guarantee* a mission, we want to *guarantee* that a spaceship hits the Moon (or another planet), and interval computations provide us with the *guaranteed* computation results.
- Second, according to the new NASA paradigm, we need all the missions to be *faster, better, cheaper*. This means, in particular, that we should preferably use off-shelf components, with no time to individually calibrate all of them (and thus, no time to find all the probabilities).
- Third, many NASA missions are missions into the unknown. We simply do not know the exact values of the parameters characterizing the distant planet's surface, or the corresponding probabilities; the only thing we may know for planning a mission are *intervals* of possible values of these parameters.

- Finally, one of the main goals of NASA missions is to produce *solid scientific results*, and "solid" means *guaranteed*.

AEROSPACE APPLICATIONS OF INTERVAL COMPUTATIONS: EXAMPLES

Robot navigation

A mobile robot has to navigate in an unknown environment by using imprecise sensors. Traditionally, statistical approach was used to describe the sensor's uncertainty, but this approach has two main drawbacks: it is very costly to calibrate, and it cannot be applied in an unknown environment, when we have no time to calibrate first. To avoid these problems, we used *interval uncertainty* in a UTEP robot. This robot won 1st place in the international competition at AAAI'97: it was more efficient, less error-prone, and at the same time rather simple to program. This technique can be used in future planetary missions.

Uncertainty in Robot Navigation: Interval Methods Are Needed. Robots have to deal with two types of uncertainty:

- first, their *sensors* are not absolutely accurate; as a result, they measure, e.g., distances to obstacles only approximately;
- second, their *actuators* are not absolutely precise; as a result, e.g., a command to turn 90 degrees can actually leads to an 85 or 95 degree turn.

Traditionally, *statistical* methods have been used to deal with these two types of uncertainty; see, e.g., [3] and references therein. There are, however, two major problems related to these methods:

- First, statistical methods are very computationally intensive. For every pixel, at any moment of time, we need to compute and store the probability that the corresponding point contains an obstacle. In a mobile robot, it is desirable to have computational methods that are as simple as possible.
- Second, even more importantly, these methods require that we know the probabilities of errors for different sensors and actuators, and we usually do not know the exact values of these probabilities. Instead, we only know the *intervals* of possible error values.

We can try to guesstimate the probabilities, but:

- if we wrongly guess the probabilities of *sensor* errors, we may erroneously hit an obstacle;
- if we wrongly guess the probabilities of *actuator* errors, and use these wrong probabilities in some filtering-type correction, we may worsen the position error instead of compensating for it.

It is therefore reasonable not to guesstimate probabilities, but to use interval-based methods instead.

These methods were successfully implemented in the robot *Diablo* designed by the University of Texas at El Paso team in 1996. This team was supervised by Dr. Chitta Baral, participants were David Morales, Tran Cao Son, Luis Floriano and Monica Nogueira (main team) and Alfredo Gabaldon, Glen Hutton, Dara Morgenstern, and Richard Watson (support team).

Before we describe the interval methods that we used, let us briefly describe our robot and the competition in which our robot participated.

Our Robot. Diablo is a B14 mobile robot manufactured by Real World Interface. It is equipped with a PC (with a 133 MHz Pentium processor running Linux OS), a speech synthesizer, 16 sonars, 16 infrared sonars, and four tactile sensors.

Communication with Diablo was achieved via a Compaq 486DX2 25 MHz laptop through a direct serial port connection.

Robot Competition. Event One of the Robot Competition, *Organize a Meeting*, required the competitors to navigate in an office-like partitioned environment consisting of corridors, a foyer, and multiple rooms. The approximate map of the environment was made known to the teams only at the beginning of the competition.

The goal was to organize a meeting between a director and two professors. The environment included several rooms that can be, in principle, used for such meetings. Some of these rooms were already occupied. The robot is, initially, at the director's office. From there, the robot had to visit the possible conference rooms. If one of these rooms was available, then this room was allocated as a meeting place; otherwise, the meeting was supposed to be held in the director's office. After allocating the room, the robot had to calculate the earliest possible meeting date (based on the robot's knowledge of distances). Then the robot had to inform all three participants about the time and location of the meeting: first, he had to visit the two professors in their offices (in any order), and then, come back to the director's office and inform the director.

In the competition, points were taken off for not fulfilling the task correctly and for hitting the walls or other obstacles. Robots that achieved the task in shorter time got extra bonus points.

The detailed description of this competition is given, e.g., in [7] (see also [15]).

Interval Methods Used In Our Robot: How to Avoid Hitting Obstacles. One of the main goals of the robot is to avoid hitting obstacles. Hence, the actual distance D to the obstacle should be always positive.

The robot constantly measures this distance. Due to measurement errors, the measured distance $\widetilde{D}$ is, in general, different from the actual distance D. Thus, even if $\widetilde{D} > 0$, it is quite possible that $D = 0$, and the robot is going to hit the obstacle.

From the documentation supplied with the robot, we got the upper estimate Δ for the measurement error $|\widetilde{D} - D|$. Therefore, if the measurement result is $\widetilde{D}$, then the only information that the robot has about the actual D is that it belongs to the interval $\mathbf{D} = [\widetilde{D} - \Delta, \widetilde{D} + \Delta]$. Hence, the only way for the robot to be sure that it

is not going to hit the obstacle in the next moment of time (i.e., that $D \neq 0$) is to guarantee that 0 does not belong to the interval $\mathbf{D}$ of possible values of D, i.e., that $\widetilde{D} > \Delta$.

Interval Methods Used In Our Robot: How to Plan a Route. The only information the robot has about the distance to the object is the measured value $\widetilde{D}$. When this measured distance is greater than Δ, the robot is safe. If this measured distance starts getting close to Δ, the robot must try to get away from the obstacle or from the wall.

So, if we want a robot to travel safely, we must thus plan a route along which the measured distance $\widetilde{D}$ to the nearest obstacle would always be greater than Δ.

For any location of a robot in which the actual distance is D, the measured value $\widetilde{D}$ can take any value from $D - \Delta$ to $D + \Delta$. Hence, to guarantee that $\widetilde{D} > \Delta$, we must choose a route along which each possible value of $\widetilde{D}$ from this interval is greater than Δ. This is equivalent to requiring that the smallest possible measured value $D - \Delta$ from this interval is greater than Δ: $D - \Delta > \Delta$, and $D > 2 \cdot \Delta$.

Thus, we must plan a route along which $D > 2\Delta$. Hence, in our planning algorithm, when, e.g., a robot was going in a corridor, we planned its path so it would be within the "virtual corridor" formed by all the points whose distances from all the walls exceed 2Δ.

Actuator Errors. In addition to sensors, actuators also have uncertainty, which was described by intervals. When we gave a robot a command, e.g., to turn 90 degrees to the right, we took into consideration that in reality it could turn any angle from $90 - \delta$ to $90 + \delta$, where δ is the upper bound on the actuator error.

To compensate for the deviations from the desired path caused by the actuator errors, we periodically adjusted the robot's orientation.

Interval Methods in Failure Detection and Recovery. Sensors are not only *inaccurate*, they sometimes *miss* the signal.

So, a robot can miss the door or a corridor. To resolve this problem, the organizers gave the approximate map of the office (without the obstacles, of course), and robot could use this map in their navigation.

If the map is precise and the sensors are precise then, when we have passed a certain point and did not see the door, this means that we have missed it, so we need to turn back and try again. However, if we try to use this idea directly and do not take sensor errors into consideration it may happen that the sensors show that we have already missed it, while in reality, the inaccurate sensors overestimate the path and we are still approaching the door. If the robot turns back at this point and tries to find the door again and again, it may go into a time-wasting loop.

To avoid that, we made the robot "panic" and turn back only when it was *guaranteed* that the robot did miss the door. The only way to guarantee that is to take into consideration the interval-bounded errors of the sensors and the inaccuracy of the map. Due to these errors and inaccuracies:

- Instead of the actual location $\vec{l}_{\text{robot}}$ of the robot, we only know an *area* $\mathbf{l}_{\text{robot}}$ of

possible locations (e.g., a circle of radius Δ).

- Instead of the actual location $\vec{l}_{\text{door}}$ of the door, we only know an *area* $\mathbf{l}_{\text{door}}$ of possible locations.

The robot only turns back if we are guaranteed that it had missed the door, i.e., every point in the area $\mathbf{l}_{\text{robot}}$ is "after" every point from $\mathbf{l}_{\text{door}}$.

Several Other Novel Ideas Have Been Used. In addition to the interval methods, the robot also used several ideas from Artificial Intelligence both in its planning algorithm and in its reactive control methods [1,14,15].

The Result.

- In comparison with statistical-based robots, Diablo proved to be:
  - computationally simpler and
  - 100% reliable, always staying on track and never hitting any obstacle;
  - also, due to the interval methods used in failure detection and recovery, our robot never turned back prematurely.
- On the other hand, since *Diablo* used the most cautious approach, it was somewhat slower than the two winning robots, that moved closer to the obstacles (thus risking a hit) to fulfill the task faster. Also, due to the over-cautious interval methods of failure detection and recovery, our robot went extra few inches after every sensor failure, on which it also lost some time.

As a result, our robot won the third place in 1996 and the first place in 1997, outperforming more than 20 much more technologically sophisticated robots from all over the world, including teams from prestigious institutions long involved in world-class robotic research such as Carnegie-Mellon University and the Universities of Stuttgart and Bonn.

Follow-Up Research: Detecting Whether a Robot is Stuck. Originally, the robot was planned to move in an environment, where it was assumed that the corridors were not completely blocked, and that therefore, a robot was always able to navigate towards its goal locations.

In real-life world, it is quite possible that the paths will be temporarily or permanently blocked. If the paths are blocked, then it is desirable not to waste the precious battery energy for futile repeated attempts to find a way, it is better to think up a new path or simply to wait until the path clears. The problem is how to check whether the robot is stuck (especially if the motor is running, but the robot is not moving anywhere).

The "Stuck" Problem Is Easy To Solve If Sensors Are 100% Accurate. If all the sensors were precise, then "stuck" would mean that the sensor's reading in the current moment of time would be identical to its reading in the previous moment of time.

For Real-Life Sensors, Interval Methods Help. Real-life sensors (sonars and infrared) are rather inaccurate, so the actual reading may change even if the robot stalls. To take this inaccuracy into consideration, the robot considers, instead of the actual sensor readings $\tilde{x}_i$ that may differ from the actual values, the *intervals* $\mathbf{x}_i = [\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$ of possible actual values.

The robot will then consider itself *stalled* if for each sensor i, the intervals that correspond to two sequential moments of time t_k and t_{k+1} have a non-zero intersection $(\mathbf{x}_i(t_k) \cap \mathbf{x}_i(t_{k+1}) \neq \emptyset)$.

Future Plans: Interval Optimization. In the present robot, the optimal planning is performed without taking interval uncertainties into consideration:

- First, we used numerical optimization.
- Then, the plans are modified to accommodate for the interval uncertainty.

It is desirable to take the interval into consideration from the very beginning, and to use interval-based optimization techniques both:

- for planning the initial trajectory and
- for planning the (optimal) trajectory corrections that are necessary, e.g., when an unexpected obstacle is encountered.

Future Plans: Interval Tools. At different points of the design, interval-related ideas were used.

Our first attempt was (almost always) to neglect the interval errors and to make the algorithms as simple as possible. Only when it turned out that these errors are essential, they were incorporated in the algorithms. This incorporation was done "by hand", without using any systematic interval computations tool.

It is definitely desirable to use interval software tools that simplify the transition from numerical to interval computations. The use of such tools would decrease the amount of time spent on programming and design.

Such tools do not directly affect the robot performance, but they can affect it indirectly, because the time that we spent on taking interval ideas into consideration was taken from the overall contest time, and could be spend on some further optimization and fine-tuning.

Future Plans: Interval Simulation. To find the best algorithms for our robot, we ran many tests and experiments, and most of the experiments were done on the actual robot. Comparing algorithms on the actual robot is a very slow process.

It is, therefore, desirable to build a simulator that would try different algorithms on the simulator before going to the actual robot. To really test the algorithms, we want this simulator to take the interval uncertainty into consideration, and in every situation, to test the robot in the hardest possible situation compatible with the measurement results.

Interval global optimization methods can also enables us to get interval estimates of the parameters of the model that best describe the robot's behavior[1]. Thus, there are three potential uses of intervals:

- in *model estimation*;
- in *optimizing* the robot's algorithm using the interval model of a robot and of its environment;
- in the actual *operational algorithm* that the robot will follow.

Future Plans: Interval Fuzzy methods. In some cases, in addition to the intervals, we have some expert information about the robot and about the environment. This expert information is usually taken into consideration by using methods of expert systems, fuzzy logic, and intelligent control; see, e.g., [21]. It is desirable to combine interval and fuzzy knowledge in controlling the robot.

For the simplest mobile robots, this combination was already implemented by Chris Wu [25,26], and it did result in the improvement of robot's performance.

Telemanipulation [9]

The idea of telemanipulation, when a robotic arm repeats the movements of the operator's arm, works perfectly well in the movies, but not so perfectly well in the real space exploration. The reasons for this imperfection are simple: both sensors (which measure the operator's movements) and the actuators (which copy them) are inaccurate. The more complicated the robotic arm, the more actuators it uses, and the more inaccuracy accumulates. It turns out that if we take interval inaccuracy into consideration, we can greatly improve the performance of the telemanipulator – namely, of the state-of-the-art MIT/Utah robotic arm.

Telemanipulators Are Needed. In many real-life mechanical tasks, it is difficult or even impossible to use humans:

- Some environments are too dangerous for a human being: for example, when we manipulate objects in space, inside a radioactive part of a nuclear reactor, in a dangerous chemical environment, or even in a potentially dangerous environment such as handling viruses that cause deadly diseases.
- In other environments, there is no danger to the human operator, but there is a significant risk of contamination of the object: e.g., in handling microchips, lunar samples, etc.

In all these cases, reasonably simple mechanical tasks can be done by an automatic mechanical hand-arm. However, there is a limit on the complexity of the tasks that automatic devices can do. For more complicated tasks, for which we cannot use a

[1] We are thankful to the anonymous referee for suggesting this idea.

completely automated system, we must use *telemanipulators*, i.e., devices in which a mechanical hand-arm copies the movements of a human operator.

Telemanipulators: Successes. The main goal of the telemanipulator is to reproduce the operator's movements as accurately as possible.

A human hand is a very flexible instrument. In mechanical terms, we can say that it has many degrees of freedom: we can move and rotate the hand itself, the arm, each finger, parts of each finger, etc. Thus, to reproduce its movements accurately, the manipulator also has to have many degrees of freedom.

At present, the best of widely available hand-arm manipulators, the Utah/MIT hand, has 22 degrees of freedom. It is still slightly less than a human hand, because, e.g., it only has 4 fingers and not 5. However, it can perform many important tasks that a human hand can do.

This hand was not designed for telemanipulation only. It has many other applications: e.g., it can even twist itself into the positions that would have been impossible for a human hand.

Telemanipulators: Problems. Both in the Hollywood movies and in the self-made movies that researchers show at robotic conferences, telemanipulation works perfectly well: a robotic hand exactly reproduces the operator's movements. This is indeed happening in many application areas, but this reproduction accuracy is extremely difficult to achieve.

If we simply measure the pressure, etc., applied by the operator's arm, and send exactly proportional control signal to the electric motors that control different degrees of freedom of the robotic arm, we get a behavior that is often drastically different from what the operator did. For example, the operator's firm grip on the object may be distorted into the robotic arm dropping it, and vice versa, the operator's tender approach to a fragile object may result in a robotic arm's bumping into the actual object and damaging it.

There are three main reasons for the difference between the movements of the human and robotic hands:

- first, the sensors that measure the human hand's pressure are not 100% accurate;
- second, the motors and actuators are not perfect, and do not react precisely to the commands;
- third, the mechanical characteristics of the robotic hand itself are somewhat different from the mechanical characteristics of the operator's hand.

As Manipulators Get More Complicated, These Problems Get More And More Important. The above inaccuracy problems can be traced even on the example of simple manipulators that have a few degrees of freedom, but for more advanced manipulators, these problems become more and more acute. Indeed, for a manipulator, more advanced means that this manipulator has more degrees of freedom. Each degree of freedom bring its own inaccuracy, so if we have 22 degrees of freedom, then in

principle, we get 22 sources of inaccuracy all leading to the huge inaccuracy of the resulting action.

Let us give a simple example.

- If we have a 3-finger manipulator, then for this manipulator to grip an object, it must place one finger below it, and two fingers above it. Due to inaccuracy, we may have a slightly distorted position, but we will still keep firmly 3 points on the object.

- For a 4-finger arm that is similar to the human arm, we need to place 3 fingers on top of the object. If, e.g., the upper surface is planar, we must have all 3 fingers on one line. Due to inaccuracy, one of these fingers may be higher than the others. As a result, this finger may not contact the object at all, and hence, the grip will not be as firm as we desired.

So inaccuracy is harmful. In order to figure out how to decrease this inaccuracy, let us first analyze how we can describe it in precise terms.

Describing Inaccuracy: Intervals. For sensors and other measuring instruments, manufacturers usually give a guaranteed upper bound on the measurement error.

> Indeed, if no such bound is supplied, this means that the manufacturer does not guarantee any accuracy. So, if we get some value from this instrument, the actual value of the measured quantity can arbitrarily differ from this "measured" value. In other words, no matter what we "measure" by this instrument, we can still have an arbitrary actual value. In other words, if no upper bound is given, this "measurement" does not give us any information about the actual value, so there is not reason to call it a measurement at all.

In addition to the upper bound, we sometimes know the *probabilities* of different values of measurement error.

To get such probabilities, we need a lot of experimental data, which we usually, for manipulators, do not have.

So, for manipulators, the typical information about the measurement error consists simply of knowing its guaranteed upper bound Δ. So, if the measured value of some quantity is $\tilde{x}$, this means that the actual value x of the measured quantity is within the interval $[\tilde{x} - \Delta, \tilde{x} + \Delta]$ (and we cannot a priori exclude the possibility of x being equal to any of the real numbers from this interval).

Similarly, the errors caused by actuators can also be described by intervals. As a result, at every moment of time, we have the following situation:

- the teleoperator applies the control values $x_1, \ldots, x_n$;

- due to measurement errors, the telemanipulator system measures the values $\tilde{x}_1, \ldots, \tilde{x}_n$ that are, in general, different from the corresponding values x_i: $\tilde{x}_i = x_i + \Delta x_i$, where $\Delta x_i \neq 0$; the only information that we have about Δx_i is that $|\Delta x_i| \leq \Delta_i$ for some manufacturer-supplied accuracies $\Delta_1, \ldots, \Delta_n$.

Main Idea. Although a human hand-arm has many degrees of freedom, we rarely use all of them in the same movement. Usually, the movement in different degrees of freedom is very much *coordinated*.

For example, if we have already firmly grasped an object, then we move the arm as a whole and, unless necessary, do not use the ability to move fingers and/or or fingertips separately.

There is a limited number of typical movements of this type, and a teleoperator can pretty well describe which of these typical movements he is applying at any given moment of time. When we get this information, we can use it to set up a similar coordination between the degrees of freedom of the robotic hand-arm. When the resulting constraints are in place, the originally flexible robotic hand acts as a new tool that is specifically designed for this type of movement. Since in reality, we are still using the same robotic hand, this is not a new *physical* tool, but a *virtual* tool.

We will see that using virtual tools can indeed be very helpful.

How to Describe Movement Type in Precise Terms. In precise terms, a fixed movement type means that we cannot have *arbitrary* values of $x_1, \ldots, x_n$: these values must satisfy one or several restrictions (constraints).

> For example, if we want the arm to move as a whole, then one of these constraints may take the form $x_1 = x_2$ (or $x_1 - x_2 = 0$), where x_1 and x_2 are pressures applied by two fingers. If we want to preserve the distance between the two fingertips, then we may require something like $(x_1 - x_2)^2 + (x_3 - x_4)^2 - \text{const} = 0$.

In general, we have one or several constraints of the type $F(x_1, \ldots, x_n) = 0$.

So, for the actual values $x_i = \tilde{x}_i - \Delta x_i$, in addition to the intervals $[\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$ of possible values, we have the additional constraints of the type

$$F(x_1, \ldots, x_n) = F(\tilde{x}_1 - \Delta x_1, \ldots, \tilde{x}_n - \Delta x_n) = 0.$$

Since inaccuracies Δx_i are small, we can expand the function F in Taylor series and ignore terms that are quadratic or of higher order in Δx_i. As a result, each original constraint $F_k(x_1, \ldots, x_n) = 0$ becomes a *linear* constraint on possible values of Δx_i:

$$F_{k1} \cdot \Delta x_1 + \ldots + F_{kn} \cdot \Delta x_n - F_k = 0, \tag{1}$$

where

$$F_{ki} = \frac{\partial F_k(x_1, \ldots, x_n)}{\partial x_i}_{|x_1 = \tilde{x}_1, \ldots, x_n = \tilde{x}_n}$$

and $F_k = F_k(\tilde{x}_1, \ldots, \tilde{x}_n)$.

How to Use the Movement Type to Decrease Uncertainty. If we know relations (1), then instead of the original intervals $[\tilde{x}_i - \Delta_i, \tilde{x}_i + \Delta_i]$, we may have *narrower* intervals

$$[\tilde{x}_i - \underline{\Delta}_i, \tilde{x}_i + \overline{\Delta}_i],$$

where $\underline{\Delta}_i$ is the solution to the minimization problem

$$\Delta x_i \to \min$$

under the conditions

$$-\Delta_j \le \Delta x_j \le \Delta_j,\ 1 \le j \le n;$$

$$F_{k1} \cdot \Delta x_1 + \ldots + F_{kn} \cdot \Delta x_n - F_k = 0,\ 1 \le k \le K,$$

and $\overline{\Delta}_i$ is the solution to the similar maximization problem

$$\Delta x_i \to \max$$

under the same constraints.

Both optimization problems are linear programming problems, and they can be easily solved by using the standard linear programming techniques.

Within these intervals, we select the control values that satisfy all required equalities.

A Simple Example Showing That Constraints Can Decrease Uncertainty. Suppose that we have a movement in which two fingers have to move in the exact same way, i.e., in which $x_1 = x_2$. Suppose that the actual movement was $x_1 = x_2 = 1$. Let us also suppose that the measurement accuracy is $\Delta_1 = \Delta_2 = \Delta_3 = 0.2$. Due to measurement errors, we get, e.g., the following two sensor readings: $\tilde{x}_1 = 1.1$, $\tilde{x}_2 = 0.9$.

If we do not take the constraint into consideration, then we get intervals $[0.9, 1.3]$ for x_1 and $[0.7, 1.1]$ for x_2. For both variable, it is possible to have movement reproduction errors as high as 0.3.

If we do take the constraint $x_1 = x_2$ into consideration, then, as one can easily see, the possible values of $x_1 = x_2$ lie in the *intersection* of the two intervals: $[0.9, 1.3] \cap [0.7, 1.1] = [0.9, 1.1]$. For values from this intersection, the largest possible reproduction error is 0.1. In other words, in this simple example, we have a 3 times decrease in reproduction error.

Implementation. Several virtual tools have been actually implemented for grasping and manipulation with the Utah/MIT hand. Ax a result, we do have an improved telemanipulation performance.

Multi-spectral satellite imaging [22,24]

The existing Earth-imaging satellites of Landsat series, whose ability is restricted to 7 channels only, already send Gigabytes of difficult-to-process information. For some imaging problems, 7 channels are not sufficient, so new satellites will be able to scan 500 channels. With 100 times more data, we need at least 100 times more time to process it; even now, processing all the satellite data is a problem, and with the expected two orders of magnitude increase, this processing seems to be getting close to impossible. Solution: take interval uncertainty into consideration. It turns out that with this uncertainty in mind, we can use *linear* models where previously only complex models were used; computations become *faster* and thus, quite feasible.

Non-destructive testing of aerospace constructions

Failure of an aerospace apparatus can be disastrous, and therefore, all mechanical parts must be thoroughly tested. Exhaustive testing, however, is extremely expensive. Here also intervals help. It turns out that:

- when the tested surface is smooth (no faults, no cracks, etc.), the dependence of the measured signal on the test ultrasound signal is also smooth, and since the test signals are small, we can approximate it by a linear dependence;
- on the other hand, if there are non-smoothnesses (faults, cracks, etc.), then non-linear terms are no longer negligible.

Checking whether the actual data is consistent with the linear dependence (within interval uncertainty), we can thus test whether there is a non-smoothness. Experiments confirmed that this is a viable and expense-saving testing method.

We also analyzed the problem of choosing the best sensor locations for aerospace testing [17–19].

Geophysical tomography [2]

Interval computations help in reconstructing the geophysical structure from observations.

Energy from space: a possible future application of interval computations

Solar energy is a very prospective renewable energy resource, but on-Earth Solar stations are not perfect: they occupy large pieces of land, they do not work in bad weather, etc. An ideal solution would be to use *orbital* solar power stations, which would generate electricity and then transmit it to Earth as a microwave beam. The problem with this solution is that a high-energy microwave beam may damage whatever it accidentally hits. So, the better solution is to have several orbital stations and several receivers, so that the resulting beams do not reach the dangerous level. Again, interval methods provide a solution to this problem.

FROM INTERVAL COMPUTATIONS TO SOFT COMPUTING

Why soft computing

It is known that some interval computation problems are not feasible [10]; this means that if we do not have any additional information, we cannot, in general, solve these problems efficiently. We can rephrase this negative result in a positive form: to solve these problems, we must add some *expert knowledge*. The methodologies which use expert knowledge to solve problems are known as *soft computing*; so, we can reformulate our conclusion as saying that many aerospace problems require soft computing.

We have shown that the use of soft computing methods can indeed make these problems feasibly solvable [5].

Two main problems of satellite data processing

One of the main objectives of PACES is processing satellite data with the purpose of extracting useful geophysical, environmental, and other earth-related information. For this data processing to be successful, we need to solve two major problems:

- First, satellite imaging provides us with an unusually *enormous* amount of data; traditional methods of data processing, which work well for smaller amounts of data, often require too long a time when applied to satellite images; thus, new methods are needed.
- Second, many traditional data and image processing techniques depend on *experts* to *do* many *routine subtasks* such as mosaicking images, identifying different vegetation or cloud patterns, etc. With a huge amount of data coming from the satellites, it is no longer possible to use experts to process all this data, these subtasks need to be automated.

In solving both problems, soft computing techniques such as fuzzy, neural, etc., naturally emerge.

Soft computing helps in solving the first problem of satellite data processing

- Traditional methods of data processing are based on thorough statistical analysis of the problems.
- Due to the continuing progress in satellite imaging techniques and to the continuing discovery of new applications, there is no time to follow a (rather slow) traditional statistical analysis approach. Therefore, new heuristic methods are needed, methods which use, in addition to statistics, also informal expert ideas.

Fuzzy, neural, and other soft computing techniques allow us to *formalize* these expert ideas, and, which is very important, to formalize these ideas in a scientifically justified consistent fashion, thus *increasing* the *reliability* of the results of data processing. An example of such formalization is given in [8]. An important heuristic idea is the idea of choosing the *simplest* explanation. In computer science, there are natural measures of complexity and simplicity, such as the length and the time of the program, but with respect to all these formal measures, finding the simplest explanation becomes a computationally un-feasible task; soft computing enables us to explain the existing feasible modifications of this idea and to come up with alternative feasible modifications [11].

Soft computing helps in solving the second problem of satellite data processing

Experts have trouble describing how exactly they mosaic or how exactly they identify features. Experts can, at best, formulate their rules in terms of words of natural language (like "a little bit"). To us these informal rules, we must use a special techniques for transforming such rules into automated control: fuzzy logic.

If even rules are not available, then the only way to automate is to observe the experts' behavior in several cases and extrapolate. One of the best extrapolation techniques, which is the most appropriate for our purposes because it simulates the way humans do extrapolation, is neural networks.

Applications of soft computing methodology include *image processing* (including processing satellite images and clustering) [12,13,20], as well as related problems such as optimization, control, and modeling.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by NSF grants No. DUE-9750858 and CDA-9522207, by United Space Alliance, grant No. NAS 9-20000 (PWO C0C67713A6), by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-95-1-0518, and by the National Security Agency under Grant No. MDA904-98-1-0561.

REFERENCES

1. C. Baral and Tran Cao Son (1996) Relating theories of actions and reactive robot control, SC-COSMIC, South Central Computational Sciences in Minority Institutions Consortium, Second Student Conference in Computational Sciences, October 25–27, 1996, El Paso, TX, Abstracts, p. 3.

2. D. I. Doser, K. C. Crain, M. R. Baker, V. Kreinovich, and M. C. Gerstenberger (1998) Estimating uncertainties for geophysical tomography, Reliable Computing 4(3), 241–268.

3. A. Elfes (1989) Using Occupancy Grids for Mobil Robot Perception and Navigation, IEEE Computer, June 1989, 46–57.

4. O. Fuentes and R. C. Nelson (1996) The Virtual Tool Approach to Dextrous Telemanipulation, Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota, April 1996.

5. O. N. Garcia, V. Kreinovich, L. Longpré, and H. T. Nguyen (1998) Complex problems: granularity is necessary, granularity helps, In: Nguyen H. P. et al. (eds.), Proceedings of the Vietnam-Japan Bilateral Symposium on Fuzzy Systems and Applications VJFUZZY'98 (HaLong Bay, Vietnam, 30th September–2nd October, 1998) 449–455.

6. M. Jägersand, O. Fuentes, and R. C. Nelson (1997) Experimental Evaluation of Uncalibrated Visual Servoing for Precision Manipulation, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, Albuquerque, New Mexico, April 1997.

7. Kortenkamp, D., Nourbakhsh, I., and Hinkle, D., The 1996 AAAI Mobile Robot Competition and Exhibition, AI Magazine, Spring 1997, pp. 25–32.

8. O. Kosheleva, V. Kreinovich, B. Bouchon-Meunier, and R. Mesiar (1998) Operations with Fuzzy Numbers Explain Heuristic Methods in Image Processing, Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'98) (Paris, France, July 6–10, 1998).

9. V. Kreinovich, and L. O. Fuentes (1998) Telemanipulation: The Virtual Tool Approach and Its Interval-Based Justification, In: G. Alefeld et al. (eds.), Interval Computations and its Applications to Reasoning Under Uncertainty, Knowledge Representation, and Control Theory. Proceedings of MEXICON'98, Workshop on Interval Computations, 4th World Congress on Expert Systems (México City, México, 1998).

10. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl (1997) Computational complexity and feasibility of data processing and interval computations, Kluwer, Dordrecht.

11. V. Kreinovich, L. Longpré, and H. T. Nguyen (1998) Towards formalization of feasibility, randomness, and commonsense implication: Kolmogorov complexity, and the necessity of considering (fuzzy) degrees, In: Nguyen H. P. et al. (eds.), Proceedings of the Vietnam-Japan Bilateral Symposium on Fuzzy Systems and Applications VJFUZZY'98 (HaLong Bay, Vietnam, 30th September–2nd October, 1998) 294–302.

12. V. Kreinovich, H. T. Nguyen, S. A. Starks, and Y. Yam (1998) Decision making based on satellite images: optimal fuzzy clustering approach, Proceedings of the 37^th IEEE Conference on Decision and Control CDC'98 (Tampa, Florida, December 16–18, 1998).

13. V. Kreinovich, H. T. Nguyen, and Y. Yam (1998) Optimal Choices of Potential Functions in Fuzzy Clustering, The Chinese University of Hong Kong, Department of Mechanical & Automation Engineering, Technical Report CUHK-MAE-98-001 (January 1998).

14. D. Morales (1996) Path replanning through virtual map manipulations, SC-COSMIC, South Central Computational Sciences in Minority Institutions Consortium, Second Student Conference in Computational Sciences, October 25–27, 1996, El Paso, TX, Abstracts, pp. 25–26.

15. D. Morales (1996) Path re-planning through virtual map manipulations for a deliberate and reactive agent, Master Thesis, The University of Texas at El Paso, Department of Computer Science, December 1996.

16. R. C. Nelson, M. Jägersand, and O. Fuentes (1995) Virtual Tools: A Framework for Simplifying Sensory-Motor Control in Complex Robotic Systems, Proceedings of the '95 Vision for Robots Workshop, Pittsburgh, PA, August 1995.

17. R. Osegueda, C. Ferregut, M. J. George, J. M. Gutierrez, and V. Kreinovich (1997) Computational geometry and artificial neural networks: a hybrid approach to optimal sensor placement for aerospace NDE, In: C. Ferregut et al. (eds), Proceedings of the International Workshop on Intelligent NDE Sciences for Aging and Futuristic Aircraft (El Paso, TX, September 30–October 2, 1997) 59–71.

18. R. Osegueda, C. Ferregut, M. J. George, J. M. Gutierrez, and V. Kreinovich (1997) Non-Equilibrium Thermodynamics Explains Semiotic Shapes: Applications to Astronomy and to Non-Destructive Testing of Aerospace Systems, Proceedings of the International Conference on Intelligent Systems and Semiotics (ISAS'97) (National Institute of Standards and Technology Publ., Gaithersburg, MD, 1997) 378–382.

19. R. Osegueda, C. Ferregut, M. J. George, J. M. Gutierrez, and V. Kreinovich (1998) Maximum entropy approach to optimal sensor placement for aerospace non-destructive testing, In: G. Erickson et al. (eds.), Maximum Entropy and Bayesian Methods, Kluwer, Dordrecht, 277–289.

20. A. T. Popov, H. T. Nguyen, and L. K. Reznik (1998) An Application of Fuzzy Mathematical Morphology to Interval-Valued Knowledge Representation: A Remark, Reliable Computing 4(3), 283–290.

21. A. Saffiotti, K. Konolige, and E. H. Ruspini (1995) A multi-valued logic approach to integrating planning and control, Artificial Intelligence 76(1–2), 481–526.

22. S. A. Starks and V. Kreinovich (1997) Soft Computing: Frontiers? A Case Study of Hyper-Spectral Satellite Imaging, In: L. Medsker (ed.), Frontiers in Soft Computing and Decision Systems, AAAI Press (Publication No. FS-97-04) 52–57.

23. S. A. Starks and V. Kreinovich (1998) Environmentally-oriented processing of multi-spectral satellite images: new challenges for Bayesian methods, In: G. Erickson et al. (eds.), Maximum Entropy and Bayesian Methods, Kluwer, Dordrecht, 271–276.

24. S. A. Starks and V. Kreinovich (1998) Multi-spectral inverse problems in satellite image processing, In: A. Mohamad-Djafari (ed.), Bayesian Inference for Inverse Problems, Proceedings of the SPIE/International Society for Optical Engineering, Vol. 3459 (San Diego, CA) 138–146.

25. K. C. Wu (1995) Interval methods in mobile robot control, Reliable Computing, 1995, Supplement (Extended Abstracts of APIC'95: International Workshop on Applications of Interval Computations, El Paso, TX, Febr. 23–25, 1995), 224–226.

26. K. C. Wu (1996) Fuzzy interval control of mobile robots, Computers and Electrical Engineering 22(3), 211–219.