Interval Constraint Solving for Camera Control and Motion Planning

FRÉDÉRIC BENHAMOU, FRÉDÉRIC GOUALARD, ÉRIC LANGUÉNOU, and MARC CHRISTIE
Laboratoire d'Informatique de Nantes-Atlantique FRE CNRS 2729

Many problems in robust control and motion planning can be reduced to either finding a sound approximation of the solution space determined by a set of nonlinear inequalities, or to the "guaranteed tuning problem" as defined by Jaulin and Walter, which amounts to finding a value for some tuning parameter such that a set of inequalities be verified for all the possible values of some perturbation vector. A classical approach to solving these problems, which satisfies the strong soundness requirement, involves some quantifier elimination procedure such as Collins' Cylindrical Algebraic Decomposition symbolic method. Sound numerical methods using interval arithmetic and local consistency enforcement to prune the search space are presented in this article as much faster alternatives for both soundly solving systems of nonlinear inequalities, and addressing the guaranteed tuning problem whenever the perturbation vector has dimension 1. The use of these methods in camera control is investigated, and experiments with the prototype of a declarative modeler to express camera motion using a cinematic language are reported and commented upon.

Categories and Subject Descriptors: D.3.3 [**Programming Languages**]: Language Constructs and Features—*Constraints*; G.1.0 [**Numerical Analysis**]: General—*Interval arithmetic*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*Animations*

General Terms: Algorithms, Theory, Reliability

Additional Key Words and Phrases: Camera control, inner approximation, interval constraint, universal quantifier

1. INTRODUCTION

Designing electronic circuits [Ebers and Moll 1954], identifying the structure of complex molecules [Emiris and Mourrain 1999], or computing the quantity of chemical elements produced by some reaction [Meintjes and Morgan 1990]

The research reported here was supported in part by the INRIA LOCO project and a project of the French/Russian A.M. Liapunov Institute.
This is a revised and extended version of an article presented at the sixth International Conference on Principles and Practice of Constraint Programming (CP '00), Singapore.
Authors' addresses: Laboratoire d'Informatique de Nantes-Atlantique FRE CNRS 2729, 2, rue de la Houssiniere—BP 92208, F-44322 Nantes Cedex 3, France; email: {Frederic.Benhamou,Frederic.Goualard,Eric.Languenou,Marc.Christie}@lina.univnants.fr.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 1515 Broadway, New York, NY 10036 USA, fax: +1 (212) 869-0481, or permissions@acm.org.
© 2004 ACM 1529-3785/04/1000-0732 $5.00

are all problems—among many others—that can be modeled by sets of real nonlinear equations and inequations. *Reliably* solving these systems (that is, delivering approximate solutions as close as possible to the true ones) with the limited set of real numbers representable on computers has generated a large amount of literature since the very dawn of computer science [Turing 1948; Rademacher 1948; Wilkinson 1963].

Following Sunaga's and Moore's seminal works [Sunaga 1958; Moore 1966], *interval analysis* has been identified as a key tool for ensuring reliability and *completeness* (that is, the capability to retain all the solutions) in the solving process.

Interval constraint solving [Older and Vellino 1990; Benhamou 1995] uses interval arithmetic in algorithms alternating *propagation steps* [Waltz 1975; Mackworth 1977], which enforce some local consistency notion to tighten the variables' feasible domain, and *search steps* to isolate different solutions and overcome the weakness of the propagation steps.

Interval constraint solvers such as clp(BNR) [Benhamou and Older 1997], ILOG Solver [Puget 1994], or Numerica [Van Hentenryck et al. 1997b] have been shown to be efficient tools for solving some challenging nonlinear constraint systems [Puget and Van Hentenryck 1998; Granvilliers and Benhamou 2001]). Relying on interval arithmetic, they guarantee completeness and isolate punctual solutions with an "arbitrary" accuracy. They take as input a constraint system and a Cartesian product of domains for the variables occurring in the constraints; their output is a set $\mathcal{S}_o$ of boxes approximating each solution contained in the input box.

However, *soundness* (i.e., the property that all boxes returned contain nothing but true solution points) is not guaranteed while it is sometimes a strong requirement. Consider, for instance, a civil engineering problem [Sam 1995] such as floor design where retaining nonsolution points may lead to a physically infeasible structure. As pointed out by Ward et al. [1989] and Shary [1999], one may expect different properties from the boxes composing $\mathcal{S}_o$ depending on the problem at hand, namely: every element in any box is a solution, or there exists at least one solution in each box. Interval constraint solvers ensure only, at best, the second property.

Interval constraint solving techniques also suffer from a severe limitation in that they can only handle constraint systems with discrete solutions. Yet, many applications in robust control or error-bounded estimation [Jaulin and Walter 1993] are modeled by systems of nonlinear inequalities, whose solution set is usually not discrete.

In addition, even more applications can be reduced to what Jaulin and Walter [1996] called the "guaranteed tuning problem," which amounts to finding the values for some tuning parameter such that a set of inequalities be verified for all the possible values of some perturbation vector.[1] More precisely:

[1]Jaulin and Walter [1996] restricted the problem to finding only one value for the tuning parameter. The generalization adopted here is of interest to any application in which the user has to be given the choice of the solution to adopt, such as in the camera control application presented in this article.

Given **B***, a box of feasible values for some tuning parameter vector* γ *and* **D***, a box of feasible values for some perturbation vector* π*, find the set* $\mathcal{S}_\gamma$ *defined by*

$$\mathcal{S}_\gamma = \{\gamma \in \mathbf{B} \mid \forall \pi \in \mathbf{D} : \mathbf{f}(\gamma, \pi) \geq \mathbf{0}\},$$

where **f** *is a vector of nonlinear functions, and the inequality is to be taken componentwise.*

Problems ranging from robust control [Abdallah et al. 1996] and camera control [Drucker and Zeltzer 1994] to motion planning [Tarabanis 1990] can all be formulated as guaranteed tuning problems. Consider for example the following application:

Example 1.1 (*A Collision-Free Problem*). A mobile robot arm is composed of three segments of respective lengths d_1, d_2, and d_3, and three motor-controlled axes (see Figure 1). The trajectory of the robot's hand $P(t)$ is therefore determined by three angle functions $\alpha_1(t)$, $\alpha_2(t)$, and $\alpha_3(t)$. The problem is to find all points in the $2D$ space that do not collide with the robot's hand, that is, computing all the (x, y) coordinates such that the distance between $P(t)$, for all t, and (x, y) is greater than some given value d (here, the size of the robot's hand):

$$\forall t \in [0, 1]: \sqrt{(x - P_x(t))^2 + (y - P_y(t))^2} \geq d,$$

where $P_x(t)$ and $P_y(t)$ represent the coordinates of $P(t)$ at time t, defined by

$$\begin{cases} P_x(t) = & d_1 \sin \alpha_1(t) + d_2 \sin(\alpha_1(t) + \alpha_2(t) - \pi) + d_3 \sin(\alpha_1(t) + \alpha_2(t) + \alpha_3(t)), \\ P_y(t) = & d_1 \cos \alpha_1(t) + d_2 \cos(\alpha_1(t) + \alpha_2(t) - \pi) + d_3 \cos(\alpha_1(t) + \alpha_2(t) + \alpha_3(t)). \end{cases}$$

Until recently, the soundness issue and the presence of quantifiers called for symbolic methods and quantifier elimination procedures such as Cylindrical Algebraic Decomposition (CAD) [Collins 1975]. Unfortunately, these techniques are either too slow, or limited to polynomial constraints.

The advent of interval analysis led to the devising of simple and sound algorithms to solve systems of inequalities [Jaulin and Walter 1993; Garloff and Graf 1999] and the guaranteed tuning problem (restricted to the finding of only one value, though) [Jaulin and Walter 1996]. By and large, many of these algorithms are but sophisticated interval extensions of simple search procedures, meaning that are they computationally expensive, even for small problems.

In this article, we present sound algorithms that draw upon efficient complete interval constraint solving pruning methods to soundly solve systems of inequalities and the guaranteed tuning problem for the case of unidimensional perturbation vectors. This kind of problem is the one occurring in motion planning and camera control applications, where the only universally quantified variable is usually the time.

The outline of the article is as follows: in order to be reasonably self-content, the basics of interval analysis are presented in Section 2; related works on both solving systems of nonlinear inequalities and the guaranteed tuning problem are presented in Section 3; their strong points and weaknesses regarding the

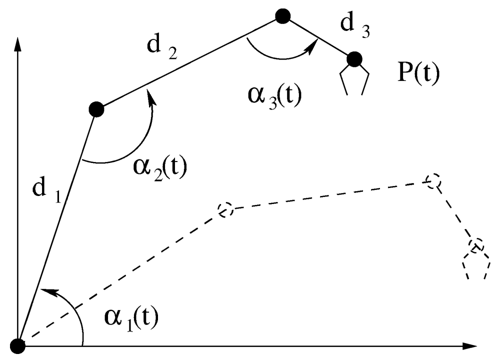


Fig. 1. Avoiding collisions with the arm of a robot.

applications targeted are also pointed out; interval constraint solving is introduced in Section 4 as a basis for the new sound algorithms presented in Section 5; the modeling of a camera control problem in terms of these new sound interval constraint methods is then described in Section 6; the results with a prototype of a declarative modeler allowing a nontechnician user to control the positioning of a camera by means of a cinematic language are commented on in Section 7 and contrasted with the ones obtained by Jardillier and Languénou [1998] on the same problems with a different approach; finally, Section 8 discusses directions for future research, and the Appendix defines some notation.

2. INTERVAL ANALYSIS

In this section, we limit the presentation of interval analysis to the concepts needed in the sequel of the article. The reader is referred to works by Moore, Hansen, and others [Moore 1966; Alefeld and Herzberger 1983; Hansen 1992; Neumaier 1990] for a more complete presentation.

The finite nature of computers implies that they can only represent and manipulate a small subset of the real numbers represented in a floating-point format. Since the mid-1980s, most computers comply with the IEEE 754 standard [IEEE 1985] specifying the format of floating-point numbers.

The set of floating-point numbers $\mathbb{F}$ is but a very small subset of real numbers. In addition, it is not closed for arithmetic operations, meaning that rounding of the results to representable floating-point numbers must usually take place. We say that an operation is *correctly rounded* when the value chosen, whenever the true result is not representable, is the closest floating-point number. Note that the IEEE 754 standard only requires the operators $+, -, \times, \div, \sqrt{}$ to be correctly rounded. The precision of the other operators is implementation dependent. See the article by Lefèvre et al. [1998] for more information on this topic.

Given a floating-point number a, let a^+ (respectively a^-) be the smallest float greater than a (respectively greatest float smaller than a).

Rounding makes the reliable solving of systems of nonlinear equalities or inequalities a challenging task, to which a large amount of articles and books has been devoted.

One solution advocated by Moore [1966] to control the rounding errors is to use *interval arithmetic*, that is to replace reals by intervals containing them, whose bounds are representable numbers. For example:

$$\pi \in [3.14, 3.15].$$

An interval I with representable bounds is called a *floating-point interval*. It is of the form $I = [a, b] = \{r \in \mathbb{R} \mid a \leq r \leq b,\ \text{with } a, b \in \mathbb{F}\}$. A nonempty interval $I = [a, b]$ such that $b \leq a^+$ is said *canonical*. In the same way, a Cartesian product of intervals (or *box*) is said to be canonical whenever it is canonical in all its dimensions. In the sequel, vectors or Cartesian products are written in boldface.

Let $\mathbb{I}$ be the set of all floating-point intervals. Since we will only deal with this kind of intervals in the rest of the article, we will refer to them as simply *intervals*.

Operations and functions over reals are also replaced by an *interval extension* having the *containment property*. Given a real function $f : \mathbb{R}^n \to \mathbb{R}$ and a box $\mathbf{B} \in \mathbb{I}^n$, let $f(\mathbf{B}) = \{f(\mathbf{r}) \mid \mathbf{r} \in \mathbf{B}\}$, $\mathcal{D}_f$ be the domain of f, and $\mathcal{D}_f^{\mathbb{I}} = \{\mathbf{B} \in \mathbb{I}^n \mid \mathbf{B} \subseteq \mathcal{D}_f\}$.

Definition 2.1 (*Interval Extension*). Given a real function $f : \mathbb{R}^n \to \mathbb{R}$, an *interval extension* $F : \mathbb{I}^n \to \mathbb{I}$ of f is an interval function verifying

$$\mathsf{Outer}(f(\mathbf{B})) \subseteq F(\mathbf{B}) \qquad \forall \mathbf{B} \in \mathcal{D}_f^{\mathbb{I}},$$

where $\mathsf{Outer}(\rho) = \bigcap\{\mathbf{B} \in \mathbb{I}^n \mid \rho \subseteq \mathbf{B}\}$ for any real relation $\rho \subseteq \mathbb{R}^n$.

The extensions of some basic operators are defined as follows:

$$\begin{aligned}
[a, b] + [c, d] &= [a + c, b + d],\\
[a, b] - [c, d] &= [a - d, b - c],\\
[a, b] \times [c, d] &= [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)],\\
\exp([a, b]) &= [\exp(a), \exp(b)].
\end{aligned}$$

Note that, in practice, the bounds computed have to be *outward rounded* in order to preserve the containment property.

A particular interval extension, the *natural interval extension* is obtained by replacing syntactically in a real function all the real constants by intervals containing them, all the real variables by interval variables, and all operators by their interval extension.

Interval arithmetic is commutative, but it is neither associative (due to floating-point numbers wanting for this property themselves), nor distributive. It enjoys instead a *subdistributive property*, namely:

$$\forall I, J, K \in \mathbb{I}: \quad I(J + K) \subseteq IJ + IK.$$

The subdistributivity property has an important impact in that equivalent forms for a function over reals may not have equivalent natural extensions.

In the rest of this article, we will often consider the set of real numbers represented by a set of Cartesian product of intervals. Given $\mathcal{P}(\mathcal{S})$ the *power set*

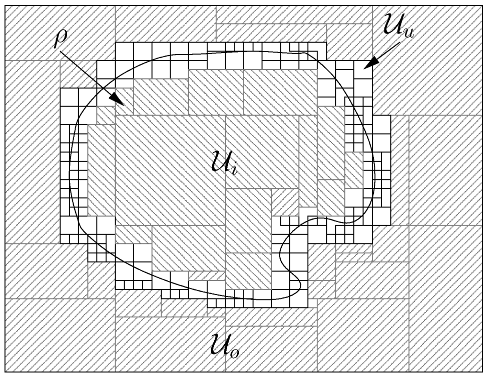


Fig. 2. Subpaving of the relation ρ.

of any set $\mathcal{S}$, we then introduce the operator $\wr\cdot\wr\colon \mathcal{P}(\mathbb{I}^n) \rightarrow \mathbb{R}^n$ defined as follows:

$$\forall \mathbf{B_1}, \ldots, \forall \mathbf{B_n} \in \mathbb{I}^n \colon \wr\{\mathbf{B_1}, \ldots, \mathbf{B_n}\}\wr = \{\mathbf{r} \in \mathbb{R}^n \mid \exists i \in \{1, \ldots, n\} \text{ s.t. } \mathbf{r} \in \mathbf{B_i}\}.$$

Last, given a box $\mathbf{B} = I_1 \times \cdots \times I_n$, an integer $k \in \{1, \ldots, n\}$, and an interval J, let $\mathbf{B}_{I_k \leftarrow J} = I_1 \times \cdots \times I_{k-1} \times J \times I_{k+1} \times \cdots \times I_n$ be the box $\mathbf{B}$ where I_k has been replaced by J.

3. RELATED WORK

Computing a *subpaving* (Figure 2) of a real relation $\rho \subseteq \mathbb{R}^n$ consists in partitioning $\mathbb{R}^n$ into three sets $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u)$ of nonoverlapping boxes with the properties:

$$\begin{cases} \wr\mathcal{U}_i\wr \subseteq \rho, \\ \wr\mathcal{U}_o\wr \cap \rho = \varnothing, \\ \wr\mathcal{U}_i \cup \mathcal{U}_u\wr \supseteq \rho. \end{cases} \tag{1}$$

Given an n-ary constraint c, let $\rho_c \subseteq \mathbb{R}^n$ be the underlying relation, that is:

$$\rho_c = \{(r_1, \ldots, r_n) \in \mathbb{R}^n \mid c(r_1, \ldots, r_n)\}$$

Given an n-ary constraint c and a box $\mathbf{B} \in \mathbb{I}^n$, and assuming the existence of some procedure GlobSat with the following properties:

$$\begin{cases} \text{GlobSat}(c, \mathbf{B}) = \text{true} & \Rightarrow \mathbf{B} \subseteq \rho_c, \\ \text{GlobSat}(c, \mathbf{B}) = \text{false} & \Rightarrow \mathbf{B} \cap \rho_c = \varnothing, \\ \text{GlobSat}(c, \mathbf{B}) = \text{unknown} & \Rightarrow \text{indeterminate}, \end{cases}$$

it is easy to devise a systematic procedure to compute the subpaving of ρ_c by alternating *evaluation steps* with GlobSat, and *splitting steps* whenever GlobSat returns “unknown.” The precise algorithm is presented in Table I for a conjunction of constraints.

The StoppingCriterion function appearing on Line 10 of Algorithm Subpaving returns “true” or “false” depending whether the box given as an argument should still be considered for splitting. A typical instance of this method is testing for canonicity of the box. It is also possible to speed up the computation by

Table I. Subpaving Algorithm for a Conjunction of Atomic Constraints $c_1 \wedge \cdots \wedge c_m$

```
1 Subpaving(in: $\{c_1, \ldots, c_m\}$, $\mathbf{B} \in \mathbb{I}^n$; out: $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \in \mathcal{P}(\mathbb{I}^n)^3$)
2 begin
3   sat $\leftarrow$ GlobSat($c_1, \mathbf{B}$) $\wedge \cdots \wedge$ GlobSat($c_m, \mathbf{B}$)
4   switch sat in
5     true:
6       return $(\{\mathbf{B}\}, \varnothing, \varnothing)$
7     false:
8       return $(\varnothing, \{\mathbf{B}\}, \varnothing)$
9     unknown:
10      if StoppingCriterion($\mathbf{B}$) then
11        return $(\varnothing, \varnothing, \{\mathbf{B}\})$
12      else
13        $(\mathbf{B_1}, \ldots, \mathbf{B_k}) \leftarrow$ Split$_k$($\mathbf{B}$)
14        return $\biguplus_{j=1}^{k}$ Subpaving($\{c_1, \ldots, c_m\}, \mathbf{B_j}$)
15      endif
16  end
17 end
```

using another instance of StoppingCriterion that would avoid splitting boxes whose width is smaller than some ε.

The Split$_k$ function on Line 13 splits a box $\mathbf{B}$ into k nonoverlapping subboxes whose union is equal to $\mathbf{B}$.

The $\biguplus$ operator on Line 14 applies on vectors of sets and performs their unions componentwise:

$$\biguplus_i \{(\mathcal{S}_1^i, \ldots, \mathcal{S}_n^i)\} = \left(\bigcup_i \mathcal{S}_1^i, \ldots, \bigcup_i \mathcal{S}_n^i\right).$$

The subpaving algorithm has been used by several authors to compute sound boxes for systems of inequalities. They differed by the way they implemented the GlobSat method.

In SIVIA, Jaulin and Walter [1993] used interval arithmetic containment properties: given a constraint c: $f(x_1, \ldots, x_n) \leq 0$, and a box $\mathbf{B}$, they evaluated the natural interval extension of f over $\mathbf{B}$. If the right bound of the result is negative, GlobSat returns "true"; if the left bound is strictly positive GlobSat returns "false"; otherwise it returns "unknown." SIVIA is able to process any kind of inequality constraints, be they linear or not, polynomial or not. The drawback of this approach is that, for unstable f functions, the natural interval evaluation leads to large intervals that do not permit deciding whether the box $\mathbf{B}$ is included in ρ_c or not. It is then necessary to split the box a lot.

Jaulin and Walter [1996] have also devised an algorithm to solve the guaranteed tuning problem (restricted to the finding of only one value). It is based on SIVIA, so it suffers from the same drawbacks as SIVIA itself.

The algorithm described by Kutsia and Schicho [1999] implements another instance of Subpaving where GlobSat is obtained by testing some criterion using floating-point numbers of arbitrary precision. An important limitation is that their algorithm can only handle polynomial strict inequalities.

Garloff and Graf [1999] also restricted themselves to polynomial strict inequalities. They expanded the polynomial inequalities into Bernstein polynomials: let $\mathbf{I} = (i_1, \dots, i_n)$ be a multi-index (vector of nonnegative integers), and $\mathbf{x}^{\mathbf{I}} = x_1^{i_1} \dots x_n^{i_n}$ be a monomial. Given a polynomial $p \in \mathbb{R}[x_1, \dots, x_n]$, let $\mathcal{S}$ be a set of multi-indices such that $p(\mathbf{x}) = \sum_{\mathbf{I} \in \mathcal{S}} a_{\mathbf{I}} \mathbf{x}^{\mathbf{I}}$. For any n-ary Cartesian product of domains $\mathbf{D}$, one can express p in terms of Bernstein coefficients:

$$p(\mathbf{x}) = \sum_{\mathbf{I} \in \mathcal{S}} b_{\mathbf{I}}(\mathbf{D}) B_{N,\mathbf{I}}(\mathbf{x}),$$

where $B_{N,\mathbf{I}}(\mathbf{x})$ is the $\mathbf{I}$th Bernstein polynomial of degree N.

It is then well known that the following property does hold [Farouki and Rajan 1987]:

$$\forall \mathbf{a} \in \mathbf{D} : \min_{\mathbf{I} \in \mathcal{S}} b_{\mathbf{I}}(\mathbf{D}) \leq p(\mathbf{a}) \leq \max_{\mathbf{I} \in \mathcal{S}} b_{\mathbf{I}}(\mathbf{D}) \quad (\textit{convex hull property}).$$

As a consequence, it is possible to accurately bound the range of $p(\mathbf{x})$ on every box, and then to know whether an inequality such as $p(\mathbf{x}) < 0$ does hold or not. By splitting the initial box and evaluating the corresponding Bernstein coefficients, one can isolate all the boxes verifying a set of inequality constraints.

Haroud and Faltings [1994] did not rely on Algorithm Subpaving. Instead, they represented explicitly the subpaving of the relation ρ_c associated to each n-ary inequality constraint by a 2^n-tree of boxes. The category of each box (in $\mathcal{U}_i$, $\mathcal{U}_o$, or $\mathcal{U}_u$) is determined by finding the intersection of the curve with the edges of the box. Sets of constraints are handled by performing intersections of the respective 2^n-trees. In order to avoid a combinatorial explosion, n must be small. As a consequence, they have chosen to decompose the user's constraints into ternary constraints, so that they only manipulated octrees. The original algorithm is able to compute inner approximations (that is, subsets) of the solution space; on the other hand, it was not designed to handle the guaranteed tuning problem.

The method presented by Collavizza et al. [1999] is strongly related to the one we present in the following, since they relied on usual interval constraint solving techniques to compute sound boxes for some constraint system. Starting from a seed that is known to belong to the solution space, they enlarged the domain of the variables around it in such a way that the new box computed was still included in the solution space. They did so by using local consistency techniques to find the points at which the truth value of the constraints change. Their algorithm is particularly well suited for the applications they targeted namely, the enlargement of tolerances. It was, however, not designed to solve the guaranteed tuning problem. In addition, it is necessary to obtain a seed for each connected subset of the solution space, and to apply the algorithm on each seed if one is interested in computing several solutions (e.g., to ensure representativeness of the samples).

In order to tighten a box $\mathbf{B}$ of variables' domains for a problem of the form $\forall v \in I_k : c_1 \wedge \cdots \wedge c_m$, Jardillier and Languénou [1998] computed an inner

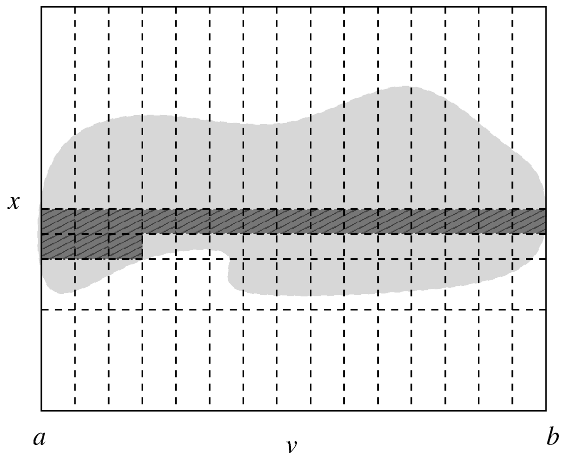


Fig. 3. JLA algorithm: solving $\forall v \in [a, b]\colon c(x, v)$.

approximation by decomposing the initial domain I_k of v into canonical intervals $I_k^1,\dots,I_k^p$, and testing whether $c_1 \wedge \cdots \wedge c_m$ does hold for the boxes $I_1 \times \cdots \times I_k^1 \times \cdots \times I_n,\dots,I_1 \times \cdots \times I_k^p \times \cdots \times I_n$. These evaluations give results in a three-valued logic (*true*, *false*, *unknown*). Boxes labeled *true* contain only solutions, boxes labeled *false* contain no solution at all, and boxes labeled *unknown* are recursively split and retested until they may be asserted true or false, or canonicity is reached (see Figure 3). Retained boxes are those verifying

$$\forall j \in \{1,\dots,p\}\colon \begin{cases} \mathsf{GlobSat}(c_1, I_1 \times \cdots \times I_k^j \times \cdots \times I_n) = \text{true} \\ \wedge \\ \dots \\ \wedge \\ \mathsf{GlobSat}(c_m, I_1 \times \cdots \times I_k^j \times \cdots \times I_n) = \text{true}. \end{cases}$$

The precise algorithm is described in Table II. In the initial call, the l parameter is equal to the left bound of v's domain. The $\mathsf{Split}_k^{\setminus v}$ procedure in Line 18 is identical to the Split_k procedure presented previously, except that it never splits the domain of the universally quantified variable v.

Once again, Algorithm JLA is but a sophisticated instance of Algorithm Subpaving, which means that its efficiency strongly depends on the quality of the GlobSat procedure. Like in SIVIA, Jardillier and Languénou [1998] used the natural interval extension of the constraints. As a consequence, their algorithm is computationally expensive in many cases, as soon as some moderate precision in the computation of the inner approximation is required. We refer the reader to Section 7 for precise figures.

Due to lack of space, we will not present the Cylindrical Algebraic Decomposition method for quantifier elimination here, and we refer the reader to the good introduction in Jirstrand [1995].

Table II. Evaluation-Based Propagation Algorithm for $\forall v \in [\underline{v}, \overline{v}]: c_1 \wedge \cdots \wedge c_m$

```
JLA(in: $\{c_1, \ldots, c_m\}$, $\mathbf{B} \in \mathbb{I}^n$, a variable $v$, $l \in \mathbb{F}$; out: $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \in \mathcal{P}(\mathbb{I}^n)^3$)
begin
    $\mathbf{B}' \leftarrow \mathbf{B}_{I_v \leftarrow [l, l^+]}$
    sat $\leftarrow$ GlobSat($c_1, \mathbf{B}$) $\wedge \cdots \wedge$ GlobSat($c_m, \mathbf{B}$)
    switch sat in
        true:
            if $l^+ = \overline{v}$ then
                return ($\{\mathbf{B}\}, \varnothing, \varnothing$)
            else
                return JLA($\{c_1, \ldots, c_m\}, \mathbf{B}, v, l^+$)
            endif
        false:
            return ($\varnothing, \{\mathbf{B}\}, \varnothing$)
        unknown:
            if StoppingCriterion($\mathbf{B}$) then
                return ($\varnothing, \varnothing, \{\mathbf{B}\}$)
            else
                ($\mathbf{B_1}, \ldots, \mathbf{B_k}$) $\leftarrow$ Split$_k^{\setminus v}$($\mathbf{B}$)
                return $\biguplus_{j=1}^{k}$ JLA($\{c_1, \ldots, c_m\}, \mathbf{B_j}, v, l$)
            endif
    end
end
```

4. INTERVAL CONSTRAINT SOLVING

Since finite representation of numbers by computers hinders the reliable solving of real equations and inequations, *interval constraint solving* relies on *interval arithmetic* [Moore 1966; Alefeld and Herzberger 1983] to compute verified approximate solutions of real constraint systems.

The reader will find in various works [Older and Vellino 1990; Benhamou and Older 1997; Benhamou 1996; Van Hentenryck et al. 1997b; Benhamou et al. 1994; Benhamou et al. 1999] a thorough presentation of the interval constraint solving framework and of the associated state-of-the-art algorithms. In this section, we will describe only the elements that are essential to our purpose. In particular, we will restrict ourselves to the case of nonlinear inequality constraints only, that is, constraints of the form $f(x_1, \ldots, x_n) \leq 0$ for some real function f.

Proofs not given here may be found in the articles cited above.

4.1 Local Consistencies

Discarding all inconsistent values from a box of variables' domains is intractable when the constraints are real ones (consider for instance the constraint $c: \sin(x) = 1, x \in [0, 2]$). Consequently, weak consistencies have been devised, among which one may cite *hull consistency* [Benhamou 1995] and *box consistency* [Benhamou et al. 1994]. Both consistencies permit narrowing variables' domains to (hopefully) smaller domains, preserving all the solutions present in the input. Since they are used as a basis for the algorithms to be introduced in Section 5, they are both presented below.

Discarding values of the variable domains for which c does not hold according to a given consistency notion is modeled by means of outer contracting operators, whose main properties are *contractance*, *completeness*, and *monotonicity*.

4.1.1 *Outer Contracting Operators.* Depending on the considered consistency, one may define different contracting operators for a constraint. In this section, hull consistency and box consistency are first formally presented. Operators based on both consistencies are then given.

Hull consistency is a strong "weak consistency" since a constraint c is said to be hull consistent with respect to a box whenever that box cannot be tightened without losing some solutions for c:

Definition 4.1 (*Hull Consistency*). A real constraint $c(x_1, \ldots, x_n)$ is said to be *hull consistent with respect to a box* $\mathbf{B} = I_1 \times \cdots \times I_n$ if and only if

$$\forall k \in \{1, \ldots, n\}\colon I_k = \mathsf{Outer}(I_k \cap \{r_k \in \mathbb{R} \mid \forall j \in \{1, \ldots, n\} \setminus \{k\}\colon \exists r_j \in I_j \text{ s.t. } (r_1, \ldots, r_n) \in \rho_c\}).$$

An operator enforcing hull consistency for a constraint c computes the smallest box containing the intersection of the input box and the relation ρ_c:

Definition 4.2 (*Outer-Hull Contracting Operator*). Let c be an n-ary constraint, ρ_c its underlying relation, and $\mathbf{B}$ a box. An *outer-hull contracting operator* for c is a function $\mathsf{HC}_c\colon \mathbb{I}^n \to \mathbb{I}^n$ defined by:

$$\mathsf{HC}_c(\mathbf{B}) = \mathsf{Outer}(\mathbf{B} \cap \rho_c).$$

PROPOSITION 4.3 (COMPLETENESS OF HC). *Given a constraint c and a box* $\mathbf{B}$, *the following relation does hold:* $(\mathbf{B} \cap \rho_c) \subseteq \mathsf{HC}_c(\mathbf{B})$.

Operationally, hull consistency is enforced over constraints by decomposing them into conjunctions of *primitive constraints*. The drawback of such a method lies in the loss of domain tightening due to the introduction of new variables during the decomposition process. Box consistency has been introduced by Benhamou et al. [1994] to overcome this problem: the operators enforcing box consistency consider constraints globally, without decomposing them.

Definition 4.4 (*Box Consistency*). Let c be an n-ary real constraint, C an interval extension for c, and $\mathbf{B} = I_1 \times \cdots \times I_n$ a box. The constraint c is said *box consistent with respect to* $\mathbf{B}$ if and only if

$$\forall k \in \{1, \ldots, n\}\colon \quad I_k = \mathsf{Outer}(I_k \cap \{r \in \mathbb{R} \mid C(I_1, \ldots, I_{k-1}, \mathsf{Outer}(\{r\}), I_{k+1}, \ldots, I_n)\}).$$

Intuitively, a constraint c is box consistent with respect to a box $\mathbf{B}$ when each projection $I_j, j \in \{1, \ldots, n\}$ of $\mathbf{B}$ is the smallest interval containing all the elements that cannot be distinguished from solutions of the univariate constraint obtained from C by replacing all the variables but x_j by their current domain due to the inherently limited precision of the computation with floating-point numbers.

Box consistency is enforced over a constraint $c(x_1, \ldots, x_n)$ as follows: n contracting operators (typically using an interval Newton method

[Van Hentenryck et al. 1997b]) are associated to the n univariate interval constraints obtained as described above. The kth operator reduces the domain of x_k by computing the leftmost and rightmost canonical intervals J such that $C(I_1, \dots, I_{k-1}, J, I_{k+1}, \dots, I_n)$ does hold (*leftmost and rightmost quasizeros*).

Using box consistency to narrow down the variable domains of a constraint leads to the notion of *outer-box contracting operator*:

Definition 4.5 (*Outer-Box Contracting Operator*). Given an n-ary constraint c and a box $\mathbf{B}$, an *outer-box contracting operator* $\mathsf{BC3r}_c : \mathbb{I}^n \to \mathbb{I}^n$ for c is defined by

$$\mathsf{BC3r}_c(\mathbf{B}) = \max\{\mathbf{B}' \mid \mathbf{B}' \subseteq \mathbf{B} \text{ and } c \text{ is box consistent with respect to } k \in \{1, \dots, n\} \text{ and } \mathbf{B}'\}.$$

PROPOSITION 4.6 (COMPLETENESS OF BC3r). *Given a constraint c, the following relation does hold for any box* $\mathbf{B}$*:*

$$(\mathbf{B} \cap \rho_c) \subseteq \mathsf{BC3r}_c(\mathbf{B}).$$

Consistencies and associated contracting operators considered so far are such that completeness is guaranteed (no solution lost during the narrowing process). Devising operators that ensure soundness of the results is the topic of the next section.

5. SOUND INTERVAL CONSTRAINT SOLVING

We have seen in the previous section some operators that compute an outer approximation of the relation ρ associated to some nonlinear constraint system. In this section, we will present operators that compute an *inner approximation* of ρ, that is, a subset of the solution space of the corresponding constraint system. As a consequence, the boxes computed by these operators will have the property that any point inside of them is a true solution to the problem, thus ensuring soundness of the results given to the user.

Several definitions for the inner approximation of a real relation exist in the literature, depending on the intended application. Given an n-ary relation ρ, one may single out at least the two following definitions for an inner approximation $\mathsf{Inner}(\rho)$ of ρ:

(1) *Definition A.* $\mathsf{Inner}(\rho) = \mathbf{B_1}$, where $\mathbf{B_1} \in \{\mathbf{B} \in \mathbb{I}^n \mid \mathbf{B} \subseteq \rho\}$ (an inner approximation is any box included in the relation) [Markov 1995; Armengol et al. 1998];

(2) *Definition B.* $\mathsf{Inner}(\rho) = \mathbf{B_1}$, where $\mathbf{B_1} \in \{\mathbf{B} = I_1 \times \cdots \times I_n \mid \mathbf{B} \subseteq \rho \wedge \forall j \in \{1, \dots, n\}, \forall I'_j \supseteq I_j : \mathbf{B}_{I_j \leftarrow I'_j} \subseteq \rho \Rightarrow I'_j = I_j\}$ (an inner-approximation is a box included in the relation that cannot be extended in any direction without containing nonsolution points) [Shary 1999].

Definitions A and B imply that disconnected relations are only very partially represented by one box only, a drawback that is avoided with the following stronger definition, which will be used in this article:

Definition 5.1 (*Inner Approximation Operator*). Given an n-ary real relation $\rho \subseteq \mathbb{R}^n$, the inner-approximation operator $\mathsf{Inner}\colon \mathbb{R}^n \to \mathbb{R}^n$ is defined by

$$\mathsf{Inner}(\rho) = \{\mathbf{r} \in \mathbb{R}^n \mid \mathsf{Outer}(\{\mathbf{r}\}) \subseteq \rho\}.$$

The inner approximation contains all the elements whose enclosing box is included in the relation. The Inner operator enjoys the following properties:

PROPOSITION 5.2 (PROPERTIES OF THE Inner OPERATOR). *The* Inner *operator is contracting, monotonic, idempotent, subdistributive with respect to the union of subsets of* $\mathbb{R}^n$*, and distributive with respect to the intersection of subsets of* $\mathbb{R}^n$.

PROOF. *Given* $\mathcal{A}$, $\mathcal{B}$, *and* $\mathcal{C}$ *three subsets of* $\mathbb{R}^n$, *with* $\mathcal{A} \subseteq \mathcal{B}$, *let* $\mathbf{r}$ *be an element of* $\mathbb{R}^n$ *and* $\mathbf{D} = \mathsf{Outer}(\{\mathbf{r}\})$.

Contractance. Given any real $\mathbf{r} \in \mathsf{Inner}(\rho)$, we have by definition of Inner that $\mathsf{Outer}(\{\mathbf{r}\}) \subseteq \rho$. By definition of Outer, we have that $\mathbf{r} \in \mathsf{Outer}(\{\mathbf{r}\})$, which leads to $\mathbf{r} \in \rho$. Consequently, $\mathsf{Inner}(\rho) \subseteq \rho$.

Monotonicity. For any $\mathbf{r}$, $\mathbf{r} \in \mathsf{Inner}(\mathcal{A})$ implies $\mathbf{D} \subseteq \mathcal{A}$, by definition of Inner; since $\mathcal{A} \subseteq \mathcal{B}$, we have $\mathbf{D} \subseteq \mathcal{B}$, and finally, by definition of Inner, $\mathbf{r} \in \mathsf{Inner}(\mathcal{B})$. Consequently, $\mathcal{A} \subseteq \mathcal{B} \Rightarrow \mathsf{Inner}(\mathcal{A}) \subseteq \mathsf{Inner}(\mathcal{B})$.

Idempotence. We first prove $\mathsf{Inner}(\mathsf{Inner}(\mathcal{A})) \subseteq \mathsf{Inner}(\mathcal{A})$: Given $\mathbf{r} \in \mathsf{Inner}(\mathsf{Inner}(\mathcal{A}))$, we have $\mathbf{D} \subseteq \mathsf{Inner}(\mathcal{A})$, by definition of Inner. By contractance of Inner, $\mathsf{Inner}(\mathcal{A}) \subseteq \mathcal{A}$. We then have $\mathbf{D} \subseteq \mathcal{A}$, and then $\mathbf{r} \in \mathsf{Inner}(\mathcal{A})$, by definition of Inner again.

Now, we prove $\mathsf{Inner}(\mathsf{Inner}(\mathcal{A})) \supseteq \mathsf{Inner}(\mathcal{A})$:

Given $\mathbf{r} \in \mathsf{Inner}(\mathcal{A})$, we have $\mathbf{D} \subseteq \mathcal{A}$ by definition of Inner. Therefore, any element in $\mathbf{D}$ is in $\mathsf{Inner}(\mathcal{A})$. As a consequence, $\mathbf{D} \subseteq \mathsf{Inner}(\mathcal{A})$. Definition of Inner then leads to $\mathbf{r} \in \mathsf{Inner}(\mathsf{Inner}(\mathcal{A}))$.

Subdistributivity with respect to Union. We prove that:

$$\mathsf{Inner}(\mathcal{B}) \cup \mathsf{Inner}(\mathcal{C}) \subseteq \mathsf{Inner}(\mathcal{B} \cup \mathcal{C})$$

We have:

$$\begin{aligned}
\mathsf{Inner}(\mathcal{B}) \cup \mathsf{Inner}(\mathcal{C}) &= \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{B}\} \cup \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{C}\} \\
&= \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{B} \vee \mathsf{Outer}(\{r\}) \subseteq \mathcal{C}\} \\
&\subseteq \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{B} \cup \mathcal{C}\} \\
&\subseteq \mathsf{Inner}(\mathcal{B} \cup \mathcal{C}).
\end{aligned}$$

Distributivity with respect to Intersection. We prove that

$$\mathsf{Inner}(\mathcal{B}) \cap \mathsf{Inner}(\mathcal{C}) = \mathsf{Inner}(\mathcal{B} \cap \mathcal{C}).$$

We have

$$\begin{aligned}
\mathsf{Inner}(\mathcal{B} \cap \mathcal{C}) &= \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq (\mathcal{B} \cap \mathcal{C})\} \\
&= \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{B} \wedge \mathsf{Outer}(\{r\}) \subseteq \mathcal{C}\} \\
&= \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{B}\} \cap \{r \in \mathbb{R}^n \mid \mathsf{Outer}(\{r\}) \subseteq \mathcal{C}\} \\
&= \mathsf{Inner}(\mathcal{B}) \cap \mathsf{Inner}(\mathcal{C}).
\end{aligned}$$

□

5.1 Inner Contracting Operators

The narrowing of variable domains occurring in a constraint is done in the same way as in the outer approximation case: an *inner contracting operator* associated to each constraint discards from the initial box all the inconsistent values along with consistent values that cannot be enclosed in a canonical computer-representable box. The result is a set of boxes.

Definition 5.3 (*Inner Contracting Operator*). Let c be an n-ary constraint. An *inner contracting operator* for c is a function $\mathsf{IC}_c\colon \mathbb{I}^n \to \mathcal{P}(\mathbb{I}^n)$ verifying

$$\forall \mathbf{B} \in \mathbb{I}^n\colon \wr \mathsf{IC}_c(\mathbf{B}) \wr \subseteq \mathsf{Inner}(\mathbf{B} \cap \rho_c).$$

PROPOSITION 5.4 (SOUNDNESS OF IC). *Given a constraint c and an inner-contracting operator IC_c for c, we have*

$$\forall \mathbf{B} \in \mathbb{I}^n\colon \wr \mathsf{IC}_c(\mathbf{B}) \wr \subseteq (\mathbf{B} \cap \rho_c).$$

PROOF. Immediate consequence of Inner and Outer definitions. □

Remark 5.5. Given a constraint c, an inner contracting operator IC_c for c, an outer contracting operator OC_c for c, and a box $\mathbf{B}$, the following relations, deriving from Propositions 4.3 and 5.4, hold:

$$\wr \mathsf{IC}_c(\mathbf{B}) \wr \subseteq (\mathbf{B} \cap \rho_c) \subseteq \mathsf{OC}_c(\mathbf{B}).$$

Inner contracting operators with stronger properties (computation of the greatest representable set included in a relation) may be defined, provided some strong assumption (namely, the ability to determine for any canonical box whether it is included or not in the relation) is fulfilled. These operators are *optimal* in the sense defined below.

Definition 5.6 (*Optimal Inner Contracting Operator*). Let c be an n-ary constraint, and IC_c an inner contracting operator for c. The operator IC_c is said *optimal* if and only if the following relation does hold for any box $\mathbf{B}$:

$$\wr \mathsf{IC}_c(\mathbf{B}) \wr = \mathsf{Inner}(\mathbf{B} \cap \rho_c).$$

Let us consider the following naive algorithm as an implementation of an optimal inner contracting operator: given a constraint c, and a box $\mathbf{B}$ of domains for the variables occurring in c, let us split $\mathbf{B}$ in all its canonical subboxes. We can apply the outer-hull contracting operator HC_c enforcing hull consistency on each of these canonical subboxes. Consider the possible cases for one of the canonical subboxes $\mathbf{D}$:

—$\mathbf{D} \cap \rho_c = \varnothing$. By definition of hull consistency, we must have $\mathsf{HC}_c(\mathbf{D}) = \varnothing$;

—$\mathbf{D} \cap \rho_c \neq \varnothing$. There are again several cases (considered exclusively, from top to bottom):

  —$\mathbf{D} \subseteq \rho_c$: By completeness of hull consistency-based operators, we must have $\mathsf{HC}_c(\mathbf{D}) = \mathbf{D}$. Now, if we consider the negation $\overline{c}$ of c, we must have $\mathsf{HC}_{\overline{c}}(\mathbf{D}) = \varnothing$, by definition of hull consistency, since $\mathbf{D}$ does not contain any element of $\rho_{\overline{c}}$,

  —$\mathbf{D} \cap \rho_{\overline{c}} \neq \varnothing$. The box $\mathbf{D}$ contains both some elements of ρ_c and $\rho_{\overline{c}}$. Consequently, by completeness, $\mathsf{HC}_c(\mathbf{D})$ and $\mathsf{HC}_{\overline{c}}(\mathbf{D})$ must return nonempty boxes included in $\mathbf{D}$ (a canonical box may still be tightened by shrinking any of its nonpunctual dimensions to one of the bounds).

We then have a procedure to determine for each canonical box $\mathbf{D}$ whether it is part of $\mathsf{Inner}(\rho_c)$ or not, namely:

—$\mathsf{HC}_c(\mathbf{D}) = \varnothing \Rightarrow \mathbf{D} \nsubseteq \mathsf{Inner}(\rho_c)$;
—$\mathsf{HC}_{\overline{c}}(\mathbf{D}) = \varnothing \Rightarrow \mathbf{D} \subseteq \mathsf{Inner}(\rho_c)$;
—$(\mathsf{HC}_c(\mathbf{D}) \subseteq \mathbf{D} \wedge \mathsf{HC}_{\overline{c}}(\mathbf{D}) \subseteq \mathbf{D}) \Rightarrow \mathbf{D} \nsubseteq \mathsf{Inner}(\rho_c)$.

By contractance and completeness of HC_c, there are no other possible outcomes (note that all cases are considered mutually exclusive for any nonempty canonical interval).

Now, the preceding rules do not allow us to implement an optimal inner contracting operator in practice for the following reasons (disregarding the fact that it would be grossly inefficient anyway):

—We have restricted our presentation of consistency operators to closed interval arithmetic only. As a consequence, for any constraint of the form $f(x_1, \ldots, x_n) \leq 0$, we cannot consider its negation $f(x_1, \ldots, x_n) > 0$, which would require open interval arithmetic as well. Instead, we will define its negation as the constraint $f(x_1, \ldots, x_n) \geq 0$. As a result, the property "$\mathsf{HC}_{\overline{c}}(\mathbf{D}) \neq \varnothing \Rightarrow \mathbf{D} \nsubseteq \mathsf{Inner}(\rho_c)$" does not hold any more. Consider for example the constraint $c\colon x \geq 0.5$ with $x \in [0.5, 0.5^+]$. We have $\mathsf{HC}_c([0.5, 0.5^+]) = [0.5, 0.5^+]$; considering $\overline{c}$ as the constraint $x \leq 0.5$, we obtain: $\mathsf{HC}_{\overline{c}}([0.5, 0.5^+]) = [0.5, 0.5]$. Applying the rules above, we would conclude that the interval $[0.5, 0.5^+]$ does not belong to $\mathsf{Inner}(\rho_c)$, though it does. It is however important to note that the "relaxed" definition for the negation we are forced to adopt has no effect on the soundness of the algorithms to be described. On the other hand, it will preclude us from devising optimal inner contractors.
—Even if we were using open interval arithmetic, we still would have to face the fact that we are usually not able to implement operators enforcing hull consistency on the constraints given by the user but on primitives obtained after the decomposition process.
—Last, we have seen in Section 2 that the correct rounding of most floating-point arithmetic operators is usually not guaranteed, which precludes us from implementing contracting operators that enforce hull consistency, even for primitive constraints.

Nevertheless, we can still use this principle to implement nonoptimal inner contracting operators based on any kind of outer contracting operator. Table III presents such an implementation. The operator $\mathsf{ICO1}$ is an inner contracting operator for a constraint c parameterized by an outer contracting operator Γ for c.

The operator $\boxbslash$ occurring on Lines 4 and 6 returns the set difference between two boxes as a set of boxes, that is:

$$\forall \mathbf{B_1}, \mathbf{B_2} \in \mathbb{I}^n\colon \mathbf{B_1} \boxbslash \mathbf{B_2} \subseteq \mathcal{P}(\mathbb{I}^n) \quad \text{with} \quad \wr \mathbf{B_1} \boxbslash \mathbf{B_2} \wr = \{r \in \mathbf{B_1} \mid \mathsf{Outer}(\{r\}) \cap \mathbf{B_2} = \varnothing\}.$$

Table III. Inner Contracting Operator for an Atomic Constraint c Based on the Outer Contracting Operator Γ

```
1  $\mathsf{ICO1}_c^{\Gamma}($ in: $\mathbf{B} \in \mathbb{I}^n$; out: $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \in \mathcal{P}(\mathbb{I}^n)^3)$
2  begin
3     $\mathbf{B}' \leftarrow \Gamma_c(\mathbf{B})$
4     $\mathcal{U}_o \leftarrow \mathbf{B} \boxbslash \mathbf{B}'$
5     $\mathbf{B}'' \leftarrow \Gamma_{\overline{c}}(\mathbf{B}')$
6     $\mathcal{U}_i \leftarrow \mathbf{B}' \boxbslash \mathbf{B}''$
7     if StoppingCriterion($\mathbf{B}''$) then
8        return $(\mathcal{U}_i, \mathcal{U}_o, \{\mathbf{B}''\})$
9     else
10       $(\mathbf{B_1}, \dots, \mathbf{B_k}) \leftarrow \mathsf{Split}_k(\mathbf{B}'')$
11       return $(\mathcal{U}_i, \mathcal{U}_o, \varnothing) \uplus \left( \biguplus_{j=1}^{k} \mathsf{ICO1}_c^{\Gamma}(\mathbf{B_j}) \right)$
12    endif
13 end
```

The result is not uniquely defined since there are many ways to represent a set by unions of boxes. This is, however, irrelevant to our purpose.

Proposition 5.7 (Soundness of ICO1). *Given an n-ary constraint c, a box $\mathbf{B} \in \mathbb{I}^n$, an outer contracting operator Γ for c, and $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) = \mathsf{ICO1}_c^{\Gamma}(\mathbf{B})$, we have*

$$\begin{cases} \forall \mathbf{D} \in \mathcal{U}_o\colon & \mathbf{D} \cap \rho_c = \varnothing, \\ \forall \mathbf{D} \in \mathcal{U}_i\colon & \mathbf{D} \subseteq \rho_c. \end{cases}$$

Proof. Consider Line 3 *of* ICO1. By completeness of Γ, $\mathbf{B} \cap \rho_c = \mathbf{B}' \cap \rho_c$. Consequently, $\wr \mathbf{B} \boxbslash \mathbf{B}' \wr \cap \rho_c = \varnothing$. Since the set $\mathcal{U}_o$ returned eventually is the union of all the sets computed on Line 4, we have that $\mathcal{U}_o \cap \rho_c = \varnothing$. We can use the dual reasoning to prove that $\forall \mathbf{D} \in \mathcal{U}_i \colon \mathbf{D} \subseteq \rho_c$. □

5.2 Solving the Unidimensional Guaranteed Tuning Problem

We now consider the Unidimensional Guaranteed Tuning Problem (UGTP): given a set of nonlinear inequalities $c_1, \dots, c_m$ on n variables $\{x_1, \dots, x_{n-1}, v\}$, a box $\mathbf{B} \in \mathbb{I}^n$, and a domain J, compute the set $\mathcal{I}$ defined by

$$\mathcal{I} = \{\mathbf{r} \in \mathbf{B} \mid \forall v \in J \colon c_1 \wedge \cdots \wedge c_m\}. \tag{2}$$

If we restrict ourselves to the case $m = 1$, it is easy to devise an algorithm implementing an inner contracting operator computing a subset of $\mathcal{I}$ along the same lines as ICO1: given the constraint $\forall v \in J \colon c$, use ICO1 to compute an inner approximation of ρ_c while retaining only the boxes for which the domain of v is equal to J. The modified algorithm is presented in Table IV. The algorithm ICO2 is parameterized by an outer contracting operator Γ for c, by the variable v that is universally quantified, and by the domain J on which v must range.

The operator $\mathsf{Dom}_{\mathbf{B}}(v)$ occurring on Line 4 returns the interval in box $\mathbf{B}$ that represents the domain of v. Note that after applying Γ_c on Line 3, we check that the domain of v has not been tightened. Otherwise, there would exist some value in the domain of v such that c does not hold. The box $\mathbf{B}$ would then have

Table IV. Inner Contracting Operator for a Constraint $\forall v \in J\colon c$

```
1 $\mathsf{ICO2}_c^{\Gamma,(v,J)}$( $\mathbf{in}\colon \mathbf{B} \in \mathbb{I}^n$;  $\mathbf{out}\colon (\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \in \mathcal{P}(\mathbb{I}^n)^3$)
2 begin
3    $\mathbf{B}' \leftarrow \Gamma_c(\mathbf{B})$
4    if $\mathsf{Dom}_{\mathbf{B}'}(v) = \mathsf{Dom}_{\mathbf{B}}(v)$ then
5       $\mathcal{U}_o \leftarrow \mathbf{B} \boxbslash \mathbf{B}'$
6       $\mathbf{B}'' \leftarrow \Gamma_{\overline{c}}(\mathbf{B}')$
7       $\mathcal{U}_i \leftarrow \mathbf{B}'_{\mathsf{Dom}_{\mathbf{B}'}(v) \leftarrow J} \boxbslash \mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v) \leftarrow J}$
8       if StoppingCriterion($\mathbf{B}''$) then
9          return $(\mathcal{U}_i, \mathcal{U}_o, \{\mathbf{B}''\})$
10      else
11         $(\mathbf{B_1}, \ldots, \mathbf{B_k}) \leftarrow \mathsf{Split}_k^{\setminus v}(\mathbf{B}'')$
12         $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \leftarrow (\mathcal{U}_i, \mathcal{U}_o, \varnothing) \uplus \left( \biguplus_{j=1}^{k} \mathsf{ICO2}_c^{\Gamma,(v,J)}(\mathbf{B_j}) \right)$
13         return $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u)$
14      endif
15   else
16      return $(\varnothing, \{\mathbf{B}\}, \varnothing)$
17   endif
18 end
```

to be discarded. The domain of v may however be tightened on Line 6 from some domain J_a to some domain J_b by the outer contracting operator for $\overline{c}$. This corresponds to having proved that $\forall x_1 \in I'_1 \setminus I''_1 \cdots \forall x_{n-1} \in I'_{n-1} \setminus I''_{n-1} \forall v \in J_a \setminus J_b\colon c$. It is then sufficient to prove $\forall v \in J_b\colon c$ in the next steps of the algorithm.

In the following, given an n-ary constraint c, an interval J, and a variable v, let us note $\rho_{c^{\forall}}$ the n-ary relation[2] associated to the constraint $\forall v \in J\colon c$.

We first state a simple lemma, which will be used in proving the proposition to follow.

LEMMA 5.8. *Given $n \in \mathbb{N}$, and two boxes $\mathbf{B} = I_1 \times \cdots \times I_n$ and $\mathbf{D} = J_1 \times \cdots \times J_n$ such that $\mathbf{B} \subseteq \mathbf{D}$, the following does hold:*

$$\mathbf{D} \setminus \mathbf{B} = \bigcup_{k=1}^{n} J_1 \times \cdots \times J_{k-1} \times (J_k \setminus I_k) \times J_{k+1} \times \cdots \times J_n.$$

In addition:

$$\wr \mathbf{D} \boxbslash \mathbf{B} \wr = \bigcup_{k=1}^{n} J_1 \times \cdots \times J_{k-1} \times \wr J_k \boxbslash I_k \wr \times J_{k+1} \times \cdots \times J_n.$$

PROOF. Immediate. □

PROPOSITION 5.9 (SOUNDNESS OF ICO2). *Let $c(x_1, \ldots, x_{n-1}, v)$ be an n-ary constraint, $\mathbf{B} = I_1 \times \cdots \times I_{n-1} \times J$ a box, Γ an outer contracting operator for c, and*

[2]For convenience, the $(n-1)$-ary relation associated to $\forall v \in J\colon c$ is extended to an n-ary relation by using the domain J for the extra dimension.

$(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) = \mathsf{ICO2}_c^{\Gamma,(v,J)}(\mathbf{B})$. *The following properties do hold:*

$$\begin{cases} \forall \mathbf{D} \in \mathcal{U}_o\colon & \mathbf{D} \cap \rho_{c^\forall} = \varnothing, \\ \forall \mathbf{D} \in \mathcal{U}_i\colon & \mathbf{D} \subseteq \rho_{c^\forall}. \end{cases} \tag{3}$$

Proof. Termination of Algorithm ICO2 depends on a reasonable choice for the functions StoppingCriterion() and $\mathsf{Split}_k^{\backslash v}()$. More specifically, termination is ensured whenever $\mathsf{Split}_k^{\backslash v}()$ creates boxes $\mathbf{B_1}, \ldots, \mathbf{B_k}$ that are all strictly smaller than $\mathbf{B}''$, and when StoppingCriterion() checks for the size of all dimensions of $\mathbf{B}''$ but the one of v being smaller than some threshold that is attainable with the floating-point format at hand.

In order to prove the proposition, we have to show that the properties given by Equation (3) always hold on Lines 9 and 16 in Table IV. Provided that boxes put into $\mathcal{U}_o$ and $\mathcal{U}_i$ on Lines 5 and 7, respectively, verify Equation (3), it is not necessary to investigate what is returned on Line 13 since it is the mere union of the sets previously computed.

—We first consider the case when $\mathsf{Dom}_{\mathbf{B}'}(v) \neq \mathsf{Dom}_{\mathbf{B}}(v)$. Let $\mathbf{B} = I_1 \times \cdots \times I_n$. By hypothesis, $\mathsf{Dom}_{\mathbf{B}}(v) = I_n = J$ during the first call of ICO2. By completeness of Γ_c, if $\mathsf{Dom}_{\mathbf{B}'}(v) \subsetneq \mathsf{Dom}_{\mathbf{B}}(v)$, there exists a value in $\mathsf{Dom}_{\mathbf{B}}(v) \setminus \mathsf{Dom}_{\mathbf{B}'}(v)$ such that c does not hold when the free variables take their values in $I_1 \times \cdots \times I_{n-1}$. Consequently, $\mathbf{B} \cap \rho_{c^\forall} = \varnothing$. The property does hold for all the subsequent calls of ICO2 since the corresponding domains for v are always included in J (by contractance of the narrowing operators used). Since $\varnothing \subseteq \rho_{c^\forall}$, we have proved that the triplet returned on Line 16 verifies Equation (3) for any call of ICO2.

—We now consider the case when $\mathsf{Dom}_{\mathbf{B}'}(v) = \mathsf{Dom}_{\mathbf{B}}(v)$.

Let us prove that for any call l of ICO2, the assignments performed on Lines 5 and 7 are such that

$$\begin{cases} \wr\mathcal{U}_o^{(l)}\wr \cap \rho_{c^\forall} = \varnothing, \\ \wr\mathcal{U}_i^{(l)}\wr \subseteq \rho_{c^\forall}. \end{cases}$$

This suffices to prove the proposition since what is returned for the case $\mathsf{Dom}_{\mathbf{B}'}(v) = \mathsf{Dom}_{\mathbf{B}}(v)$ is either $\mathcal{U}_o^{(l)}$ and $\mathcal{U}_i^{(l)}$, or the union, componentwise, of $\mathcal{U}_o^{(l)}$ and $\mathcal{U}_i^{(l)}$ with sets of boxes $\mathcal{U}_o^{(m)}$ and $\mathcal{U}_i^{(m)}$ $(m > l)$ computed in subsequent calls, and since we have already proved that what is returned whenever $\mathsf{Dom}_{\mathbf{B}'}(v) \neq \mathsf{Dom}_{\mathbf{B}}(v)$ verifies the property.

We first consider Line 5: by completeness of Γ_c, $\wr\mathbf{B} \boxbslash \mathbf{B}'\wr \subseteq \rho_{\overline{c}}$. Hence, $\wr\mathcal{U}_o\wr \subseteq \rho_{\overline{c}}$, and then $\wr\mathcal{U}_o\wr \cap \rho_{c^\forall} = \varnothing$.

We now consider Line 7: we will first prove that for any call l of ICO2

$$\wr\mathbf{B}^{(l)}_{\mathsf{Dom}_{\mathbf{B}^{(l)}}(v) \leftarrow J} \boxbslash \mathbf{B}'^{(l)}\wr \subseteq \rho_c. \tag{4}$$

This is trivially true for the first call $(l = 1)$ since, by hypothesis, $\mathsf{Dom}_{\mathbf{B}^{(1)}}(v) = J$.

Assume that Equation (4) does hold for the lth call of ICO2. Since $\mathbf{B}' \subseteq \mathbf{B}$ and $\mathsf{Dom}_{\mathbf{B}'^{(l)}}(v) = \mathsf{Dom}_{\mathbf{B}^{(l)}}(v)$, we have

$$\wr\mathbf{B}'^{(l)}_{\mathsf{Dom}_{\mathbf{B}'^{(l)}}(v) \leftarrow J} \boxbslash \mathbf{B}'^{(l)}\wr \subseteq \rho_c. \tag{5}$$

By completeness of $\rho_{\overline{c}}$, we have

$$\lbag \mathbf{B}''^{(l)}_{\mathsf{Dom}_{\mathbf{B}''^{(l)}}(v)\leftarrow\mathsf{Dom}_{\mathbf{B}'^{(l)}}(v)} \boxslash \mathbf{B}''^{(l)} \rbag \subseteq \rho_c. \tag{6}$$

By contractance of $\Gamma_{\overline{c}}$, $\mathbf{B}''^{(l)} \subseteq \mathbf{B}'^{(l)}$. From Equations 6 and 5, we deduce

$$\lbag \mathbf{B}''^{(l)}_{\mathsf{Dom}_{\mathbf{B}''^{(l)}}(v)\leftarrow J} \boxslash \mathbf{B}''^{(l)} \rbag \subseteq \rho_c. \tag{7}$$

The box $\mathbf{B}''^{(l)}$ is split on Line 11 in k boxes $\mathbf{B_i}^{(l)}$ ($i \in \{1,\ldots,k\}$), with $\mathsf{Dom}_{\mathbf{B_i}^{(l)}}(v) = \mathsf{Dom}_{\mathbf{B}''^{(l)}}(v)$ (no splitting on the domain of v).

Consequently, we obtain from Equation 7:

$$\forall i \in \{1,\ldots,k\}: \quad \lbag \mathbf{B_i}^{(l)}_{\mathsf{Dom}_{\mathbf{B_i}^{(l)}}(v)\leftarrow J} \boxslash \mathbf{B_i}^{(l)} \rbag \subseteq \rho_c.$$

From Line 12, it follows that Equation (4) does hold for $l+1,\ldots,l+k$ whenever it does hold for l.

Having proved that Equation (4) does hold for any call l, we reconsider Line 7: by completeness of $\Gamma_{\overline{c}}$, it comes

$$\lbag \mathbf{B}' \boxslash \mathbf{B}'' \rbag \subseteq \rho_c.$$

Hence:

$$\lbag \mathbf{B}' \boxslash \mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v)\leftarrow J} \rbag \subseteq \rho_c \tag{8}$$

since $\mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v)\leftarrow J} \supseteq \mathbf{B}''$.

From Equation (5) and (8), we deduce

$$\lbag \mathbf{B}'_{\mathsf{Dom}_{\mathbf{B}'}(v)\leftarrow J} \boxslash \mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v)\leftarrow J} \rbag \subseteq \rho_c. \tag{9}$$

We also know from Lemma 5.8 that all the boxes in $\mathbf{B}'_{\mathsf{Dom}_{\mathbf{B}'}(v)\leftarrow J} \boxslash \mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v)\leftarrow J}$ have J as their last dimension. From this and Equation (9), we obtain

$$\lbag \mathbf{B}'_{\mathsf{Dom}_{\mathbf{B}'}(v)\leftarrow J} \boxslash \mathbf{B}''_{\mathsf{Dom}_{\mathbf{B}''}(v)\leftarrow J} \rbag \subseteq \rho_{c^{\forall}}.$$

That is, $\lbag \mathcal{U}_i \rbag \subseteq \rho_{c^{\forall}}$ for any call of ICO2. □

Handling the Unidimensional Guaranteed Tuning Problem described by Equation (2) is done by the propagation algorithm IPA presented in Table V as follows: each constraint of the system is considered in turn together with the sets of elements verifying all the constraints already considered; the main point concerning IPA lies in that *each constraint needs only be taken into account once*, since after having been considered for the first time, the elements remaining in the variable domains are all solutions of the constraint. As a consequence, narrowing some domain later does not require additional work. Algorithm IPA is parameterized by an inner contracting operator Υ. Solving the UGTP can be done by instantiating IPA with ICO2. Using ICO1 instead leads to an algorithm computing the inner approximation of the relation associated to the conjunction of constraints in $\mathcal{S}$.

PROPOSITION 5.10 (SOUNDNESS OF ALGORITHM IPA). *Given a set of constraints $\mathcal{S} = \{c_1,\ldots,c_m\}$, a box $\mathbf{B} \in \mathbb{I}^n$, an inner contracting operator Υ, a variable v, an interval J, a set of outer contracting operators $\{\Gamma_1,\ldots,\Gamma_m\}$ for, respectively, $c_1, \ldots, c_m$, and $(\mathcal{U}_i,\mathcal{U}_o,\mathcal{U}_u) = \mathsf{IPA}^{\Upsilon}(\mathcal{S},\mathbf{B})$, let us note $\rho_{\mathcal{S}}$, the relation associated*

Table V. Inner Propagation Algorithm for a Set of Constraints $\mathcal{S}$ Based on the Inner Contracting Operator Υ

```
1 $\mathsf{IPA}^{\Upsilon}$( in: a set of constraints $\mathcal{S}, \mathbf{B} \in \mathbb{I}^n$;  out: $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \in \mathcal{P}(\mathbb{I}^n)^3$)
2 begin
3     $\mathcal{T} \leftarrow \{\mathbf{B}\}$
4     $(\mathcal{U}_o, \mathcal{U}_u) \leftarrow (\varnothing, \varnothing)$
5     foreach $c \in \mathcal{S}$ do
6         $\mathcal{U}_i \leftarrow \varnothing$
7         foreach $\mathbf{D} \in \mathcal{T}$ do
8             $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \leftarrow (\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u) \uplus \Upsilon_c(\mathbf{D})$
9         end
10        $\mathcal{T} \leftarrow \mathcal{U}_i$
11    end
12    return $(\mathcal{U}_i, \mathcal{U}_o, \mathcal{U}_u)$
12 end
```

to the constraint $c_1 \wedge \cdots \wedge c_m$, and $\rho_{\mathcal{S}^\forall}$, the relation associated to the constraint $\forall v \in J : c_1 \wedge \cdots \wedge c_m$. We have

$$\text{If } \Upsilon_{c_i} = \mathsf{ICO1}_{c_i}^{\Gamma_i}: \quad \begin{cases} \forall \mathbf{D} \in \mathcal{U}_o\colon\ \mathbf{D} \cap \rho_{\mathcal{S}} = \varnothing, \\ \forall \mathbf{D} \in \mathcal{U}_i\colon\ \mathbf{D} \subseteq \rho_{\mathcal{S}}, \end{cases}$$

$$\text{If } \Upsilon_{c_i} = \mathsf{ICO2}_{c_i}^{\Gamma_i,(v,J)}: \quad \begin{cases} \forall \mathbf{D} \in \mathcal{U}_o\colon\ \mathbf{D} \cap \rho_{\mathcal{S}^\forall} = \varnothing, \\ \forall \mathbf{D} \in \mathcal{U}_i\colon\ \mathbf{D} \subseteq \rho_{\mathcal{S}^\forall}. \end{cases}$$

PROOF. For each constraint c, we consider only the boxes that were put in $\mathcal{U}_i$ by the preceding constraint (Lines 7 and 10). By soundness of ICO1 (respectively ICO2), when considering constraint c_j, $\mathcal{U}_i$ then contains on Line 10 only the boxes $\mathbf{D}$ verifying $\mathbf{D} \subseteq \rho_{c_1} \wedge \cdots \wedge \mathbf{D} \subseteq \rho_{c_j}$ (respectively $\mathbf{D} \subseteq \rho_{c_1^\forall} \wedge \cdots \wedge \mathbf{D} \subseteq \rho_{c_j^\forall}$).

Once again, by soundness of ICO1 (respectively ICO2), when considering a constraint c_j on Line 5, $\mathcal{U}_o$ contains only boxes $\mathbf{D}$ such that there was a constraint c_k with $k < j$ for which $\mathbf{D} \cap \rho_{c_k} = \varnothing$ (respectively $\mathbf{D} \cap \rho_{c_k^\forall} = \varnothing$). Consequently, $\mathbf{D} \cap (\rho_{c_1} \cap \cdots \cap \rho_{c_k} \cap \cdots \wedge \rho_{c_j}) = \varnothing)$ (respectively $\mathbf{D} \cap (\rho_{c_1^\forall} \cap \cdots \cap \rho_{c_k^\forall} \cap \cdots \wedge \rho_{c_j^\forall}) = \varnothing$). □

Note that it is straightforward to modify IPA in order to be able to solve constraints of the form $\forall v \in J^{(1)} : c_1 \wedge \cdots \wedge \forall v \in J^{(m)} : c_m$ by replacing the set of constraints $\mathcal{S}$ by a set of pairs $(c_i, J^{(i)})$ and by initializing accordingly the box passed to Υ_c on Line 8. Since we have not encountered applications requiring such extension, we will not consider it any more in the following.

6. CAMERA CONTROL AND THE VIRTUAL CAMERAMAN PROBLEM

Camera control is of interest to many fields, from computer graphics (visualization in virtual environments [Blinn 1988]) to robotics (sensor planning [Abrams and Allen 1997]), and cinematography [Davenport et al. 1991]. Whatever the activity, the objective is always to provide the user with an adequate view of some points of interest in a scene for a predefined duration. In sensor planning [Tarabanis 1990], for instance, one is interested in positioning a camera over a robot in order to be able to monitor its work whatever the position of its manipulating hand may be. The targeted applications are mostly real-time ones, which implies seeking for only one solution through some optimization process. The modeling adopted is usually very close to the camera representation (involving, for instance, the direct control of the camera parameters through the use of sliders), thereby impeding inexperienced users from predicting the exact behavior of the controlled device.

Yet, some authors [Gleicher and Witkin 1992; Drucker et al. 1992] from the computer graphics and cinematography fields have worked on determining camera parameters from the given properties of a desired scene. Once more, most of these works are concerned with the computation of only one solution by means of an optimization criterion.

Among these works, one may single out the attempt by Gleicher, which permits using some constraints including the position of a three-dimensional point on the screen, and the orientation of two points along with their distance on the projected image. Higher level of control is obtained through the ability to bound a point within a region of the image or to bound the size of an object. Time derivatives of the camera parameters are computed in order to satisfy user-defined controls. The method is devoted to maintaining user-defined constraints while manipulating camera parameters. Camera motion in an animated scene is not treated.

Another approach [Jardillier and Languénou 1998]—inspired by Snyder's [1992] seminal work—to the camera motion computation problem relies on interval arithmetic to take into account multiple constraints and screen space properties. Satisfying camera movement parameters—with respect to some given scene description—are obtained through constraint solving. The constraints involved are handled with Algorithm JLA (see Table II).

The main objective that motivated our work was to build a high-level tool allowing an artist to specify the desired camera movements for a "shot" using *cinematic primitives*. The resulting description is then translated into a constraint system in terms of the camera parameters, and solved using local consistency-based pruning techniques. A huge set of solutions is usually output as a result, from which a challenging task is to extract a limited representative sample for presentation to the user.

The mathematical model used for representing the camera, its motion, and the objects composing a scene is presented in the following section; then follows the description of some of the cinematic primitives together with their translation in terms of constraint systems. Addressing the problem of the isolation of a representative solution sample is deferred until Section 7.2.

6.1 Modeling the Camera and the Objects

A camera produces a two-dimensional (2D) image by using a projection transformation of a *three-dimensional (3D) space scene*. In the following, the image is referred to as the *screen space* (or *image space*) and the scene filmed as the *scene space* or *3D space*.

The standard camera model is based on Euler angles to specify its location and orientation. The work presented here is not bound to this representation, though, and any other representation would be convenient as well.

A camera (Figure 4) possesses seven degrees of freedom, namely, its Cartesian position in the space, its orientation, and its focal length:

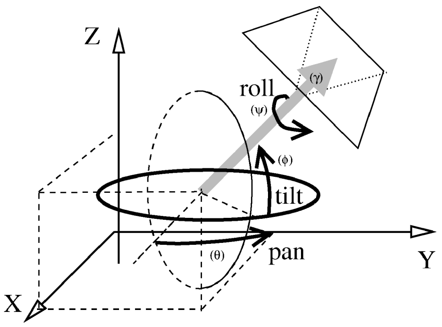


Fig. 4. Camera model.

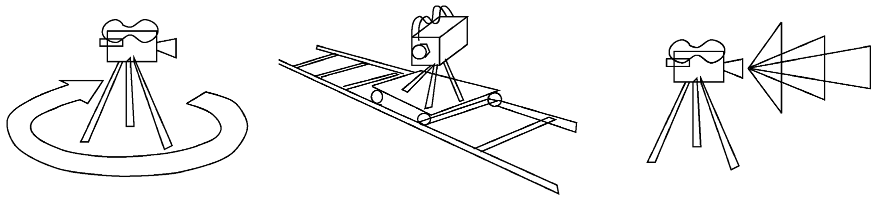


Fig. 5. Camera movements.

—Position. Three scalars: x, y, and z;
—View direction. Three scalars:
  —Pan. θ,
  —Tilt. ϕ,
  —Roll. ψ;
—Focal length. One scalar: γ.

Most movies are made of a large number of short elementary "shots." Therefore, a simple model for camera motion may be adopted without losing too much expressiveness. Sophisticated camera movements are obtained by assembling sequences of shots. (Refer to Christie et al. [2002] for a way to perform this task.) Due to a lack of space, the following description of camera movements (Figure 5) is restricted to primitive ones, and the reader is referred to the excellent book by Arijon [1976] for a thorough presentation of the others:

—*Panoramic shot.* A panoramic shot may be horizontal or vertical (i.e., around vertical or horizontal axis). The camera location is usually constant;
—*Traveling (tracking shot or dolly).* A general term for a camera translation;
—*Zoom in or zoom out.* A variation of the focal length of the camera.

To our knowledge, existing declarative camera movement generators compute camera animation frame by frame [Drucker 1994] or use calculated key frames [Shoemake 1985] (fixed camera location and orientation), and interpolation of the camera parameters in between.

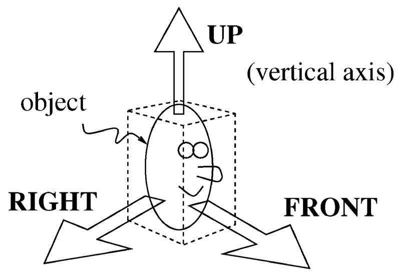


Fig. 6. Object axis and bounding volume.

In contrast, the tool described here is based on a parametric representation. Computing a satisfying solution boils down to determining all the camera parameters such that some constraints on the scene are satisfied. Using the previous remark concerning simple elementary shots, parameters are modeled with degree 3 polynomials whose unknown is time t. For example, an horizontal pan (i.e., a movement along the horizontal orientation angle θ) might be defined as $\theta(t) = c_\theta t + d_\theta$, where d_θ represents the initial horizontal orientation of the camera (at time $t = 0$) and c_θ is a constant velocity. Consequently, given V_θ the maximum allowed pan velocity, one may note that c_θ must lie in the domain $[-V_\theta, V_\theta]$, and then d_θ must lie in the domain $[0, 2\pi]$, in order to make any orientation starting point eligible.

In our view, the scene is considered as a problem data. Hence, from this point onward, every object composing a scene is assumed to have its location, orientation, and movement already set by the user.

Object properties rely on bounding volumes (location) and object axis (orientation): each object is bounded by a volume called a *bounding box*. Compounds' bounding box is the smallest box containing the bounding boxes of all the objects involved. In addition, bounding boxes may be associated with any set of objects (like a group of characters). Many geometric modellers provide such a hierarchy of bounding volumes.

The orientation of any object on the image is determined by three vectors: **Front**, **Up**, and **Right** (Figure 6).

6.2 Properties and Constraints

The translation of declarative descriptions of scenes into constraints is now presented. Three kinds of desired properties may be distinguished:

(1) Properties on the camera;
(2) properties on the object screen location;
(3) properties on the object screen orientation.

Camera properties are a means to set constraints on the camera motion. Translation of properties on the objects of a scene is described below. The notion of *frame* is first introduced as an aid to constrain an object location on the screen: a *frame* is a rectangle whose borders are parallel to the screen borders. A frame may be inside, outside, or partially inside the screen, though it is

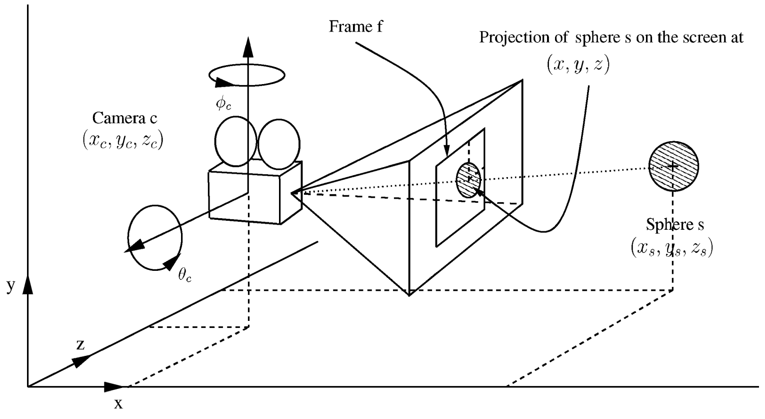


Fig. 7. Projection in a frame.

usually fully contained in the screen. For special purposes, the frame size or/and location may be modified during the animation.

With frames, an artist can define the precise projection zone of an object from the scene (3D) to the screen (2D). Our belief in a *creation helping tool* leads us to prefer this kind of soft constraint to the exact 2D screen location of the projection of a 3D point.

The user defines frames (interactively or off-line), then chooses properties to apply to a frame and an object. For example, the statement (Figure 7):

> *"The sphere $\mathfrak{s}$ with center $(x_{\mathfrak{s}}(t), y_{\mathfrak{s}}(t), z_{\mathfrak{s}}(t))$ and radius r must be fully included in the frame $\mathfrak{f}$ defined by the bottom-left point $(x_{\mathfrak{f}}^1(t), y_{\mathfrak{f}}^1(t))$ and the top-right point $(x_{\mathfrak{f}}^2(t), y_{\mathfrak{f}}^2(t))$ during the 20 seconds of the shot filmed by the camera $\mathfrak{c}$ located at $(x_{\mathfrak{c}}(t), y_{\mathfrak{c}}(t), z_{\mathfrak{c}}(t))$ with orientation $\theta_{\mathfrak{c}}(t)$ and $\phi_{\mathfrak{c}}(t)$."*

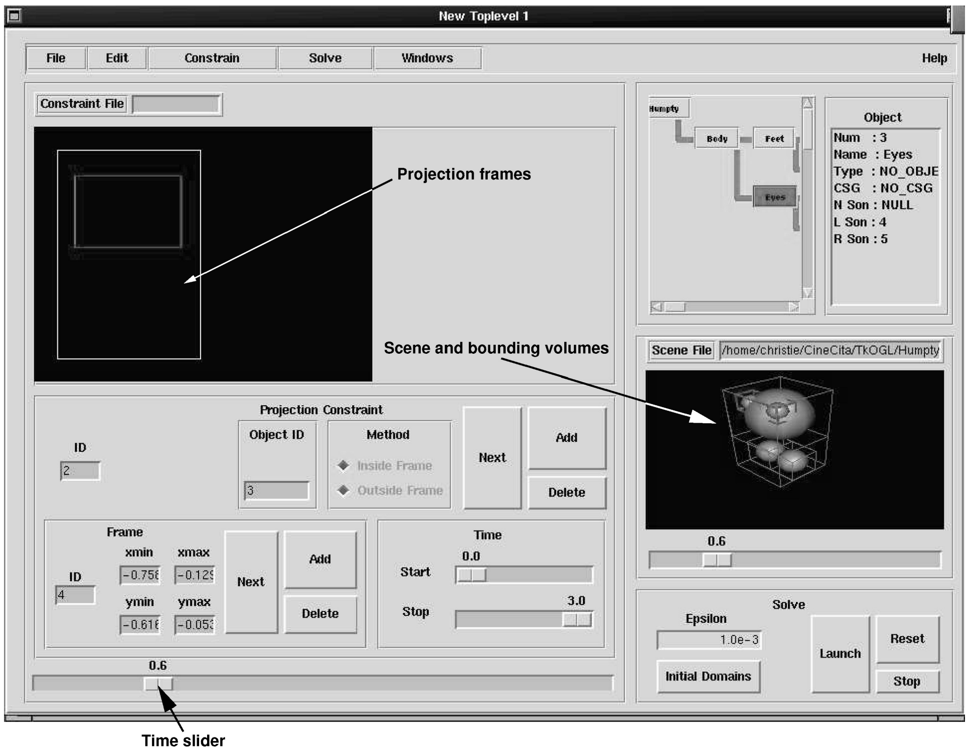
is translated into the nonlinear constraint system of equations and inequations

$$\forall t \in [0, 20]: \begin{cases} x_f^1(t) & \leq (x(t) + r)/(z(t)/\gamma_c(t)), \\ x_f^2(t) & \geq (x(t) - r)/(z(t)/\gamma_c(t)), \\ y_f^1(t) & \leq (y(t) + r)/(z(t)/\gamma_c(t)), \\ y_f^2(t) & \geq (y(t) - r)/(z(t)/\gamma_c(t)), \end{cases}$$

with

$$\begin{cases} x(t) & = -(x_{\mathfrak{s}}(t) - x_{\mathfrak{c}}(t)) \sin\theta_{\mathfrak{c}}(t) + (y_{\mathfrak{s}}(t) - y_{\mathfrak{c}}(t)) \cos\theta_{\mathfrak{c}}(t), \\ y(t) & = -(x_{\mathfrak{s}}(t) - x_{\mathfrak{c}}(t)) \cos\theta_{\mathfrak{c}}(t) \sin\phi_{\mathfrak{c}}(t) \\ & \quad + (y_{\mathfrak{s}}(t) - y_{\mathfrak{c}}(t)) \sin\phi_{\mathfrak{c}}(t) \sin\theta_{\mathfrak{c}}(t) + (z_{\mathfrak{s}}(t) - z_{\mathfrak{c}}(t)) \cos\phi_{\mathfrak{c}}(t), \\ z(t) & = -(x_{\mathfrak{s}}(t) - x_{\mathfrak{c}}(t)) \cos\theta_{\mathfrak{c}}(t) \cos\phi_{\mathfrak{c}}(t) \\ & \quad + (y_{\mathfrak{s}}(t) - y_{\mathfrak{c}}(t)) \sin\theta_{\mathfrak{c}}(t) \cos\phi_{\mathfrak{c}}(t) + (z_{\mathfrak{s}}(t) - z_{\mathfrak{c}}(t)) \sin\phi_{\mathfrak{c}}(t), \end{cases}$$

where $(x(t), y(t), z(t))$ is the projection of the center of $\mathfrak{s}$ in the screen space at time t.



```
Num   : 3
Name  : Eyes
Type  : NO_OBJE
CSG   : NO_CSG
N Son : NULL
L Son : 4
R Son : 5
```

Scene File: /home/christie/CineCita/TkOGL/Humpty

Projection Constraint — ID: 2; Object ID: 3

Frame — ID: 4; xmin: -0.758; xmax: -0.129; ymin: -0.618; ymax: -0.053

Time — Start: 0.0; Stop: 3.0

Epsilon: 1.0e-3

Sliders: 0.6

Fig. 8. The declarative modeler tool.

Note that, in order to fall back to an instance of the guaranteed tuning problem, it is possible to discard the three equations by replacing $x(t)$, $y(t)$, and $z(t)$ in the inequalities by the corresponding right-hand side.

The constraints resulting from the translation of the declarative description of "shots" contain occurrences of the universally quantified time variable t.

7. EXAMPLES AND BENCHMARKING

A high-level declarative modeler tool for camera motion has been devised in order to validate the algorithms presented in Section 5. The prototype is written in C++ and Tcl/Tk; Figure 8 presents its graphical user interface: the animated scene to be filmed is displayed in a window together with the bounding volumes, while another window contains some previously drawn projection frames. The user constructs frames, selects objects, and assigns properties to objects in the scene. The output is a set of satisfying camera paths, and corresponding animations are shown in an output window.

In the next section, we present some problems used to assess the quality of Algorithm IPA^{Υ} (Table V) using Algorithm $\mathsf{ICO2}^{\mathsf{BC3r}}$ (Table IV and Definition 4.5) as the inner contracting operator parameter Υ (hereafter referred as IPABC). Comparisons with Algorithm JLA (Table II) used by Jardillier and Languénou [1998] for the same kind of applications are produced

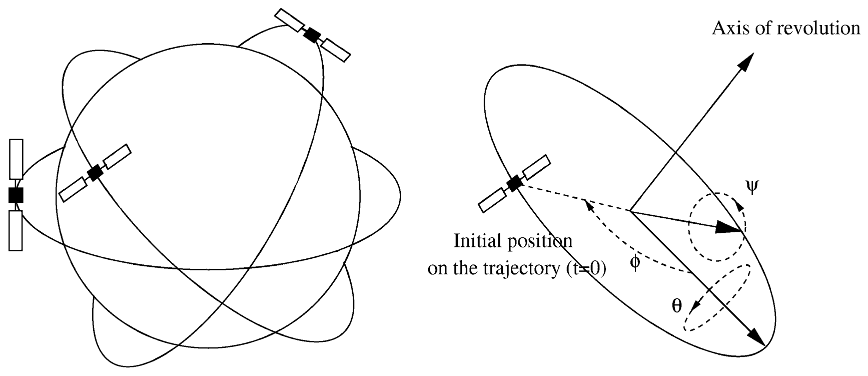


Fig. 9. A collision-free problem.

and commented. Some techniques to speed up the computation and improve the representativeness of the boxes output are also described.

7.1 Description of the Benchmarks

Parabola: a curve fitting problem. This simple benchmark [Jardillier and Languénou 1998] corresponds to finding all the parabolas lying above a given line:

$$\forall t \in [0,2]\colon at^2 + bt + c \geq 2t - 1 \quad \text{with} \quad a \in [0,1], b \in [0,1], c \in [0,1].$$

Circle: a trivial collision problem. Benchmark *Circle* is a collision problem: given B a point moving along a circling path, find all points A such that the distance between A and B is always greater than a given value. Benchmarks *Circle*$_2$ and *Circle*$_3$ are instances of the same problem with, respectively, two and three points moving round in circles. For only one circling point, we have

$$\begin{array}{l}\forall t \in [-\pi,\pi]\colon \\ \quad \sqrt{(r_1 \sin t - x)^2 + (r_1 \cos t - y)^2} \geq d_1\end{array} \quad \text{with} \quad \begin{cases} x \in [-5,5], \\ y \in [-5,5], \\ d_1 = 0.5, \end{cases}$$

where d_1 is the mandatory minimal distance between A and B, and $r_1 = 2.5$ is the radius of B's circling path.

Satellite: a collision problem. Given n satellites swiveling around a planet (Figure 9), we are looking for the parameters of an $(n+1)$th trajectory on which to put another satellite. Obviously, this trajectory must be such that the added satellite never collides with the already launched ones.

The position of the ith satellite at time t is defined by

$$\mathbf{f_i}(t) = \begin{pmatrix} x_i(t) \\ y_i(t) \\ z_i(t) \end{pmatrix} = \begin{pmatrix} d_i \cos\theta_i \sin\omega_i t + \phi_i \\ d_i(\sin\psi_i \sin\theta_i \sin(\omega_i t + \phi_i) + \cos\psi_i \cos(\omega_i t + \phi_i)) \\ d_i(-\cos\psi_i \sin\theta_i \sin(\omega_i t + \phi_i) + \sin\psi_i \cos(\omega_i t + \phi_i)) \end{pmatrix},$$

where variables θ_i and ϕ_i define the orientation of the plane of revolution, ω_i the angular velocity of the satellite, and ϕ_i its initial position at $t = 0$. Variable d_i stands for the radius of the circle of revolution. Considering n satellites, we are looking for consistent values of the unknowns $\theta_j, \phi_j, \omega_j, d_j$ of the satellite $j = n+1$ such that

$$\forall t \in [-\pi, \pi]: \begin{cases} \mathrm{dist}(\mathbf{f_1}(t), \mathbf{f_j}(t)) \geq s \\ \mathrm{dist}(\mathbf{f_2}(t), \mathbf{f_j}(t)) \geq s \\ \vdots \\ \mathrm{dist}(\mathbf{f_n}(t), \mathbf{f_j}(t)) \geq s \end{cases},$$

where s represents the minimal distance allowed between two satellites.

The unknowns to be computed are θ_j, ϕ_j, and ψ_j, with domains $[0, 2\pi]$. In this benchmark, we consider three satellites in the air with the following parameters:

Parameter	Satellite 1	Satellite 2	Satellite 3
d_i	5.0	5.0	5.0
ω_i	1.0	1.0	1.0
ϕ_i	0.0	1.0	2.0
θ_i	0.0	1.0	1.5
ψ_i	0.0	1.0	1.5

Robot: a collision problem. This benchmark is based on Example 1.1. The actual constraint system to solve is

$$\forall t \in [0, 2]: \sqrt{\left(x - P_x(t)\right)^2 + \left(y - P_y(t)\right)^2} \geq d,$$

where

$$\begin{cases} P_x(t) &= d_1 \sin \alpha_1(t) + d_2 \sin(\alpha_1(t) + \alpha_2(t) - \pi) + d_3 \sin(\alpha_1(t) + \alpha_2(t) + \alpha_3(t)), \\ P_y(t) &= d_1 \cos \alpha_1(t) + d_2 \cos(\alpha_1(t) + \alpha_2(t) - \pi) + d_3 \cos(\alpha_1(t) + \alpha_2(t) + \alpha_3(t)), \\ \alpha_1(t) &= t + \pi/4, \\ \alpha_2(t) &= 2t - 1, \\ \alpha_3(t) &= 0.2t + 0.1. \end{cases}$$

The initial domains for the unknowns x and y are both set to $[0, 5]$, the size of the robot's hand is set to 0.5, and the respective lengths of the arm's segments are $d_1 = 1.0$, $d_2 = 2.0$, and $d_3 = 1.0$.

PointPath: a motion planning problem. This benchmark [Jaulin and Walter 1996] introduces a simple motion planning problem (Figure 10). We need to compute an object's path, starting at position M_0 and ending at position M_1, while avoiding collision with the ground and some objects. The path $M(t)$ is tentatively chosen as a polynomial written as a linear combination of Bernstein polynomials of degree 3, and controlled by the points P_1 and P_2 of unknown coordinates:

$$M(t) = M_0 B_0^3(t) + P_1 B_1^3(t) + P_2 B_2^3(t) + M_1 B_3^3(t),$$

where the Bernstein polynomials are

$$B_0^3(t) = (1-t)^3,\ B_1^3(t) = 3t(1-t)^2,\ B_2^3(t) = 3t^2(1-t),\ B_3^3 = t^3.$$

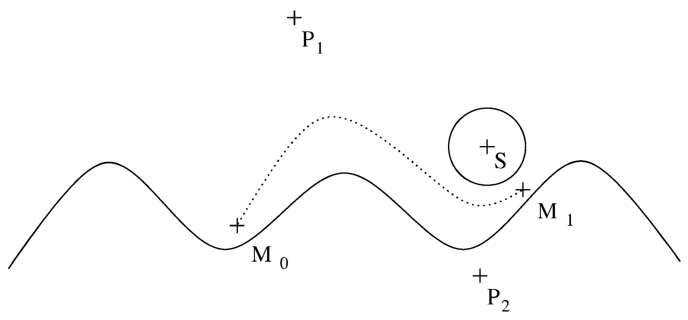


Fig. 10. Solution for the motion planning problem.

The constraints specify that for all t in the time frame:

(1) $M(t)$ must be above a curve representing the floor;
(2) the distance between $M(t)$ and a static object $S = (4.8, 1.0)$ must be greater than 1.

Which leads to

$$\forall t \in [0,1]: \begin{cases} (x(t) - 4.8)^2 + (y(t) - 1)^2 \geq 1 \\ y(t) \geq \sin(x(t)) \end{cases}$$

with $M(t) = \binom{x(t)}{y(t)}$. Domains for the coordinates of P_1 and P_2 are initialized to $[-10, 10]$.

Projection: a camera control problem. Benchmark *Projection* corresponds to the problem of projecting a moving sphere in a moving frame, already presented in Section 6.2. The initial values chosen for the camera are the following: $x_c \in [-3, 3]$, $y_c \in [-3, 3]$, $z_c = 2$, $\phi_c \in [-0.5, 0.5]$, $\theta_c = 0$, and $\gamma_c = 0.8$.

Projection$_4$ is the original problem presented in Section 6.2; *Projection*$_5$ is the same problem with one more constraint on the distance between the sphere and the camera; *Projection*$_8$ is the problem with two frames and two spheres.

GarloffGraf 1: *inner approximation computation* [*Garloff and Graf* 1999]. Find the values for the parameters v and w ensuring the stability of the polynomial:

$$p(s) = s^3 + vs^2 + (w - 5v - 13)s + w.$$

Using the *Linard-Chipart criterion*, the problem can be transformed into the equivalent system:

$$\begin{aligned} v, w &> 0, \\ -5v^2 - 13v + vw - w &> 0. \end{aligned}$$

Following Garloff and Graf, we compute the inner approximation of the relation $-5v^2 - 13v + vw - w > 0$ for $v \in [2, 10]$ and $w \in [40, 50]$.

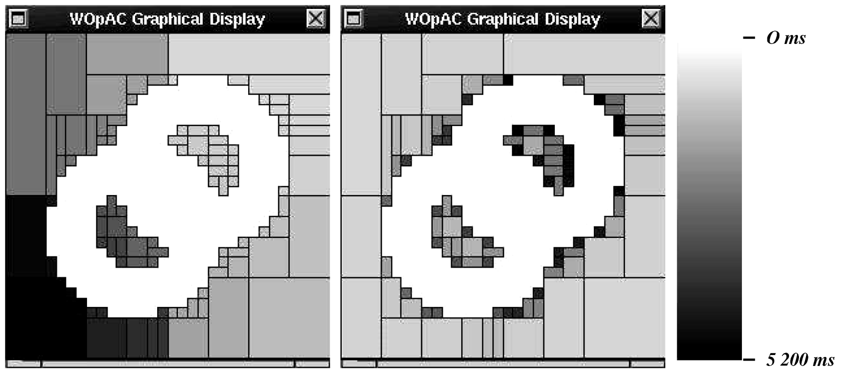


Fig. 11. Depth-first versus semi-depth-first.

GarloffGraf 2: *inner approximation computation* [*Garloff and Graf* 1999]. This is again a stability problem: find sound values for A, B, and D verifying

$$\begin{aligned}A, B, D &> 0,\\ AB^2 - D^2 &> 0,\\ -AB + A + D^2 - D - 1 &> 0,\\ AB - AD - 2A + D^3 + 4D^2 + 4D &> 0,\\ AB^3 - AB^2D - 4AB^2 + 2ABD + 4AB + 2BD^3 + 5BD^2 + 2BD - D^3 - 4D^2 - 4D &> 0,\\ AB - 2A - BD^2 - 4BD - 4B - 2D^2 + 3D - 2 &> 0.\end{aligned}$$

Following Garloff and Graf, the initial domains chosen are $A \in [100, 120]$, $B \in [0, 2]$, and $D \in [10, 20]$.

According to Abdallah et al. [1996], a CAD-based software such as QEPCAD (`http://www.cs.usna.edu/~qepcad/B/QEPCAD.html`) requires more than 2 h to prove the existence of a solution to the problem.

7.2 Improving Computation

Solvers such as Numerica usually isolate solutions with variable domains around 10^{-8} or 10^{-16} in width. By contrast, the applications this article focuses on are less demanding since the resulting variable domains are used in the context of a display screen, a "low-resolution" device. In practice, one can consider that a reasonable threshold ε for the splitting process during the search is some value lower or equal to 10^{-2} or 10^{-3}.

One of the drawbacks of Algorithm JLA [Jardillier and Languénou 1998] is that successive output solutions are very similar, while it is of importance to be able to provide the user with a representative sample of solutions as soon as possible. Tackling this problem using Algorithm IPABC is done as follows: given a constraint system of the form $\forall t \in I_t{:}c_1 \wedge \cdots \wedge c_m$ and a Cartesian product of domains $\mathbf{B} = I_1 \times \cdots \times I_n$, we have two degrees of freedom during the solving process, namely, the selection of the next constraint to consider, and the selection of the next variable to split. Figure 11 presents the differences with regard to the order of generation of solutions for *Circle*$_2$ for two strategies

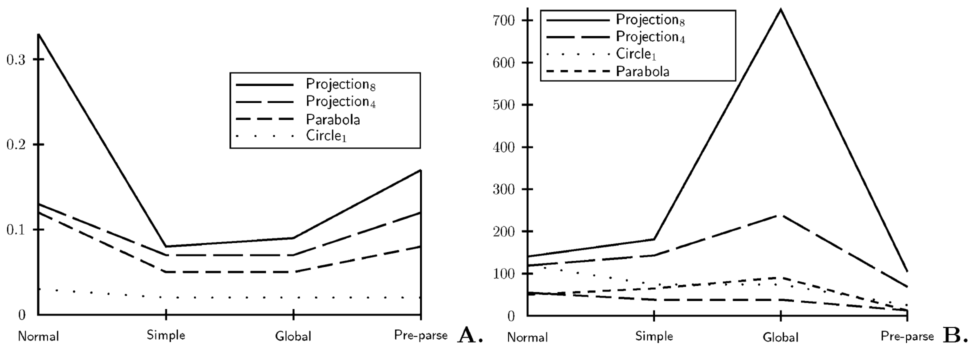


Fig. 12. (A) First solution. (B) All solutions (*times in seconds*).

concerning the variable splitting order:

—*depth-first*, where each constraint is considered in turn, and each domain is split to the threshold splitting limit;

—*semi-depth-first* where each constraint is considered in turn, but each variable is split only once and then put at the end of the domain queue.

As one may see, the semi-depth-first algorithm computes consecutive solutions spread over all the search space, while the depth-first algorithm computes solutions downward and from right to left.

Some strategies on constraint consideration order have also been investigated, whose impact on speed is described hereunder. Four strategies may be singled out:

—*Simple method.* Box consistency is applied on the negation of each constraint in turn.

—*Pre-parse method.* This method interleaves the *simple method* with a *pre-parse algorithm*: given a constraint c, and t the universally quantified variable, k canonical intervals are extracted from the domain I_t of t, and the consistency of c is tested for every one of them. At this stage, a failed check is sufficient to initiate a backtrack.

—*Normal method.* Box consistency is computed both for each constraint and for its negation.

—*Global method.* Given constraints $c_i, c_{i+1}, \dots, c_m$, the *global method* applies the *normal method* on c_i, then checks whether the output boxes are consistent with the remaining constraints by mere evaluation.

Charts in Figure 12 present the time spent for obtaining the first and all solutions for four benchmarks on a SUN UltraSparc 1/167:MHz under Solaris 2.5.

Considering first Chart A, one may see that the *simple method* is the most interesting strategy for computing the first solution, while the *normal method* is very time consuming for problems with "many" constraints (projection$_8$). On the other hand, while the *pre-parse method* is a bad choice for computing only one solution, it is competitive for obtaining all solutions.

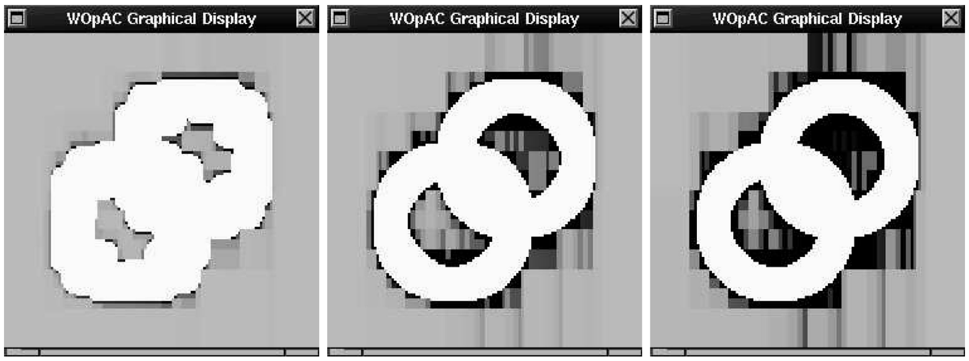


Fig. 13. Impact of the time splitting threshold ω on precision in JLA (from left to right: $\omega = 0.5$, $\omega = 0.1$, and $\omega = 0.05$.)

Table VI. JLA Versus IPABC—First Solution

Benchmark	JLA	IPABC	JLA/IPABC
Parabola ($\varepsilon = 10^{-2}$)	0.04	0.02	2.0
Parabola ($\varepsilon = 10^{-3}$)	0.77	0.02	38.5
Circle$_2$ ($\varepsilon = 10^{-2}$)	3.52	0.01	352000
Circle$_2$ ($\varepsilon = 10^{-3}$)	3.57	0.01	357000
Satellite ($\varepsilon = 10^{-2}$)	0.91	0.99	0.91
Satellite ($\varepsilon = 10^{-3}$)	68.7	0.99	68.48
Robot ($\varepsilon = 10^{-1}$)	0.13	0.01	13
Robot ($\varepsilon = 10^{-2}$)	0.50	0.01	50
Projection$_8$ ($\varepsilon = 10^{-1}$)	0.05	0.07	0.71
Projection$_8$ ($\varepsilon = 10^{-2}$)	0.14	0.07	2.0
Projection$_8$ ($\varepsilon = 10^{-3}$)	3.01	0.07	43.0
Projection$_4$ ($\varepsilon = 10^{-2}$)	0.11	0.03	3.66
Projection$_4$ ($\varepsilon = 10^{-3}$)	1.11	0.03	37
GarloffGraf1 ($\varepsilon = 10^{-2}$)	15.79	0.01	1579
GarloffGraf2 ($\varepsilon = 10^{-2}$)	5.58	0.04	139
PointPath ($\varepsilon = 0.5$)	???	7.85	???

7.3 Results

Algorithms JLA and IPABC provide different sets of solutions for the same problem. Consequently, a direct comparison of their performances is quite difficult. Moreover, the actual implementation of JLA uses a splitting threshold ω for slicing the domain of the universally quantified variable t instead of checking consistency by eventually reaching canonicity of the samples of the domain I_t. Figure 13 shows the impact of the threshold on the computed solutions for benchmark *Circle*$_2$.

Table VI (respectively VII) compares algorithms JLA and IPABC from the speed point of view for computing the first solution (respectively all solutions). Times are given in seconds on a Pentium III at 800 MHz under Linux.

As one can see, the efficiency of IPABC compared to that of JLA increases steadily with the precision required.

We were not able to obtain any result for *PointPath* with JLA after several hours.

Table VII. JLA Versus IPABC—All Solutions

Benchmark	JLA	IPABC	JLA/IPABC
Parabola	10.65	1.22	8.72
Circle$_2$	3.16	0.15	21.06
Circle$_3$	3.41	0.55	6.2
Satellite	>600	54.91	>175
Robot	22.65	1.97	11.5
Projection$_8$	>600.00	3.43	>175.4
Projection$_5$	182.09	16.01	11.35
GarloffGraf1 ($\varepsilon = 10^{-2}$)	422.54	1.49	283.6
GarloffGraf2 ($\varepsilon = 10^{-2}$)	29.54	1.04	28.40
PointPath ($\varepsilon = 0.5$)	???	534.14	???

8. CONCLUSION

Unlike the methods used to deal with universally quantified variables described in Hong and Buchberger [1991], the algorithms presented in this article are purely numerical ones (except for the negation of constraints). Since they rely on "traditional" techniques used by most of the interval constraint-based solvers, they may benefit from the active researches led to speed up these tools. What is more, they are applicable to the large range of constraints for which an outer contracting operator may be devised. By contrast, CAD-based methods deal with polynomial constraints only, as is the case with the method based on Bernstein expansion [Garloff and Graf 1999].

Despite the dramatic improvement of the new method described herein over the one given by Jardillier and Languénou [1998], the handling of complex scenes with many objects and a camera allowed to move along all its degrees of freedom in a reasonable time is beyond reach for the moment. Nevertheless, a comforting idea is that most of the traditional camera movements involve but few of the degrees of freedom, thereby reducing the number of variables to consider.

Following the work of Markov [1995] and Shary [1995], a direction for future research is to compare the use of *Kaucher arithmetic* [Kaucher 1980] to compute inner approximations of relations with the use of outer contracting operators. Their work is also related to that by Armengol et al. [1998] and Gardeñes et al. [Gardeñes and Trepat 1980; Gardeñes and Mielgo 1986] on modal interval arithmetic. Yet, these approaches require algebraizing trigonometric constraints, an operation known to slow down computation [Pau and Schicho 2000].

APPENDIX

A. NOTATIONS

$\mathbb{F}$	Set of floating-point numbers
a^+	Smallest floating-point number greater than a
a^-	Greatest floating-point number smaller than a
$\mathbb{I}$	Set of closed intervals whose bounds are floating-point numbers
$\text{Outer}(\rho)$	Smallest box containing ρ: $\text{Outer}(\rho) = \bigcap\{\mathbf{B} \in \mathbb{I}^n \mid \rho \subseteq \mathbf{B}\}$

$\mathsf{Inner}(\rho)$	Inner approximation of ρ: $\mathsf{Inner}(\rho) = \{\mathbf{r} \in \mathbb{R}^n \mid \mathsf{Outer}(\{\mathbf{r}\}) \subseteq \rho\}$
$\wr\mathcal{S}\wr$	Union of the boxes in $\mathcal{S}$: $\wr\{\mathbf{B_1}, \ldots, \mathbf{B_n}\}\wr = \{\mathbf{r} \in \mathbb{R}^n \mid \exists i \in \{1, \ldots, n\}$ such that $\mathbf{r} \in \mathbf{B_i}\}$
ρ_c	Relation associated to the constraint c: $\rho_c = \{(r_1, \ldots, r_n) \in \mathbb{R}^n \mid c(r_1, \ldots, r_n)\}$
$\rho_{c^\forall}$	Relation associated to $\forall v \in I : c$
$\overline{c}$	"Negation" of the constraint c: ($\overline{f \leq 0}$ is defined as $f \geq 0$)
$\mathsf{Dom}_{\mathbf{B}}(x)$	Domain of the variable x in the box $\mathbf{B}$
$\mathbf{B_1} \boxslash \mathbf{B_2}$	Set difference of $\mathbf{B_1}$ and $\mathbf{B_2}$ as a set of boxes: $\wr\mathbf{B_1} \boxslash \mathbf{B_2}\wr = \{r \in \mathbf{B_1} \mid \mathsf{Outer}(\{r\}) \cap \mathbf{B_2} = \varnothing\}$
$\mathbf{B}_{I \leftarrow J}$	Box $\mathbf{B}$ where the interval I is replaced by the interval J
$\mathcal{P}(\mathcal{S})$	Power set of the set $\mathcal{S}$
$\mathsf{Split}_k(\mathbf{B})$	Split in k boxes the box $\mathbf{B}$
$\mathsf{Split}_k^{\setminus v}(\mathbf{B})$	Split in k boxes the box $\mathbf{B}$ (never splits the interval corresponding to the domain of v)
$\biguplus_i \{(\mathcal{U}_1, \ldots, \mathcal{U}_n)\}$	union of vectors of sets componentwise: $$\biguplus_i \left\{(\mathcal{U}_1^i, \ldots, \mathcal{U}_n^i)\right\} = \left(\bigcup_i \mathcal{U}_1^i, \ldots, \bigcup_i \mathcal{U}_n^i\right)$$

ACKNOWLEDGMENTS

The authors gratefully acknowledge the insightful perusal of the anonymous referees and discussions with Laurent Granvilliers and Lucas Bordeaux that helped improve preliminary versions of this article.

REFERENCES

ABDALLAH, C., DORATO, P., YANG, W., LISKA, R., AND STEINBERG, S. 1996. Applications of quantifier elimination theory to control theory. In *Proceedings of the 4th IEEE Mediterranean Symposium on Control and Automation* (Malene, Crete, Greece). 340–345.

ABRAMS, S. AND ALLEN, P. K. 1997. Computing camera viewpoints in an active robot work-cell. IBM Res. rep. RC 20987. IBM Research Division, Yorktown Heights, NY.

ALEFELD, G. AND HERZBERGER, J. 1983. *Introduction to Interval Computations*. Academic Press Inc., New York, NY.

ARIJON, D. 1976. *Grammar of the Film Language*. Hastings House Publishers, New York, NY.

ARMENGOL, J., TRAVÉ-MASSUYÈS, L., VEHÍ, J., AND SÁINZ, M. A. 1998. Modal interval analysis for error-bounded semiqualitative simulation. In *1r Congrés Català d'Intelligència Artificial*. 223–231.

BENHAMOU, F. 1995. Interval constraint logic programming. In *Constraint Programming: Basics and Trends: 1994 Châtillon Spring School, Châtillon-sur-Seine, France, May 16–20, 1994*, A. Podelski, Ed. Lecture Notes in Computer Science, vol. 910. Springer-Verlag, Berlin, Germany, 1–21.

BENHAMOU, F. 1996. Heterogeneous constraint solving. In *Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP'96)*. Lecture Notes in Computer Science, vol. 1139. Springer-Verlag, Aachen, Germany, 62–76.

BENHAMOU, F., GOUALARD, F., GRANVILLIERS, L., AND PUGET, J.-F. 1999. Revising hull and box consistency. In *Proceedings of the 16th International Conference on Logic Programming (ICLP'99)*. MIT Press, Cambridge, MA. 230–244.

Benhamou, F., McAllester, D., and Van Hentenryck, P. 1994. CLP(Intervals) revisited. In *Proceedings of the International Symposium on Logic Programming (ILPS'94)*. MIT Press, Cambridge, MA, 124–138.

Benhamou, F. and Older, W. J. 1997. Applying interval arithmetic to real, integer and Boolean constraints. *J. Logic Programm. 32,* 1, 1–24.

Blinn, J. 1988. Where am I? What am I looking at? *IEEE Comput. Graph. Appl. 8*, 4 (July), 76–81.

Christie, M., Languénou, E., and Granvilliers, L. 2002. Modeling camera control with constrained hypertubes. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP'2002)*, P. V. Hentenryck, Ed. Lecture Notes in Computer Science, vol. 2470. Springer-Verlag, Berlin, Germany, 618–632.

Collavizza, H., Delobel, F., and Rueher, M. 1999. Extending consistent domains of numeric CSP. In *Proceedings of the 16th IJCAI*. (Stockholm, Sweden). Vol. 1. 406–411.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*. Lecture Notes in Computer Science, vol. 33. Springer, Kaiserslauten, Germany, 134–183.

Davenport, G., Smith, T. A., and Pincever, N. 1991. Cinematic primitives for multimedia. *IEEE Comput. Graph. Appl. 11*, 4 (July), 67–74.

Drucker, S. M. 1994. Intelligent camera control for graphical environments. Ph.D. dessertation. Program in Media Arts and Sciences, School of Architecture and Planning, MIT, Cambridge, MA.

Drucker, S. M., Galyean, T. A., and Zeltzer, D. 1992. CINEMA: A system for procedural camera movements. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, M. Levoy and E. E. Catmull, Eds. ACM Press, New York, NY, 67–70.

Drucker, S. M. and Zeltzer, D. 1994. Intelligent camera control in a virtual environment. In *Proceedings of Graphics Interface '94*. Canadian Information Processing Society, Banff, Alta., Canada, 190–199.

Ebers, J. J. and Moll, J. L. 1954. Large-scale behaviour of junction transistors. *IRE Procs. 42*, 1761–1772.

Emiris, I. Z. and Mourrain, B. 1999. Computer algebra methods for studying and computing molecular conformations. *Algorithmica 25*, 2/3, 372–402.

Farouki, R. T. and Rajan, V. T. 1987. On the numerical condition of polynomials in Bernstein form. *Comput. Aided Geom. Des. 4*, 191–216.

Gardeñes, E. and Trepat, A. 1980. Fundamentals of SIGLA, an interval computing system over the completed set of intervals. *Comput. 24*, 2–3, 161–179.

Gardeñes, E. H. and Mielgo, H. 1986. Modal intervals: Reasons and ground semantics. In *Interval Mathematics*. Lecture Notes in Computer Science, vol. 212. Springer-Verlag, Berlin, Germany.

Garloff, J. and Graf, B. 1999. Solving strict polynomial inequalities by Bernstein expansion. In *The Use of Symbolic Methods in Control System Analysis and Design*, N. Munro, Ed. The Institution of Electrical Engineers, London, England, 339–352.

Gleicher, M. and Witkin, A. 1992. Through-the-lens camera control. In *Comput. Graph. (SIGGRAPH '92 Proceedings*, E. E. Catmull, Ed.). 26, 2, 331–340.

Granvilliers, L. and Benhamou, F. 2001. Progress in the solving of a circuit design problem. *J. Global Opt. 20*, 2, 155–168.

Hansen, E. R. 1992. *Global Optimization Using Interval Analysis*. Pure and Applied Mathematics. Marcel Dekker Inc, New York, NY.

Haroud, D. and Faltings, B. 1994. Global consistency for continuous constraints. Lecture Notes in Computer Science, vol. 874. Springer-Verlag, Berlin, Germany, 40–50.

Hong, H. and Buchberger, B. 1991. Speeding-up quantifier elimination by Gröbner bases. Tech. Rep. 91-06. RISC-Linz, Johannes Kepler University, Linz, Austria.

IEEE. 1985. IEEE standard for binary floating-point arithmetic. Tech. rep. IEEE Std 754-1985. Institute of Electrical and Electronics Engineers, Piscataway, NJ. Reaffirmed 1990.

Jardillier, F. and Languénou, É. 1998. Screen-space constraints for camera movements: the virtual cameraman. In *Eurographics'98 proceedings*, N. Ferreira and M. Göbel, Eds. Vol. 17. Blackwell Publishers, Oxford, UK, 175–186.

Jaulin, L. and Walter, E. 1993. Set inversion via interval analysis for nonlinear bounded-error estimation. *Automatica 29*, 4, 1053–1064.

JAULIN, L. AND WALTER, E. 1996. Guaranteed tuning with application to robust control and motion planning. *Automatica 32*, 8, 1217–1221.

JIRSTRAND, M. 1995. Cylindrical algebraic decomposition—an introduction. Tech. Rep. 1995-10-18, Department of Electrical Engineering, Linköping University, Linköping, Sweden.

KAUCHER, E. W. 1980. Interval analysis in the extended interval space $\mathbb{IR}$. *Comput. (Supplementum) 2*, 33–49.

KUTSIA, T. AND SCHICHO, J. 1999. Numerical solving of constraints of multivariate polynomial strict inequalities. Tech. rep. 99-31. RISC Institute, Johannes Kepler University, Linz, Austria.

LEFÈVRE, V., MULLER, J.-M., AND TISSERAND, A. 1998. Towards correctly rounded transcendeatals. *IEEE Trans. Comput. 47*, 11 (Nov.), 1235–1243.

MACKWORTH, A. K. 1977. Consistency in networks of relations. *Art. Intell. 1*, 8, 99–118.

MARKOV, S. 1995. On directed interval arithmetic and its applications. *J. Univers. Comput. Sci. 1*, 7, 514–526.

MEINTJES, K. AND MORGAN, A. P. 1990. Chemical equilibrium systems as numerical test problems. *ACM Trans. Math. Softw. 16*, 2 (June), 143–151.

MOORE, R. E. 1966. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ.

NEUMAIER, A. 1990. *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications, vol. 37. Cambridge Unversity Press, Cambridge, U.K.

OLDER, W. J. AND VELLINO, A. 1990. Extending Prolog with constraint arithmetic on real intervals. In *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*. IEEE Computer Society Press, New York, NY.

PAU, P. AND SCHICHO, J. 2000. Quantifier elimination for trigonometric polynomials by cylindrical trigonometric decomposition. *J. Symbol. Computat. 29*, 6 (June), 971–983.

PUGET, J.-F. 1994. A C++ implementation of CLP. In *Proceedings of the Singapore Conference on Intelligent Systems* (SPICIS'94, Singapore).

PUGET, J.-F. AND VAN HENTENRYCK, P. 1998. A constraint satisfaction approach to a circuit design problem. *J. Global Opt. 13*, 75–93.

RADEMACHER, H. A. 1948. On the accumulation of errors in processes of integration on high-speed calculating machines. *Annals Comput. Laor. Harvard Univ. 16*, 176–185. Cited in Stoer and Bulirsch [1993].

SAM, J. 1995. Constraint consistency techniques for continuous domains. Ph.D. dissertation. École polytechnique fédérale de Lausanne, Lausanne, Switzerland.

SHARY, S. P. 1995. Algebraic solutions to interval linear equations and their applications. In *Numerical Methods and Error Bounds, Proceedings of the IMACS-GAMM International Symposium on Numerical Methods and Error Bounds*, G. Alefeld and J. Herzberger, Eds. Akademie Verlag, Oldenburg, Germany, 224–233.

SHARY, S. P. 1999. Interval Gauss-Seidel method for generalized solution sets to interval linear systems. In *MISC'99—Workshop on Applications of Interval Analysis to Systems and Control* (Girona, Spain). 51–65.

SHOEMAKE, K. 1985. Animation rotations with quaternion curves. *Comput. Graph. Proceedings of SIGGRAPH'83, San Francisco, CA). 19*, 3, 245–254.

SNYDER, J. M. 1992. Interval analysis for computer graphics. In *Computer Graphics (SIGGRAPH '92 Proceedings*, E. E. Catmull, Ed.). *26*, 2, 121–130.

STOER, J. AND BULIRSCH, R. 1993. *Introduction to Numerical Analysis*, 2nd ed. in Texts in Applied Mathematics, vol. 12. Springer, Heidelberg, Germany.

SUNAGA, T. 1958. Theory of an interval algebra and its application to numerical analysis. In *Research Association of Applied Geometry Memoirs of the Unifying Study of Basic Problems in Engineering and Physical Sciences by Means of Geometry*, K. Kondo, Ed. Vol. 2. Gakujutsu Bunken Fukyu-Kai, Japan, Chapter Miscellaneous Subjects II, 547–564.

TARABANIS, K. 1990. Automated sensor planning and modeling for robotic vision tasks. Tech. rep. CUCS-045-90. Columbia University, New York, NY.

TURING, A. M. 1948. Rounding-off errors in matrix processes. *Quart. J. Mech. 1*, 287–308. Cited in Wilkinson [1963].

VAN HENTENRYCK, P., MCALLESTER, D., AND KAPUR, D. 1997a. Solving polynomial systems using a branch and prune approach. *SIAM J. Numer. Anal. 34*, 2 (April), 797–827.

Van Hentenryck, P., Michel, L., and Deville, Y. 1997b. *Numerica: A Modeling Language for Global Optimization*. MIT Press, Cambridge, MA.

Waltz, D. L. 1975. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, P. H. Winston, B. Horn, M. Minsky, and Y. Shirai, Eds. McGraw-Hill, New York, NY, Chap. 2, 19–91.

Ward, A. C., Lozano-Pérez, T., and Seering, W. P. 1989. Extending the constraint propagation of intervals. In *Proceedings of IJCAI'89*. 1453–1458.

Wilkinson, J. H. 1963. *Rounding Errors in Algebraic Processes*. Prentice-Hall, New Jersey. Reprinted as Dover Publications (1994).

Received July 2000; revised September 2002; accepted May 2003