Improving a Constraint Programming Approach for Parameter Estimation

Bertrand Neveu, Martin de La Gorce, Gilles Trombettoni

Improving a Constraint Programming Approach for Parameter Estimation

Bertrand Neveu
LIGM
Université Paris Est
Marne-la-Vallée, France
Email: Bertrand.Neveu@enpc.fr

Martin de la Gorce
LIGM
Université Paris Est
Marne-la-Vallée, France
Email: martin.delagorce@gmail.com

Gilles Trombettoni
LIRMM
Université de Montpellier
Montpellier, France
Email: Gilles.Trombettoni@lirmm.fr

***Abstract*—The parameter estimation problem is a widespread and challenging problem in engineering sciences consisting in computing the parameters of a parametric model that fit observed data. Calibration or geolocation can be viewed as specific parameter estimation problems. In this paper we address the problem of finding all the instances of a parametric model that can explain at least q observations within a given tolerance. The computer vision community has proposed the RANSAC algorithm to deal with outliers in the observed data. This randomized algorithm is efficient but non-deterministic and therefore incomplete. Jaulin et al. proposes a complete and combinatorial algorithm that exhaustively traverses the whole space of parameter vectors to extract the valid model instances. This algorithm is based on interval constraint programming methods and on a so called q-intersection operator, a relaxed intersection operator that assumes that at least q observed data are inliers. This paper proposes several improvements to Jaulin et al.'s algorithm. Most of them are generic and some others are dedicated to the shape detection problem used to validate our approach. Compared to Jaulin et al.'s algorithm, our algorithm can guarantee a number of fitted observations in the produced model instances. Also, first experiments in plane and circle recognition highlight speedups of two orders of magnitude.**

I. Introduction

The *parameter estimation* problem handled as an *inverse problem* is the process of calculating the *parameters vectors* of a parametric model from a set of measurements or more generally *observed data*.

Inverse problems are well-studied mathematical problems because they tell us about parameters that we cannot directly observe. In this article, we propose a new parameter estimation algorithm based on constraint programming. We validate it in shape detection that can be cast as an inverse problem: the observed data are 2D or 3D points of a scene; the unknown parameters vector defines a primitive like a plane or a circle that is compatible with a minimum number q of observations.

The parametric model is specified by a function M that maps an n-dimensional real vector $\mathbf{p} = (p_1, ..p_N)$, referred as the *parameters vector*, to a *model instance* $M(\mathbf{p})$ that is a subset of the space $\mathbb{R}^d$ in which the observed data lie. This mapping is specified through the use of a real valued function f and an implicit equation $f(\mathbf{x}, \mathbf{p}) = 0$. Given a parameter vector $\mathbf{p}$, the corresponding model instance $M(\mathbf{p})$ is given by

$$M(\mathbf{p}) = \{\mathbf{x} \mid f(\mathbf{x}, \mathbf{p}) = 0\} \tag{1}$$

Given a finite set of observations $\{\mathbf{o}_1, ..., \mathbf{o}_m\} \subset \mathbb{R}^d$, we search for all parameters vectors that are compatible with at least q of these observations. An observation $\mathbf{o}_i$ is said compatible with the parameters vector $\mathbf{p}$, using a tolerance value τ, when it satisfies an *observation constraint*:

$$-\tau \leq f(\mathbf{o}_i, \mathbf{p}) \leq +\tau \tag{2}$$

Given a model instance $M(\mathbf{p})$, we will define the *consensus set* $C(\mathbf{p})$ as the set of observations compatible with $\mathbf{p}$:

$$C(\mathbf{p}) = \{\mathbf{o}_i | -\tau \leq f(\mathbf{o}_i, \mathbf{p}), \leq \tau\} \tag{3}$$

In the case f is a linear function of $\mathbf{p}$ and we search for all parameters vectors $\mathbf{p}$ that are compatible with all the observations, i.e. we assume that there are no outliers, then the set of solutions is a convex polyhedron (intersection of a finite number of half-spaces). The problem of finding a single point in that polyhedron is equivalent to finding a feasible point for a set of linear inequality constraints, which can be done using any Linear Programming algorithm. The inverse problem becomes more challenging when the function f used to define the parametric model is not linear and/or in presence of outliers.

Outliers can have numerous origins, including extreme values of the noise, erroneous measurements and data reporting errors. In order to cope with outliers we search for model instances whose consensus set contains at least q elements. We refer to them as *valid model instances*. This problem can be reformulated as a numerical constraint satisfaction problem (NCSP) with N variables $p_1, \ldots, p_N$ having a real interval domain, and a single constraint:

$$|C((p_1, \ldots, p_N))| \geq q \tag{4}$$

Because we use inequalities, this problem does not have point-wise isolated solutions but a finite set of continuums of feasible points. In this paper we aim at obtaining a concise outer approximation of the whole solution set as a union of boxes refined by the solving process. Each box returned

either 1) reaches a size that is below a fixed precision threshold or 2) contains a feasible point that matches at least q observations such that no other point matches more observations.

A. RANSAC: Robust parameter estimation coping with outliers

The random sample consensus algorithm (RANSAC) [8] has become a state-of-the-art tool in the computer vision and image processing communities to achieve parameter estimation robust to outliers. In addition to the observed dataset, the input of RANSAC is a minimum number q of inliers and the tolerance τ defined above. RANSAC is an iterative and incomplete method that searches for the best parameters vector by repeating the following steps.

1) *Random sampling:* A subset of the observations is randomly selected. The cardinality of the sample is the smallest sufficient to determine the model parameters: N observations are generally required to characterize the N parameters.
2) The parameters vector corresponding to the model instance that best fits the selected sample is computed by solving a system of N equality constraints.
3) *Consensus:* One checks if at least q elements of the entire observation dataset are inliers given the tolerance parameter τ.
   Note that several trials of these first three steps can be necessary, especially to select N points that will lead to a valid model instance.
4) If a consensus is obtained, a second and better model may be computed by using the whole consensus set.

In shape detection, the goal is to detect *all* the hidden primitives that correspond to valid model instances. It is important to understand that RANSAC is a randomized algorithm that may not find all the valid model instances. Indeed, the four-steps RANSAC explained above can find only one primitive. In order to find several primitives, the RANSAC version used in our experiments [16] and dedicated to shape recognition finds the different primitives in a greedy way: it definitely removes the data points involved in the consensus set of the current model instance before searching for a next one. Therefore we cannot know whether this iterative process converges onto the whole set of model instances that contain at least q inliers.

B. Complete interval constraint programming approach

A parameter estimation method based on interval constraint programming that is robust to outliers was first described in [11].

Intervals are the first ingredient of the approach. We denote by $[x_i] = [\underline{x_i}, \overline{x_i}]$ the interval/domain of the real-valued variable x_i, where $\underline{x_i}$, $\overline{x_i}$ are floating-point numbers. A Cartesian product of intervals like the domain $[\mathbf{x}] = [x_1] \times ... \times [x_N]$ is called a (parallel-to-axes) *box*. $width(x_i)$ denotes the size or *width* $\overline{x_i} - \underline{x_i}$ of an interval $[x_i]$. The width of a box is given by the width $\overline{x_m} - \underline{x_m}$ of its largest dimension x_m. Interval methods also provide *contracting operators* (called contractors), i.e. methods that can reduce the variable domains involved in a constraint or a set of constraints without loss of solutions. Let us consider one observation $\mathbf{o}_i$ and a subspace of the parameter space given by a box $[\mathbf{p}]$. Given an observation $\mathbf{o}_i$, we denote V_i the set of parameters vectors that are compatible with that observation, i.e.

$$V_i = \{\mathbf{p} \mid -\tau \leq f(\mathbf{o}_i, \mathbf{p}) \leq +\tau\} \tag{5}$$

The set of parameter values within that box that are compatible with $\mathbf{o}_i$ is given by $V_i \cap [\mathbf{p}]$. A contractor is able to reduce the input box $[\mathbf{p}]$ while keeping all the points in $V_i \cap [\mathbf{p}]$, i.e. without losing any solution. The main contractor used in this paper is the well-known `HC4-revise` [1], [14], also called forward-backward. This contractor handles a single constraint and obtains a (generally non optimal [6]) contracted box including all the solutions of that constraint. Let us give the principles of `HC4-revise` on the constraint $f(\mathbf{o}_i, \mathbf{p}) \leq +\tau$.

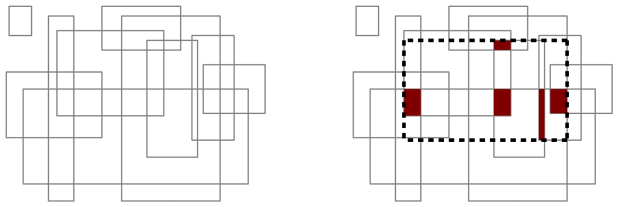
The forward phase traverses the expression tree of the constraint bottom up using interval arithmetic. In the example, the unknowns $\mathbf{p}$ are replaced by their intervals $[\mathbf{p}]$ in the expression and interval arithmetic evaluates $[f]([\mathbf{o}_i], [\mathbf{p}])$. For instance, if $f(\mathbf{o}_1, \mathbf{p}) = o_1 + p_1 + p_2$ with $o_1 = 2.5$ and $[\mathbf{p}] \in [-2, 3] \times [0, 1]$, we have $[f]([o_1], [p_1], [p_2]) = [2.5, 2.5] + [-2, 3] + [0, 1] = [1.5, 6.5]$. The image interval is then intersected with $[-\infty, \tau]$ because the constraint is an inequality. If the tolerance τ is `1e-1`, then the intersection is empty. The contractor terminates with an empty contracted box, which proves that no vector inside the initial domain $[\mathbf{p}]$ satisfies the constraint. If the resulting interval $[r]$ is not empty, e.g. the tolerance is 5, then the computed interval is $[r] = [1.5, 5]$ and one can run the second backward phase.

In the backward phase, the expression tree is traversed top-down, and interval arithmetic is applied on so-called *inverse* operators. Without detailing, this phase amounts to evaluating all the (inverse) functions isolating every variable occurrence. Consider for instance the inverse function f^{p_1} used to contract p_1: $f^{p_1}(o_1, p_2) = r - o_1 - p_2$. We evaluate f^{p_1} using interval arithmetic, which produces the following contraction: $[p_1] := [p_1] \cap [f^{p_1}]([o_1], [p_2]) = [-2, 3] \cap ([1.5, 5] - [2.5, 2.5] - [0, 1]) = [-2, 3] \cap [-2, 2.5] = [-2, 2.5]$.

To contract a system of constraints, the HC4 algorithm performs a *propagation loop* applying iteratively the HC4-Revise procedure introduced above on each constraint individually until a quasi fixpoint is obtained in terms of contraction.

Jaulin et al. proposed in [12] a simple deterministic algorithm based on interval methods to handle inverse problems:

- A search tree is built to exhaustively explore the parameter space. $[\mathbf{p}]$ is recursively subdivided: one variable



(a) A set S of 2-dimensional boxes

(b) The dashed box is $\cap^4 S$. Zones that belong to at least 4 boxes are darkened.

Figure 1. Illustration of q-intersection for $q = 4$, $m = 2$.

p_i in $\mathbf{p}$ is selected, its domain $[p_i]$ is bisected into two sub-intervals and the two corresponding sub-boxes are explored recursively. The combinatorial process stops when a precision is reached, i.e. when the width of the current box is inferior to ϵ_{sol} (the box is thus a leaf of the search tree).

- At each node of the tree, the current box is contracted w.r.t. the observation constraints, and sometimes eliminated, using the forward-backward contractor described above.

Compared to RANSAC, the main advantage of this approach is that all the valid model instances can be produced in an exhaustive way. In addition, bounded errors can also be taken into account in the observed data: a measurement with a bounded error can be modeled by a (constant) box $[\mathbf{o}]$ and not a vector. The main drawback is that outliers lead to an empty contracted box (i.e., no model instance exists in the box) since the parameter box contracted using an outlier observation constraint does not intersect the other ones.

The second ingredient used in the parameter estimation tool based on interval constraint programming was also proposed by Jaulin et al. to deal with outliers (see [10]). Like RANSAC, the approach assumes that at least q observations are inliers, the other ones being viewed as potential outliers.

This idea is implemented by the *q-intersection* operator. This operator relaxes the previous intersection of m boxes (corresponding in our inverse problem to the contraction of $[\mathbf{p}]$ w.r.t. all the m observation constraints) by the union of all the intersections obtained with q boxes. More formally:

Definition 1: Let S be a set of boxes. The q-intersection of S, denoted by $\cap^q S$, is the box of smallest perimeter that encloses the set of points of $\mathbb{R}^n$ belonging to at least q boxes. For instance, the box in dotted lines in Fig. 1–b is the 4-intersection of the $m = 10$ two-dimensional boxes (in plain lines).

The parameter estimation tool proposed by Jaulin et al. has been extended to cope with outliers using a q-intersection operator. The contraction phase is replaced by a more sophisticated routine called `contAndQinter` in this paper:

1) A forward-backward contraction is achieved on each of the m observation constraints, which produces a set S of boxes. Note that no (standard) intersection is achieved on these boxes.
2) The box returned by `contAndQinter` is the q-intersection of these contracted boxes: $\cap^q S$.

The q-intersection of boxes is a difficult problem and has been proven DP-complete in [3] where an exact algorithm based on the search of q-cliques has been proposed. In this paper, we simply resort to a non optimal q-intersection operator often used in parameter estimation. Computing a reasonable enclosure of $\cap^q S$ may be satisfactory, provided it can be done in polynomial time, since the q-intersection operator is typically used to filter the search space at each node of a search tree. It appeared that the exact q-intersection algorithm was too costly for the shape detection instances studied in this paper.

This algorithm, called here `q-proj`, solves the problem on each dimension independently. For each dimension i, the algorithm computes the projection $S[i]$ of the boxes and solves the q-intersection problem on $S[i]$. Since $S[i]$ is a set of intervals, each $\cap^q S[i]$ can be computed in polynomial time by sorting the lower and upper bounds using a method first introduced in [5]. The overall complexity of `q-proj` is $O(nm\log(m))$. Figure 2 illustrates the algorithm.

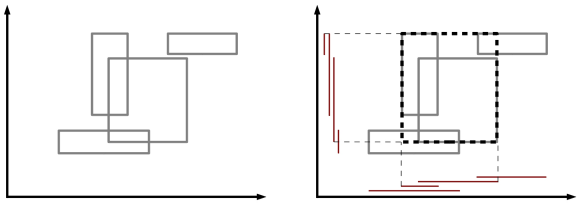


Figure 2. Principle of `q-proj` for $q = 2$, $n = 2$. The algorithm outputs the dashed box, an overestimate of $\cap^2 S$.

Jaulin et al.'s interval-based approach has been used in several parameter estimation applications, including geolocation. More recently, a 3D reconstruction problem has been handled by estimating the coefficients of the transformation matrix between two poses [2].

Contribution

This paper describes several improvements to Jaulin et al.s interval CP parameter estimation solver. Contrarily to RANSAC, our tool is still exhaustive (complete), but its answer is more interesting than Jaulin et al.'s approach because a large fraction of the model instances found are guaranteed to fit at least q observations. Generic improvements are described in Section II while improvements dedicated to shape detection are described in Section III. Overall, gains in performance of 2 or 3 orders of magnitude are measured on first artificial and realistic shape (i.e., planes in 3D and circles in 2D) detection instances. The approach gives us the

hope of superseding the incomplete RANSAC approach for real applications.

II. Generic improved parameter estimation algorithm

A. Combinatorial q-inter based algorithm

Algorithm 1 describes the proposed generic parameter estimation algorithm that exhaustively explores the whole parameter space by achieving a depth-first search. Several informations are updated at each node of the tree search, using backtrackable data structures:

- a box $[\mathbf{p}]$ corresponding to the current domain,
- a set $possibleCS$ of observations that is superset of the consensus set of any point in $[\mathbf{p}]$,[1]
- a *valid point* $\mathbf{p} \in [\mathbf{p}]$,
- a set $validCS = C(\mathbf{p})$ of *valid observations*. We have $validCS \subset possibleCS$.

Hence, the following property holds for any node:

$$|node.validCS| \leq \max_{\mathbf{p}\in[\mathbf{p}]} |C(\mathbf{p})| \leq |node.possibleCS| \quad (6)$$

The set $possibleCS$ is initialized with all the observations.[2]

The tree search (i.e., the while loop of Algorithm 1) terminates when the stack of open nodes becomes empty. At each iteration, the most recent open node is popped up, implying a depth-first search tree traversal that limits the memory used. The box is contracted using the `contAndQinter` procedure mentioned in Section I: for every possible observation, a box is produced by contracting the current box with a forward-backward contractor; if this box becomes empty, then $V_i \cap [\mathbf{p}] = \emptyset$ and the corresponding observation is removed from $possibleCS$; the q-intersection of these boxes then contracts the current box (box is an in-out parameter of the procedure). box becomes empty if the q-intersection eliminates it or if less than q observations remain possible.

If the box is not empty, we try to validate it. The `validateBox` procedure checks whether $mid(box)$ satisfies more observations than the valid point inherited from the node parent in the search tree. In that case, $mid(box)$ becomes the new valid point for the current box and $node.validCS$ is assigned with its consensus set. Note that since the q-intersection did not empty the box, we have $|node.possibleCS| \geq q$.

We then look at the box to take a decision. If the box is large and some possible observations are not valid ($node.possibleCS \neq node.validCS$), then the box is split on one dimension into two sub-boxes, and these sub-boxes are pushed into the stack of nodes. $node_1$ and $node_2$ inherit from the backtrackable data structures of $node$ (procedure `quasiCopy` in Algorithm 1). In particular, $node.possibleCS$ is inherited in both $node_1$ and $node_2$; $node.validCS$ and the valid point are inherited in the child node where the valid point of $node$ is still present.

Otherwise, the box is stored in $solutions$. If $|node.validCS| = |node.possibleCS|$, it is guaranteed that no other point in the box can have a strictly larger consensus set. The other condition related to the box size (i.e., $width(box) < \epsilon_{sol}$) aims at speeding up the solving process. In this case, the box returned does not have necessarily $|node.validCS| \geq q$ and thus does not constitute necessarily a guaranteed solution.

Note that basic Jaulin et al.s q-intersection based parameter estimation tool never guarantees the existence of a valid point inside the boxes returned. Indeed, Jaulin et al.'s tool does not manage a set $node.validCS$, so that the boxes returned at the end may contain *no* real vector satisfying at least q observations.

```
Algorithm QinterEstim (box, observations, q, ε_sol, τ)
    solutions ← ∅
    node ← new Node
    node.box ← box
    node.possibleCS ← observations
    node.validCS ← ∅
    nodeStack ← {node}
    while nodeStack ≠ ∅ do
        node ← pop (nodeStack)
        box ← node.box
        contAndQinter (box, τ, q, node.possibleCS)
        if box ≠ ∅ then
            validateBox (box, node, τ, node.possibleCS,
            node.validCS)
            if width(box) < ε_sol or node.validCS =
            node.possibleCS then
                solutions ← solutions ∪ {node}
            else
                bisect (box, box_1, box_2) /* split the box */
                node_1 ← quasiCopy (node, box_1)
                node_2 ← quasiCopy (node, box_2)
                push (nodeStack, node_1)
                push (nodeStack, node_2)
    return solutions
```

Algorithm 1: Generic exhaustive parameter estimation algorithm based on q-intersection.

B. Update of possibleCS during q-intersection

The `q-proj` q-intersection algorithm has been slightly extended to better comply with our parameter estimation tool. The implemented `q-proj` algorithm also returns the set of observations corresponding to boxes eliminated by

[1] In other terms, each point in $[\mathbf{p}]$ violates all observation constraints outside (excluded from) $possibleCS$.

[2] For the sake of simplicity, one considers that an observation is given by its index (between 1 and m) that allows one to reach the information for retrieving the corresponding constraint. In the case of 3D points belonging to a plane, an observation is a point (x, y, z) allowing one to retrieve a constraint "point belonging to plane", i.e. $ax + by + cz + d = 0$ (with e.g. $a^2 + b^2 + c^2 = 1$), where a, b, c and d are the parameters to estimate.

the algorithm. These observations are removed from the set $possibleCS$ managed in the search tree, thus improving the performances. In addition, the `q-proj` algorithm has been modified to also eliminate boxes, inside the bounds of the q-intersection box, that have been detected invalid.

C. Procedure `ProjNewDim`: q-intersection in a new dimension

The procedure `contAndQinter` has been also extended to better contract, using a new dimension. More precisely, once the `q-proj` algorithm ends, a new procedure `ProjNewDim` linearizes all the observation constraints and projects them on a new axis. On this new dimension the intervals are expected to be smaller and q-intersecting them will more likely reduce the resulting box.

We will first assume that the function f used in the model parameterization can be rewritten using a function $\tilde{f}$ of $n-1$ parameters

$$f(\mathbf{x}, \mathbf{p}) = \tilde{f}(\mathbf{x}, \mathbf{p}_{1:N}) - p_0 \tag{7}$$

Given an observation $\mathbf{o}_i$, the function $\tilde{f}$ allows one to estimate the compatible interval for the first parameter p_0 as a function of the remaining parameters $\mathbf{p}_{1:N}$.

$$\tilde{f}(\mathbf{o}_i, \mathbf{p}_{1:N}) - \tau \leq p_0 \leq \tilde{f}(\mathbf{o}_i, \mathbf{p}_{1:N}) + \tau \tag{8}$$

Given a box in the parameter space $B = [\underline{b_0}, \overline{b_0}] \times \cdots \times [\underline{b_N}, \overline{b_N}]$ we denote $\tilde{B}$ the projected lower dimensional box obtained by removing the first dimension interval, i.e. $\tilde{B} = [\underline{b_1}, \overline{b_1}] \times \cdots \times [\underline{b_N}, \overline{b_N}]$. We have $B = [\underline{b_0}, \overline{b_0}] \times \tilde{B}$. Given an observation $\mathbf{o}_i$, we use affine arithmetic [7] to find affine upper and lower bounds of $\tilde{f}(\mathbf{o}_i, .)$ in the box $\tilde{B}$, i.e.

$$\forall \mathbf{p}_{1:N} \in \tilde{B} : \mathbf{a}_i \cdot \mathbf{p}_{1:N} + \underline{c_i} \leq \tilde{f}(\mathbf{o}_i, \mathbf{p}_{1:N}) \leq \mathbf{a}_i \cdot \mathbf{p}_{1:N} + \overline{c_i} \tag{9}$$

where the $\mathbf{a}_i$, $\underline{c_i}$ and $\overline{c_i}$ are the coefficients of the bounding affine functions generated.
From the definition of V_i (see eq. (5)), we get

$$\forall \mathbf{p} \in B \cap V_i : \begin{cases} \mathbf{a}_i \cdot \mathbf{p}_{1:N} + \underline{c_i} - p_0 \leq f(\mathbf{o}_i, \mathbf{p}) \leq \tau \\ \mathbf{a}_i \cdot \mathbf{p}_{1:N} + \overline{c_i} - p_0 \geq f(\mathbf{o}_i, \mathbf{p}) \geq -\tau \end{cases} \tag{10}$$

We denote A_i the region in the parameter space defined by

$$A_i = \{\mathbf{p} | -\tau - \overline{c_i} \leq \mathbf{a}_i \cdot \mathbf{p}_{1:N} - p_0 \leq \tau - \underline{c_i}, \mathbf{p}_{1:N} \in \tilde{B}\} \tag{11}$$

The region A_i is a parallelogram that contains the set of valid parameters vectors in that box. More precisely we can show that we have $(V_i \cap B) \subset A_i$. Figure 3 shows two parallelograms associated to two measures. Given a box B, we want to detect that B does not contain valid solutions using one-dimensional q-intersection after projections of the sets $V_i \cap B$ onto a line. In [10] the projection is done along the box directions, however when looking at Figure 3, it appears that a projection onto a slanted line almost aligned with the small axis of the parallelogram leads to a better separability of the intervals after projection, and thus better rejection rates.

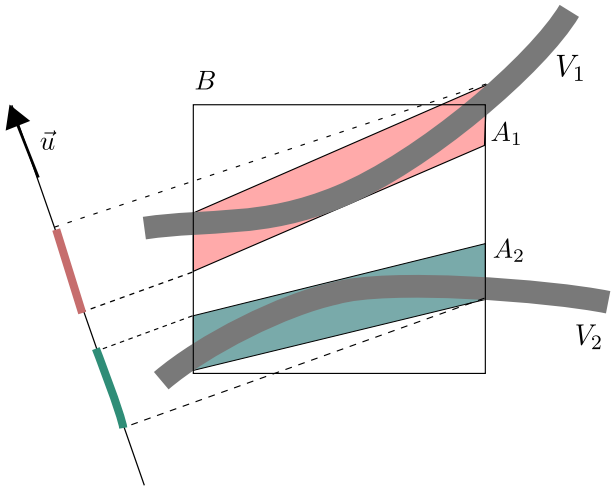


Figure 3. Projecting parallelograms A_i along the mean normal direction. Note that the y-coordinate corresponds to p_0 and the x-coordinate corresponds to p_1.

We compute the direction by taking the mean μ of the vectors $\mathbf{a}_i$ (slopes of the slanted parallelogram side when $N = 2$) associated to all observations that have not yet been rejected for the box B, and we compute an averaged normal vector $\vec{u} = (1, -\mu_1, \ldots, -\mu_{N-1})$. We denote $[\underline{l_i}, \overline{l_i}]$ the one-dimensional interval that corresponds to the projection of the set A_i along the direction $\vec{u}$ and that is defined by

$$\underline{l_i} = \min_{\mathbf{p} \in A_i} \vec{u} \cdot \mathbf{p} \tag{12}$$

$$\overline{l_i} = \max_{\mathbf{p} \in A_i} \vec{u} \cdot \mathbf{p} \tag{13}$$

We can show that we have

$$\underline{l_i} = \min_{\mathbf{p}_{1:N} \in \tilde{B}} (\mathbf{a}_i - \mu) \cdot \mathbf{p}_{1:n} + \underline{c_i} - \tau \tag{14}$$

$$\overline{l_i} = \max_{\mathbf{p}_{1:N} \in \tilde{B}} (\mathbf{a}_i - \mu) \cdot \mathbf{p}_{1:n} + \overline{c_i} + \tau \tag{15}$$

The minimization or the maximization of a linear function over a box domain is straightforward and we get $\underline{l_i} = \gamma_i - \delta_i + \underline{c_i} - \tau$ and $\overline{l_i} = \gamma_i + \delta_i + \overline{c_i} + \tau$ with

$$\gamma_i = \frac{1}{2} \sum_k (\mathbf{a}_{ik} - \mu_k)(\overline{b_k} + \underline{b_k}) \tag{16}$$

$$\delta_i = \frac{1}{2} \sum_k |\mathbf{a}_{ik} - \mu_k|(\overline{b_k} - \underline{b_k}) \tag{17}$$

Once we have computed the intervals $[\underline{l_i}, \overline{l_i}]$ for all the observations that have not been rejected when filtering the box B, we can compute the q-intersection of these intervals and reject the box if empty.

III. Improvements dedicated to shape detection

We also propose three other improvements dedicated to shape detection: the branching strategy (variable choice heuristic) used to build the search tree, the implementation of the HC4-Revise algorithm and the modeling (parameterization) of planes.

A. Branching strategy

For both plane and circle detection, the same types of variable interval bisection heuristics have been designed. All the variables are selected in a round-robin manner, except the distance to origin for planes and the radius for circles. These variables are selected only when the precision on the others has been reached. This can be explained by the mathematical expressions underlying the corresponding observation constraints. The variables selected first have a more significant impact on the image of the respective functions and therefore on contraction, the distance to origin and circle radius being involved in a sum.

B. Forward-backward algorithm implementation

In our first implementation, we used the general-purpose HC4-Revise procedure available in `Ibex`. A performance overhead of this implementation is due to the data structures created when the mathematical expressions are parsed, like trees, bitsets, etc. However, in our shape detection applications, the constraints have a few operators and all the observation constraints follow the same pattern. Therefore we have produced an implementation dedicated to the observation constraints involved. It is not original and Jaulin et al. often resort to dedicated contraction algorithms. However, the gains in performance are significant in our case tests (see Section IV).

For plane detection, we have even removed the backward phase, i.e. the forward phase keeps the box unchanged or eliminates it entirely, leaving the contraction part to the q-intersection algorithm.

C. Efficient line/plane parameterization

A classic 2D line parameterization when doing line detection (using for example the Hough transform [9]) consists in using an angle θ and a distance to the origin ρ using the equation:

$$x\,cos(\theta) + y\,sin(\theta) = \rho \tag{18}$$

This equation uses trigonometric functions that have a computational cost that is greater than simple additions and multiplications.

We use instead a parameterization, where the angle θ is replaced by 2 linked parameters d_x and d_y

$$xd_x + yd_y = \rho \tag{19}$$

with

$$d_x^2 + d_y^2 = 1 \tag{20}$$

In order to suppress symmetric solutions, we can restrict the domains of $d_x \in [0,1]$ and $d_y \in [-1,1]$. However, this parameterization has 3 parameters (d_x, d_y and ρ) linked by a quadratic equation.

Finally, in order to decrease the number of parameters, we propose a new parameterization, with 2 cases:

$$d_x + d_y = 1, d_x \in [0,1] \text{ and } d_y \in [0,1] \tag{21}$$

or

$$d_x - d_y = 1, d_x \in [0,1] \text{ and } d_y \in [-1,0] \tag{22}$$

We can then eliminate the parameter d_x, studying two cases ($d_y \in [-1,0]$ and $d_y \in [0,1]$), where the equation becomes linear. The contractions and projections become then cheaper to compute.

For plane detection, we extend this parameterization in 3D, where the initial equation is :

$$d_x x + d_y y + d_z z = \rho \tag{23}$$

and where we replace d_x by a linear expression in d_y and d_z. We have then 4 cases to study, depending on the sign of d_y and d_z. The experiments show the interest of this new parameterization.

IV. Experiments

A. Instances

We have performed some experiments on plane and circle detection. All q-intersection variants have been implemented in the `Ibex` interval C++ library [4] (version of January 2015) with its affine relaxation [15] and run on an X86-64 under Linux Ubuntu.

For plane detection, we have generated 9 artificial test cases, with different numbers of points, numbers of planes to detect, and inlier rates (i.e., percentage of points belonging to a plane). All the 3D points are inside a bounded box B. The q points belonging to each plane have been placed near the plane: 2 coordinates are random, and the third one is computed to satisfy the thick equation of the plane. The remaining points have been uniformly randomly generated in B. In order to be close to the real case described further, the planes are made of 2 sets of orthogonal planes (i.e., 2 planes in x and y and 2 planes in y and z for instances P_1 to P_3).

Table I presents the characteristics of the studied test cases.

Table I
Characteristics of the artificial plane detection test cases

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
points	1000	1000	1000	1000	1000	1000	4000	4000	4000
planes	4	4	4	25	25	25	25	25	25
inlier rate	10%	5 %	4 %	2 %	1.5%	1%	2%	1.5%	1%

We have also tried our algorithm on a real 3D point cloud, issued from a part of `scene1` of `velodyndata` from

sites.google.com/site/kevinlai726/datasets [13]. We selected from this urban scene the 529 points labeled in `house40`, a specific building in the scene. This problem is named H_{40} in the tables. We want to find the quasi horizontal and quasi vertical planes of the building façade. We run our algorithm with the parameters $q = 21$, $\tau = 0.001$ and $\epsilon_{sol} = 0.001$.

We made finally an experiment for circle detection in a 2D photo. The data are issued from a buoy detection problem [10] amounting to finding a circle in a submarine photo. We used the data of the paper with 614 points (C_1), and created a (more) noisy problem C_2 with 1228 points, randomly adding to the original problem 614 outliers. The parameters are the center coordinates (x and y) and the radius (r) of the circle to be detected. An observation $\mathbf{o}_i$ is given by the coordinates ox_i and oy_i of the point in the image. The equation is then

$$(x - ox_i)^2 + (y - oy_i)^2 = r^2 \tag{24}$$

We run our algorithm with the parameters $q = 57$, $\tau = 11$ and $\epsilon_{sol} = 0.02$.

B. Results

Table II shows the results in CPU time, with a timeout of one hour, each line corresponding to the introduction of an additional improvement into our algorithm, in the following order:

- Jaulin: the existing interval-based parameter estimation tool
- QinterEstim: the generic algorithm 1
- q-projBis: update of the set of possible observations after the projection algorithm (see Section II-B)
- ded. fwdbwd: use of dedicated forward backward procedures written for plane and circle detection (instead of using HC4-Revise) described in Section III-B
- projNewDim: the additional projection described in Section II-C
- bisect. strat.: the new bisection strategy described in Section III-A
- parameteriz.: the efficient parameterization described in Section III-C (for plane detection only)

We observe that each improvement brings significant speedups. In all, about three orders of magnitude are gained w.r.t. Jaulin et al.'s implementation of `QInterEstim` on all the instances. On Table III, we remark that both the new projection (that allows early rejection of boxes) and the bisection strategy greatly decrease the number of nodes. The new projection is especially efficient for the real plane detection problem (H_{40}), where we obtain a gain of one order of magnitude in time and two orders of magnitude in number of nodes.

Table IV shows the CPU time results when we remove one of the improvements. The first line (no) reports the results of the most advanced algorithm. We observe that each improvement improves the efficiency of our approach.

C. Comparison with RANSAC

We compared our approach with RANSAC. We run the RANSAC algorithm from [16] available online at cg.cs.uni-bonn.de/en/publications/paper-details/schnabel-2007-efficient/. The first line of Table II gives the CPU time and the number of planes found by one run of this algorithm. Since this algorithm is stochastic, several runs give different sets of solutions and even by running it many times, we are never sure to get all the solutions found by our algorithm. For example, we ran RANSAC on P_6: in the first run, 10 planes were detected; after 10 runs, a total number of 21 planes were found; after 15 runs, 23 planes were detected, whereas the 25 planes were found by our algorithm.

Another phenomenon appears with the real point cloud H_{40}. With parameter values q and τ close to ours, RANSAC is able to find only the horizontal planes. Indeed, since it is greedy, the points used in the horizontal planes are removed and cannot be used anymore to find the vertical ones.

V. Discussion

RANSAC is efficient but can miss valid model instances. However, the model instances computed are guaranteed to fit at least q observed data (provided RANSAC were corrected to check the observation constraints using interval arithmetic...). Jaulin et al.'s interval CP approach is deterministic but does not guarantee the answered model instances since no parameters vectors inside the returned boxes are checked. Our extension of Jaulin et al.'s parameter estimation tool is both complete and can guarantee a large fraction of its answered model instances while showing speedups of orders of magnitude. However, the CP parameter estimation tool is still used as a low-level tool, like RANSAC today. In particular, the input τ and q must be tuned manually by the user in real applications. Both parameters are not completely independent and different values of (τ, q) should be tried to better detect different primitives. For instance, if q and τ are too small, several vertical planes, close to each other, can be detected by the algorithm in the H_{40} instance. A straightforward post-processing keeps only the *maximal* model instances, i.e. model instances that fit a superset of the observations used in another one. To conclude, since our approach is deterministic, and the performances are good, it gives hope of a more sophisticated tool able to (semi) automatically adapt (τ, q) for finding the model instances.

References

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. ICLP*, pages 230–244, 1999.

[2] A. Bethencourt and L. Jaulin. 3D Reconstruction Using Interval Methods on the Kinect Device Coupled with an IMU. *Int. Journal of Advanced Robotic Systems*, 10, 2013.

Table II
CPU TIME RESULTS ON PLANE AND CIRCLE DETECTION TEST CASES

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	C_1	C_2
RANSAC	0.1/4	0.1/3	0.1/2	0.3/8	0.4/5	0.6/5	1.4/24	1.4/11	1.8/10	0.2/11		
Jaulin	187.6	1908.1	TO	TO	TO	TO	TO	TO	TO	TO	377.2	3127.5
QInterEstim	72.1	384.2	672.2	3201.1	TO	TO	TO	TO	TO	TO	94.7	286.7
q-projBis	49.6	241.4	425.0	2017.2	TO	TO	TO	TO	TO	TO	81.9	249.6
ded. fwdbwd	7.8	35.1	60.8	313.9	746.9	3015.4	694.1	1432.9	TO	717.3	11.1	33.5
projNewDim	2.8	8.5	14.9	74.1	161.2	570.3	197.8	379.7	932.1	73.5	3.1	12.3
bisect strat.	1.2	3.8	6.5	33.0	71.5	242.4	91.8	178.7	462.0	33.4	2.3	6.0
parameteriz.	1.1	2.4	3.8	18.3	36.4	107.5	53.7	92.5	221.4	24.5		

Table III
NUMBER OF NODES RESULTS ON PLANE AND CIRCLE DETECTION TEST CASES

Test case	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	C_1	C_2
Jaulin	27,810	244,020	TO	TO	TO	TO	TO	TO	TO	TO	88,466	271,546
QInterEstim	30,982	249,268	529,752	4,641,584	TO	TO	TO	TO	TO	TO	108,816	296,034
q-projBis	30,970	249,000	528,370	46,117,178	TO	TO	TO	TO	TO	TO	108,766	268,778
ded. fwdbwd	46,292	316,080	690,582	6,342,124	20,063,224	123,967,652	2,795,762	7,784,852	TO	127.10^6	108,766	268,778
projNewDim	12,802	48,740	102,432	947,052	2,773,000	16,010,328	455,202	1,102,296	4,141,764	1,297,564	34,728	80,752
bisect strat.	4,212	4,070	25,144	181,488	481,822	2,361,030	84,504	163,428	516,496	646,024	30,344	40,600
parameteriz.	4,786	11,626	20,334	122,590	285,898	1,226,748	61,786	111,558	293,834	646,694		

Table IV
CPU TIME WHEN ONE IMPROVING FEATURE IS REMOVED

Removed feat.	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	H_{40}	C_1	C_2
no	1.1	2.4	3.8	18.3	36.4	107.5	53.7	92.5	221.4	24.5	2.3	6.0
QInterestim	7.1	17.1	27.7	231.8	556.7	2887.8	413.3	745.1	1892.5	881.8	10.8	36.0
q-projBis	2.4	8.7	14.8	99.4	232.8	999.8	205.6	385.2	997.6	641.5	3.6	15.7
ded. fwdbwd	13.4	35.7	62.3	305.9	600.6	1783.7	957.6	1636.0	TO	266.0	17.0	58.5
projNewDim	2.0	5.9	9.7	38.8	85.6	286.1	101.9	197.9	519.1	2882.2	10.8	32.7
bisect strat.	2.8	7.9	12.5	63.7	130.4	426.4	179.9	303.6	718.9	59.8	3.1	12.3
parameteriz.	1.2	3.8	6.5	33.0	71.5	242.4	91.8	178.7	462.0	33.4		

[3] C. Carbonnel, G. Trombettoni, P. Vismara, and G. Chabert. Q-intersection algorithms for constrained-based robust parameter estimation. In *Proc. of AAAI 2014*, pages 2630–2636, 2014.

[4] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

[5] P. Chew and K. Marzullo. Masking failures of Multidimensional Sensors. In *Proc. of Reliable Distributed Systems*, pages 32–41, 1991.

[6] H. Collavizza, F. Delobel, and M. Rueher. Comparing Partial Consistencies. *Reliable Comp.*, 5(3):213–228, 1999.

[7] L. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 37(1-4):147–158, 2004.

[8] M. A. Fischler and R. C. Bolles. Random Sampling Consensus: Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. of the ACM*, 24(6):381–395, 1981.

[9] P. Hough. Machine Analysis of Bubble Chamber Pictures. In *Int. Conf. on High Energy Accelerators and Instrumentation*, 1959.

[10] L. Jaulin and S. Bazeille. Image Shape Extraction using Interval Methods. In *Proc. of the 16th IFAC Symposium on System*, pages 378–383, 2009.

[11] L. Jaulin and E. Walter. Guaranteed robust nonlinear minimax estimation. *IEEE Trans. on Automatic Control*, 47, 2002.

[12] L. Jaulin, E. Walter, and O. Didrit. Guaranteed robust nonlinear parameter bounding. In *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, pages 1156–1161, 1996.

[13] K. Lai and D. Fox. Object recognition in 3d point clouds using web data and domain adaptation. *International Journal of Robotics Research*, 29(8):1019–1037, 2010.

[14] F. Messine. *Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution des problèmes avec contraintes*. PhD thesis, LIMA-IRIT-ENSEEIHT-INPT, Toulouse, 1997.

[15] J. Ninin, F. Messine, and P. Hansen. A Reliable Affine Relaxation Method for Global Optimization. *4OR*, pages 1–31, 2014.

[16] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, 26(2):214–226, 2007.