Proceedings of GOW'16, pp. 1 – 4.

A Generic Interval Branch and Bound Algorithm for Parameter Estimation

Bertrand Neveu,[1] Martin de la Gorce[1], Pascal Monasse[1] and Gilles Trombettoni[2]

[1]*LIGM, Ecole des Ponts Paristech, Université Paris Est, France,* Bertrand.Neveu@enpc.fr

[2]*LIRMM, Université de Montpellier, CNRS, France,* Gilles.Trombettoni@lirmm.fr

Abstract The parameter estimation problem is a challenging problem in engineering sciences consisting in computing the parameters of a parametric model that fit observed data. The system is defined by unknown parameters and sometimes internal constraints. The observed data provide constraints on the parameters. The parameter estimation problem is particularly difficult when some observation constraints correspond to outliers and/or the constraints are non convex. For dealing with outliers, the RANSAC randomized algorithm is efficient, but non deterministic, and must be specialized for every problem. In this work, we propose a generic interval branch and bound algorithm that produces a model maximizing the number of observation constraints satisfied within a given tolerance. This tool is inspired by the IbexOpt Branch and Bound algorithm for constrained global optimization (NLP) and is endowed with an improved version of a relaxed intersection operator applied to the observations. The latest version of our B&B follows the Feasible diving strategy to visit the nodes in the search tree. Experiments on a stereovision problem have validated the approach.

Keywords: Interval branch and bound, Parameter estimation, Essential matrix estimation

1. Parameter Estimation

Parameter estimation is a difficult problem widely studied by engineering sciences. It consists in determining the n numerical parameters of a model based on m observations. Calibration or geolocation can be viewed as specific parameter estimation problems.

A parameterized model is defined by an implicit equation $f(\mathbf{x},\mathbf{p})=0$, $\mathbf{p}=(p_1,\ldots,p_n)$ being the n-vector of parameters to be determined. An observation $\mathbf{o}_i$ is a d-dimensional vector of observed data (values) for $\mathbf{x}$. Given a finite set of observations $\{\mathbf{o_1},\ldots,\mathbf{o_i},\ldots,\mathbf{o_m}\}$, we search for a parameter vector that fits the observations, i.e. that satisfies the observation constraints $f(\mathbf{o_i},\mathbf{p})=0$. Because the model is not perfect or due to uncertainties on the observations $\mathbf{o_i}$, there exists generally no model fitting all the observation constraints exactly. In addition, the answer of least square methods or numerical local methods (for nonlinear observation constraints) is poor in the presence of *outliers*. Outliers can have numerous origins, including extreme values of the noise, erroneous measurements and data reporting errors.

That is why we consider an approach where the observation constraints are handled within a tolerance value τ:

$$-\tau \le f(\mathbf{o}_i,\mathbf{p}) \le +\tau.$$

The observations are partitioned into a group of outliers and a group of *inliers*. The inliers correspond to the *consensus set* ($Consensus(\mathbf{p})$), i.e. the set of observations compatible with $\mathbf{p}$:

$$Consensus(\mathbf{p}) = \{\mathbf{o}_i \mid -\tau \le f(\mathbf{o}_i,\mathbf{p}) \le +\tau\}. \tag{1}$$

Outliers do not satisfy the corresponding relaxed observation constraints.

The problem is to compute a model fitting a maximum number of observations. More precisely, a parameterized model can be defined by the observation constraints and internal constraints between parameters $C(\mathbf{p})$, so that the estimation parameter problem handled by the interval Branch and Bound proposed in this paper is defined as follows:

$$\arg\max_{\mathbf{p}}(card(Consensus(\mathbf{p}))) \;\; s.t. \;\; C(\mathbf{p}). \tag{2}$$

2. Parameter Estimation Fitting a Maximum Number of Inliers

Interval arithmetic and contractors

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 $\overline{x_{max}} - \underline{x_{max}}$ of its largest dimension x_{max}. The *perimeter* of a box, given by $\sum_i \overline{x_i} - \underline{x_i}$, is another size measurement.

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 parameter vectors that are compatible with that observation, i.e. $V_i = \{\mathbf{p} \mid -\tau \leq f(\mathbf{o}_i, \mathbf{p}) \leq +\tau\}$. The set of parameter values inside this 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. An important contractor used in our optimization code is the well-known `HC4-revise` [1, 2], also called forward-backward. This contractor traverses twice the expression tree corresponding to a given constraint to contract the domains of its variables.

Another contractor is applied to the m boxes obtained after having contracting the box studied using the m observation constraints. The q-intersection operator relaxes the (generally empty) intersection of the m boxes by the union of all the intersections obtained with q boxes ($q < m$). 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.*

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, called here `q-proj`, that solves the problem on each dimension independently [4]. The overall complexity of `q-proj` is $O(nm\log(m))$.

Interval Branch and Bound for Parameter Estimation

We have designed an interval B&B algorithm for parameter estimation that computes a model maximizing the consensus of a parameterized model. It appeared that the standard best-first search strategy has difficulties to find a good solution and that a depth first search strategy gets stuck in a local optimum area. Therefore we have used a mixed strategy based on best-first search where, at each node, a *feasible diving* procedure is performed [5].

The main `EstimB&B` algorithm returns the best solution found (*best* is the node where the best solution has been found), with a number $q_{\min}$ of validated inliers, and an interval $[q_{\min}, q_{\max}]$. Either the precision Δ_{obj} required on the number of inliers is obtained, i.e. $q_{\max} - q_{\min} \leq \Delta_{obj}$, or $q_{\max} - q_{\min} > \Delta_{obj}$, which means that "small" boxes with width inferior to ϵ_{sol} remain open (unexplored) with a number of inliers at most equal to $q_{\max}^{\epsilon_{sol}}$. The `EstimB&B` algorithm achieves a best-first search by maintaining a set of open nodes *nodeHeap*. This set is implemented by a heap data structure having at its top the node with the largest number of possible inliers (i.e., an upper bound $q_{\max}$ of the objective function value to maximize). `EstimB&B` selects iteratively the node having the greatest q_{max} and calls the `FeasibleDiving` function on it.

Feasible diving. The `FeasibleDiving` function is called with the model as parameter: the parameters $\mathbf{p}$ of the model, i.e. the variables to be determined using the internal constraints

C and the observation constraints $C_{obs} = \{-\tau \leq f(\mathbf{o}_i, \mathbf{p}) \leq +\tau\}$. From the selected *node*,

```
FeasibleDiving ($node$, $C$, $C_{obs}$, $best$, $nodeHeap$, $\epsilon_{sol}$, $q_{max}^{\epsilon_{sol}}$)
while $node.box \neq \emptyset$ and $width(node.box) > \epsilon_{sol}$ and not $node.isLeaf$ do
    ($node_1$, $node_2$) ← Bisect ($node$)        // Bisection of $node.box$
    ($node_1$, $best$, $q_{max}^{\epsilon_{sol}}$) ← Contract&Bound ($node_1$, $C$, $C_{obs}$, $\epsilon_{obj}$, $\Delta_{obj}$,$best$,$q_{max}^{\epsilon_{sol}}$)
    ($node_2$, $best$, $q_{max}^{\epsilon_{sol}}$) ← Contract&Bound ($node_2$, $C$, $C_{obs}$, $\epsilon_{obj}$, $\Delta_{obj}$,$best$,$q_{max}^{\epsilon_{sol}}$)
    ($node_{better}$, $node_{worse}$) ← Sort ($node_1$, $node_2$)
    if $node_{worse}.box \neq \emptyset$ and $width(node_{worse}.box) > \epsilon_{sol}$ and not $isLeaf$ then
        $nodeHeap \leftarrow Push(node_{worse}, nodeHeap)$
    $node \leftarrow node_{better}$
return ($nodeHeap$, $best$, $q_{max}^{\epsilon_{sol}}$)
```

Algorithm 1: The *Feasible Diving* procedure run at each node of our interval B&B algorithm.

`FeasibleDiving`, described in Algorithm 1, builds a tree in depth-first order and keeps only the most promising node at each iteration. More precisely, the box in the node is bisected along one dimension and the two sub-boxes are handled by a `Contract&Bound` procedure. The sub-box with the largest $q_{\max}$ is handled in the next step while the other sub-box is pushed into the heap *nodeHeap* of open nodes. Thus, `FeasibleDiving` is a greedy algorithm that does not perform any backtracking.

Contract and Bound. The `Contract&Bound` function handles every open node. It contracts the node box and tries to prove (guarantee) that more than $best.q_{\min} + \Delta_{obj}$ observation constraints are satisfied inside the box. Recall that the best node found so far has at least $q_{\min}$ valid inliers, so that the next best node (if any) is searched for with a cost at least equal to $best.q_{\min} + \Delta_{obj} + 1$.

1. The box is first contracted by the constraints C between parameters, if any: first, the HC4 constraint propagation algorithm is performed. We add calls to a linear programming solver for improving variable interval bounds, using a linear relaxation of the constraints based on affine arithmetics [6].

2. The q-intersection projection algorithm, using $q = best.q_{\min}+\Delta_{obj}+1$ is then called on the n parameter directions. We also perform a q-projection on an additional direction where we hope to obtain small intervals, thus favoring a failure of the q-intersection. To this end, we linearize and relax every observation constraint, and project the parallelograms obtained on the direction corresponding to the mean normal vector of the "parallelogram" gradients. Details can be found in [7].

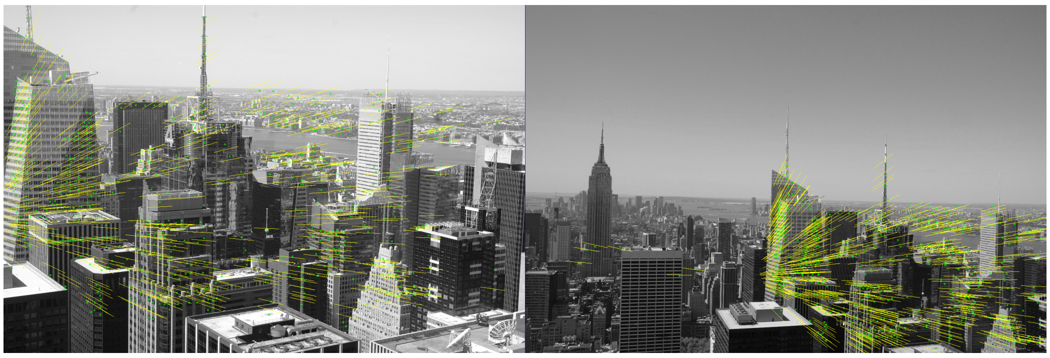
3. An upper bound $node.q_{\max}$ of the number of inliers in the node is given by the minimum over all dimensions of the maximum number of intersected intervals found by the q-projection procedure in a dimension. If $node.q_{\max} < q$, then the node box becomes empty and the corresponding branch in the tree will be pruned.

4. To improve the lower bound, one has to find a feasible point in the current box, i.e. a point that satisfies all the constraints C, if any, and a greater number of observation constraints C_{obs} than the current solution. When C is not empty, the search of the solution validating inliers is a difficult task in itself. We have designed a variant of an algorithm used in `IbexOpt` (see [8]) that tries to build an *inner polytope* inside the box where all the points satisfy the constraints C. If such a polytope is built, a point (vertex) is picked inside it and we compute the number of inliers at this feasible point for updating the new current solution.

In conclusion, with the implementation of our `EstimBB` code, we are close to provide a generic tool for parameter estimation taking into account outliers.

3. Experiments

Our `EstimBB` code has been implemented in the Interval Based EXplorer (`Ibex`) [9], a free C++ library devoted to interval computing. We made preliminary experiments on the shape (plane or circle) detection problem studied in [7], replacing the search of all solutions with at least q inliers by the optimization problem presented in this paper. These problems have a parameter space of dimension $n = 3$ and have no additional (internal) constraints. Second, we tried to solve the more challenging problem of essential matrix estimation in stereo vision having $n = 5$ parameters. We tested our method on the 6 real instances for essential matrix estimation from USAC library [10], named *test1* to *test6*, with between $m = 534$ and $m = 2245$ pairs of 2D points. The *test2*, ..., *test5* instances are solved to optimality within a 1% precision on the number of inliers w.r.t the total number of correspondences in several hours. The *test6* instance seems more difficult because the percentage of inliers is small, but we succeeded in computing a model to optimality by solving a partial instance with 320 correspondences. The *test1* instance is the most difficult one, since the tolerance is smaller and one has to go deeper in the search tree to validate the solution. The algorithm did not terminate in 100,000 seconds. The figure below shows the set inliers found on *test5*.



4. Conclusion

This paper has presented an exact interval B&B approach implemented in IBEX for parameter estimation taking into account outliers. The corresponding code is generic. First experiments on different computer vision problems suggest that the current interval B&B algorithm provides good results in medium dimension, i.e. up to 4 or 5 in number of parameters (depending on the inlier rate), whereas most exact approaches cannot cope with dimension 3.

References

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising Hull and Box Consistency. In *Proc. of International Conference on Logic Programming (ICLP)*, pages 230–244, 1999.

[2] 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.

[3] C. Carbonnel, G. Trombettoni, P. Vismara, and G. Chabert. Q-Intersection Algorithms for Constrained-Based Robust Parameter Estimation. In *Proc. AAAI*, pages 2630–2636, 2014.

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

[5] B. Neveu, G. Trombettoni, and I. Araya. Node Selection Strategies in Interval Branch and Bound Algorithms. *Journal of Global Optimization*, 64(2):289–304, 2016.

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

[7] B. Neveu, M. de la Gorce, and G. Trombettoni. Improving a Constraint Programming Approach for Parameter Estimation. In *In Proc. ICTAI*, pages 852–859, 2015.

[8] I. Araya, G. Trombettoni, B. Neveu, and G. Chabert. Upper Bounding in Inner Regions for Global Optimization under Inequality Constraints. *J. Global Optimization (JOGO)*, 60(2):145–164, 2014.

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

[10] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. USAC: A Universal Framework for Random Sample Consensus. *IEEE Trans. PAMI*, 35(8), 2013.