Interval-based Modelling with Constraints Propagation

Agostino Dovier, Michela Farenzena, and Andrea Fusiello

[1] Univ. di Udine (Italy), Dip. di Matematica e Informatica.
`dovier@dimi.uniud.it`
[2] Univ. di Verona (Italy), Dipartimento di Informatica.
`(farenzen|afusiell)@sci.univr.it`

Abstract. In this paper we show how Interval Analysis and Constraint (logic) Programming on intervals can be used to obtain an accurate geometric model of a scene that rigorously takes into account the propagation of data errors and roundoff. Image points, captured by a camera, are represented as small rectangles. As a consequence, the output of an n-views triangulation is not a single point in space, but a polyhedron that contains all the possible solutions. Interval Analysis is used to bound this polyhedron with a box. Geometrical constraints such as orthogonality, parallelism, and collinearity are subsequently enforced in order to reduce the size of those boxes, using constraint programming. Experiments with real calibrated images illustrate the effectiveness of the approach.

1 Introduction

Triangulation consists in recovering the coordinates of a point in three-dimensional (3D) space given its images in two or more known cameras. In the absence of errors, this problem is trivial, involving only finding the intersection of rays in the space. If data are perturbed, however, the rays corresponding to back-projections of the image points do not intersect in a common point, and obtaining the best estimate of the 3D point is not a trivial task. In [7] the problem is solved for the case of two views, taking advantage of the epipolar constraint and involving the solution of a sixth-degree polynomial. However, the method is not generalisable to more than two views. For the n views case, a simple algebraic method exists [4], but the value being minimised has no geometric meaning, so the method is not reliable: a minimisation of a suitable (non-linear) cost function, like the re-projection error in the image plane, should be performed to achieve better accuracy [4, 14]. An alternative approach is to find the closest point in 3D space to the rays back-projected from the image points. In the case of two views, this is the mid-point of the common perpendicular to the two rays. However, it is known that this method fails badly in the case where the rays are almost parallel, corresponding to a point near infinity, since in this case the computed point will be close to the point half-way between the two camera centres. In [6] a infinite-norm (L_∞) minimisation of the re-projection error is explored. Using the L_∞ cost function is significantly simpler, and computationally faster, than the two-norm (L_2) cost, but the method is extremely not robust.

In this paper, instead of selecting one "best" solution, as customary, we aim at describing the set of *all* the possible solutions, given a bounded error affecting the image points. In practice, points are modelled as 2D intervals, and the *solution set* is defined as the set of all the 3D points that can be obtained as the intersection of two conjugate points contained in the 2D intervals. Interval Analysis is used to obtain a box that encloses the solution set.

Interval Analysis (IA) [12] is an approach to the solution of numerical problems by performing computations on sets of reals rather than on floating point approximations to reals. It was firstly introduced for bounding the measurement errors of physical quantities for which no statistical distribution was known.

There are two principal advantages of IA over classical numerical analysis. The first is that the input errors and the roundoff errors are automatically incorporated into the result interval. Thus, interval evaluation can be viewed as automatically performing both a calculation and an error analysis. The second is that IA allows one to compute provably correct upper and lower bounds on the range of a function over an interval, and this proves useful in the construction of verifiable *constraint solvers*, which return intervals that are guaranteed to contain all the real solutions.

Adhering to the IA paradigm, we do not model a probability distribution inside the intervals, therefore there is not a preferred solution in the solution set. Nevertheless, a pointwise solution is needed to make the 3D model usable for other applications. One could choose at random a point inside the intervals. Yet, in many cases, geometric constraints such as orthogonality, parallelism, co-planarity and others can be imposed on the reconstructed structure. The idea is to formalise this as a constraint satisfaction problem, use Constraint Programming on Intervals to narrow the intervals as much as possible, thus approximating a reasonable pointwise solution.

Constraint Logic Programming (CLP) languages (c.f., e.g., [8, 1]) are declarative programming languages developped to deal with various classes of constraints. Let us refer to the language of this family designed to handle constraints over intervals of reals as $CLP(\mathsf{Intervals})$. In this context, a Constraint Satisfaction Problem (CSP) is modelled by assigning admissible intervals of values and constraints to each variable. A built-in constraint solver then contracts these intervals preserving the solution set. The output of the constraint solver is an (hopefully little) interval associated to each variable of the problem, and we can choose any value of the variables in their intervals as a reasonable approximation of the solution to the initial problem. As $CLP(\mathsf{Intervals})$ language, we have used the library `ic` of the ECLiPSe 5.8 system [2]. As shown in Section 6, the results are very encouraging.

2 Problem formulation

Let P_i, $i = 1, \dots, n$ be a sequence of n known cameras and m_i be the image of some unknown point M in 3D space, both expressed in homogeneous coordinates. Thus, we write $\kappa m_i = \mathrm{P}_i M$, where κ is the depth of M wrt the camera. The problem of computing the point M given the camera matrices P_i and the image points m_i is known as the *triangulation* problem. In the absence of errors, this problem is trivial, involving only finding the intersection point of rays in the space. When data are perturbed by errors, however, the rays corresponding to back-projections of the image points do not intersect in a common point, therefore only an approximate solution can be defined. This approximation can be circumvented if one refrains from searching for *one* solution and compute instead a *set* of solutions that contains the error-free solution and can be defined precisely in terms of the error affecting the image points.

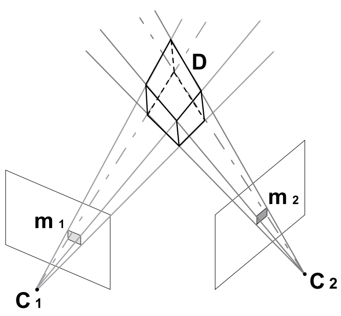

Fig. 1. Interval-based triangulation.

In the case of two views, assuming that errors are bounded by rectangles B_1 and B_2 in the images, the *solution set* of triangulation is a diamond-shaped polyhedron D_2 as in Fig. 1. Geometrically, D_2 is obtained by intersecting the two semi-infinite pyramids defined by the two rectangles B_1 and B_2 and the respective camera centres.

In the general case of n views, the solution set is defined as the polyhedron formed by the intersection of the n semi-infinite pyramids generated by the intervals $B_1, \dots B_n$. Analytically, this region is defined as the set

$$D_n = \{M \,:\, \forall i = 1, \dots, n \, \exists m_i \in B_i \text{ s.t. } m_i \simeq \mathrm{P}_i M\},$$

where $\simeq$ denotes equality up to a scale factor. In the following section we will show how the solution set can be enclosed with an axis-aligned box using Interval Analysis.

These 3D polyhedra are the best piece of information about the localisation of the error-free 3D point one can deduce from the bounded correspondences. In order to narrow the solution set we must include additional information, possibly automatically deducible from images. Thus, we seek for geometric constraints, such as orthogonality and parallelism, between scene primitives that are able to reduce the solution set. If we add a sufficient set of constraints we will be able to isolate, ideally, a single solution.

The constraints considered in this paper are orthogonality, parallelism, collinearity, and equality of distances or angles. They are extracted semiautomatically from the 3D model obtained by the interval-based triangulation. A pointwise solution is derived by constraint propagation, as we will discuss in Section 5.

3 Interval Analysis

Interval Analysis [12] is an arithmetic defined on intervals, rather than on real numbers. In the sequel of this section we shall follow the notation used in [11], where intervals are denoted by boldface. Underscores and overscores will represent respectively lower and upper bounds of intervals. The midpoint of an interval $\boldsymbol{x}$ is denoted by $\mathrm{mid}(\boldsymbol{x})$. $\mathbb{IR}$ and $\mathbb{IR}^n$ stand respectively for the set of real intervals and the set of real interval vectors of dimension n. If $f(x)$ is a function defined over an interval $\boldsymbol{x}$ then $\mathrm{range}(f, \boldsymbol{x})$ denotes the range of f over $\boldsymbol{x}$.

If $\boldsymbol{x} = [\underline{x}, \overline{x}]$ and $\boldsymbol{y} = [\underline{y}, \overline{y}]$, a binary operation between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined in interval arithmetic as:

$$\boldsymbol{x} \circ \boldsymbol{y} = \{x \circ y \mid x \in \boldsymbol{x} \wedge y \in \boldsymbol{y}\}, \forall \circ \in \{+, -, \times, \div\}.$$

Operationally, interval operations are defined by the min-max formula:

$$\boldsymbol{x} \circ \boldsymbol{y} = \left[\min\left\{\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}\right\},\ \max\left\{\underline{x} \circ \underline{y}, \underline{x} \circ \overline{y}, \overline{x} \circ \underline{y}, \overline{x} \circ \overline{y}\right\}\right] \quad (1)$$

Thus, the ranges of the four elementary interval operations are exactly the ranges of the corresponding real operations.

The above definitions imply the ability to perform the four operations with arbitrary precision. When implemented on a digital computer, however, truncation errors occur that may cause the resulting interval not to contain the true result. In order to preserve the guarantee that the true value always lie within the interval, end-points of the interval must be rounded *outward*, i.e., the lower endpoint of the interval must be rounded down and the upper endpoint must be rounded up.

In general, interval computation cannot produce the exact range of a function, but only approximate it.

Definition 1 (Interval extension). *[10] A function* $\boldsymbol{f} : \mathbb{IR} \to \mathbb{IR}$ *is said to be an* interval extension *of* $f : \mathbb{R} \to \mathbb{R}$ *provided* $\mathrm{range}(f, \boldsymbol{x}) \subseteq \boldsymbol{f}(\boldsymbol{x})$ *for all intervals* $\boldsymbol{x} \subset \mathbb{IR}$ *within the domain of* $\boldsymbol{f}$*.*

This property is particularly suited for error propagation: if $\boldsymbol{x}$ bounds the input error on the variable x, $\boldsymbol{f}(\boldsymbol{x})$ bounds the output error. Therefore, if the exact value is contained in interval data, the exact value will be contained in the interval result. This approach is different from the established techniques for error propagation [4, 5, 9], mainly based on statistical analysis: a statistical distribution of the error need not to be assumed, and the result is mathematically guaranteed to contain the exact value.

Operationally, a straightforward interval extension is defined as follows:

Definition 2 (Natural interval extension). *Let us consider a function f computable as an arithmetic expression f, composed of a finite sequence of operations applied to constants, argument variables or intermediate results. A* natural interval extension *of such a function, denoted by $\mathbf{f}(\boldsymbol{x})$, is obtained by replacing variables with intervals and executing all arithmetic operations according to the rules (1).*

Similar definitions apply for interval vectors (or boxes) in $\mathbb{IR}^n$. Some points are worth noting:

- Different expressions for the same function yield different natural interval extensions. For instance, $\mathbf{f_1}(\boldsymbol{x}) = \boldsymbol{x}^2 - \boldsymbol{x}$, and $\mathbf{f_2}(\boldsymbol{x}) = \boldsymbol{x}(\boldsymbol{x} - 1)$ are both natural interval extensions of the same function.
- Variable dependency: Evaluating the expression $\mathsf{f}(x) = x - x$ with the interval [1,2], the result is $\mathbf{f}([1,2]) = [1,2] - [1,2] = [-1,1]$, not 0, as expected, because the piece of information that the two intervals represent the same variable is lost.
- Overestimation: Although the ranges of interval arithmetic operations are exact, this is not so if operations are composed. For example, if $\boldsymbol{x} = [0,1]$ we have $\mathbf{f_2}(\boldsymbol{x}) = [0,1]\,([0,1] - 1) = [0,1]\,[-1,0] = [-1,0]$, which strictly includes $\text{range}(\mathsf{f}_2, [0,1]) = [-1/4, 0]$. This effect arises as a consequence of the previous two.
- Wrapping effect: This is a phenomenon intrinsic to interval computation in $\mathbb{R}^n$, namely the fact that the image of a box $\boldsymbol{x}$ under a map $F : \mathbb{R}^n \to \mathbb{R}^n$ is not a box, in general. Interval computation can yield, at best, the *interval hull* of the range $\text{range}(F, \boldsymbol{x})$, i.e. the smallest box containing $\text{range}(F, \boldsymbol{x})$.

These drawbacks have nourished the idea that Interval Analysis gives results too pessimistic to be useful, yet a careful use of the tools of IA can provide realistic bounds.

4 Interval-based triangulation

Given the camera matrices P_1 and P_2, let m_1 and m_2 be two corresponding points. It follows that m_2 lies on the epipolar line of m_1 and so the two rays back-projected from image points m_1 and m_2 lie in a common epipolar plane. As they lie in the same plane, they will intersect at some point. This point is the reconstructed 3D scene point M.

The equation of the epipolar line can be derived from the equation describing the optical ray of m_1:

$$M = \begin{pmatrix} -\mathrm{P}_{3\times3,1}^{-1}\mathrm{P}_{\cdot\,4,1} \\ 1 \end{pmatrix} + \lambda \begin{pmatrix} \mathrm{P}_{3\times3,1}^{-1} m_1 \\ 0 \end{pmatrix}, \quad \lambda \in \mathbb{R}, \tag{2}$$

where $\mathrm{P}_{3\times 3,1}$ is the matrix composed by the first three rows and first three columns of P_1, and $\mathrm{P}_{\cdot\,4,1}$ is the fourth column of P_1. The epipolar line corresponding to m_1 represents the projection of the optical ray through m_1 onto the image plane 2:

$$\kappa m_2 = e_2 + \lambda m_1' \tag{3}$$

where

$$e_2 = \mathrm{P}_2 \begin{pmatrix} -\mathrm{P}_{3\times 3,1}^{-1}\mathrm{P}_{\cdot\,4,1} \\ 1 \end{pmatrix} \text{ and } m_1' = \mathrm{P}_{3\times 3,2}\mathrm{P}_{3\times 3,1}^{-1} m_1 .$$

Analytically, the reconstructed 3D point M can be found using Equation (3), by solving for parameters κ and λ, using the following closed form expressions [3]:

$$\begin{aligned} \frac{1}{\kappa} &= \frac{(m_2 \times m_1') \cdot (e_2 \times m_1')}{||e_2 \times m_1'||^2}, \\ \frac{\lambda}{\kappa} &= \frac{(m_2 \times e_2) \cdot (m_1' \times e_2)}{||m_1' \times e_2||^2}. \end{aligned} \tag{4}$$

The coordinates of M are then obtained by inserting the value λ into Equation (3). After doing all the substitutions, an expression is obtained that relates the reconstructed point to the two conjugate image points:

$$M = \mathsf{f}(m_1, m_2) \tag{5}$$

If we let m_1 and m_2 vary in B_1 and B_2 respectively, then range$(\mathsf{f}, B_1 \times B_2)$ describes the solution set D_2 (Fig. 1). Interval Analysis gives us a way to compute an axis-aligned bounding box containing D_2 by simply evaluating $\mathbf{f}(\boldsymbol{m}_1, \boldsymbol{m}_2)$, the natural interval extension of f, with $B_1 = \boldsymbol{m}_1$ and $B_2 = \boldsymbol{m}_2$. IA guarantees that if the conjugate intervals $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ contain the exact point correspondences, then the interval result contains the exact (i.e. error-free) 3D reconstructed point.

It may be worth noting that the result is not to be interpreted in a probabilistic or fuzzy way: no assumption is made on error statistical distribution, hence no point inside the resulting 3D interval is more probable or more important than others.

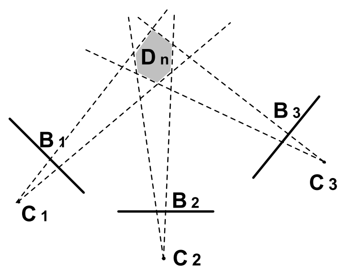

Fig. 2. n-views interval-based triangulation.

This approach is easily extendible to the general case. As defined in Section 2, the solution set of triangulation is the 3D polyhedron formed by the intersection of the semi-infinite pyramids generated by back-projecting in space the intervals $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_n$ (Fig. 2). Thanks to the associativity of intersection, D_n can be obtained by first intersecting pairs of such pyramids and then intersecting the results. Let $D_2^{i,j}$ be the solution set of the triangulation between view i and view j. Then:

$$D_n = \bigcap_{\substack{i=1,\ldots,n \\ j=i+1,\ldots,n}} D_2^{i,j}. \tag{6}$$

An enclosure of the solution set D_n is obtained by intersecting the $n(n-1)/2$ enclosures of $D_2^{i,j}$ computed with the method described in Sec. 4. Since each enclosure contains the respective solution set $D_2^{i,j}$, their intersection will contain D_n. Similarly, as the error-free solution is contained in each $D_2^{i,j}$, then it must be contained in D_n as well.

5 Constraints Propagation

The interval-based triangulation yields boxes in 3D space, each representing the error-free 3D point bounded by its error. Given the connectivity information between these points, geometric constraints can be deduced, such as orthogonality, parallelism, collinearity, equality of lengths and angles. They will be used to narrow the boxes and to obtain a pointwise solution.

As previously said, this problem is suitable for being formalised as a constraint satisfaction problem: each reconstructed box is associated to an interval variable, and these variables must satisfy the aforementioned geometric constraints. Finding the solution of this problem is committed to the ECLiPSe constraint solver, with the built-in predicate `locate` [2]. This predicate requires a precision parameter, that specifies the width of the interval solution. Since computation time grows as this parameter gets smaller, the level of precision attainable in a reasonable time strictly depends on the size of the problem's instance, namely the number of variables and the number of constraints.

We address this issue with an incremental approach: instead of solving the original problem, we solve a sequence of increasingly bigger problems, where constraints and points are added incrementally, from the outline of the structure to the details. In our experiments, this allowed us to reach a good accuracy in a fair amount of time. The drawback is that the results depend on the order in which constraints are added, and this decision is left to the user, in the current implementation.

The solution achieved by ECLiPSe is composed by intervals, though very thin. We finally attain a point solution taking a random point within each of these intervals. As can be noted in Fig. 3, the pointwise solution may differ from the true one by a small rigid transformation.

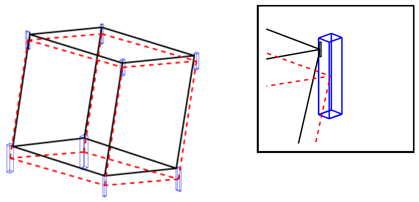

Fig. 3. Ground truth structure (dashed line) and pointwise solution (solid line). Boxes are the result of interval-based triangulation.

6 Experimental results

The triangulation algorithm has been implemented in MATLAB. Thanks to the overloading of arithmetic operations provided by the INTLAB toolbox [13], the implementation was straightforward and the resulting code is simple and compact.

The interval-based triangulation was tested on synthetic data, which consisted of 50 points randomly scattered in a sphere of unit radius, cantered at the origin. Views were generated by placing cameras at random positions, at a mean distance from the centre of 2.5 units with a standard deviation of 0.25. The orientations of the cameras were chosen randomly with the constraint that the optical axis should point toward the centre. The world reference frame was fixed on the first camera. A 2D interval (a square) of a given width was centred onto each image point. The number of views and the interval's width had been varied, and the corresponding average side length of the 3D boxes is reported in Table 1.

	Image interval width [pixel]			
# views	1.0	2.0	3.0	4.0
2	0.056	0.13	0.21	0.30
3	0.014	0.028	0.043	0.056
4	0.010	0.021	0.032	0.041
5	0.0088	0.018	0.026	0.034

Table 1. Average side length of the reconstructed 3D box vs number of views and image intervals width.

Two effects are noticeable in the results: i) the volume of the boxes increases with the width of the image intervals; ii) the volume of the boxes decreases as the number of views increases.

The overall technique (interval-based triangulation and constraint propagation) was tested on real calibrated sequences. We report here the results relative to the *Tribuna* and *Castle* sequences (Fig. 4), consisting of five frames each. We assume that feature points are contained in 2-pixel wide intervals, so the average

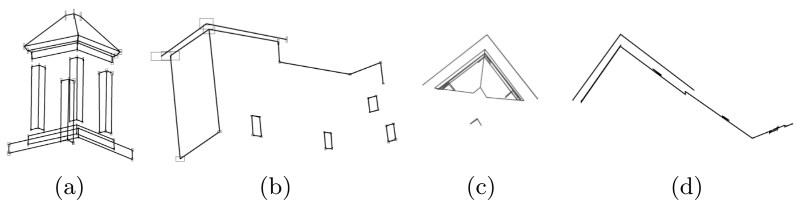
Fig. 4. Interval-based triangulation of *Tribuna* (a) and *Castle* (b). To better visualise the 3D structure, the segments joining the midpoints of the intervals have been drawn. Top views of the 3D models after constraint propagation, (c) and (d).

side length of the 3D boxes obtained by interval-based triangulation is about 10 cm and 60 cm, respectively. With constraint propagation we are able to shrink these boxes up to an average side length of about 3 mm and 6 cm, respectively (about one order of magnitude). The whole process took a few seconds on a Pentium III 500 MHz machine.

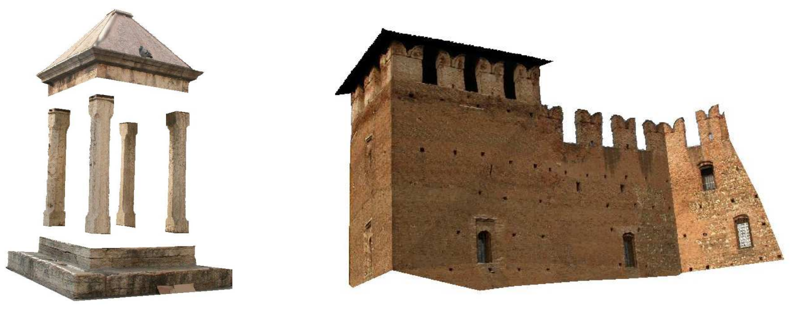
Fig. 5. A textured view of the final 3D model of *Tribuna* (left) and *Castle* (right).

Finally, a pointwise solution is obtained taking a point at random in each box. The top views depicted in Fig. 4 highlight the faithfulness of the structure. The same structure is shown in Fig. 5 as a textured model.

7 Conclusions

In this paper we presented a new approach to scene modelling from many calibrated views based on Interval Analysis and Constraint Logic Programming.

The idea is to take into account errors affecting data explicitly by computing a solution that rigorously includes this error. As a result, the output of triangulation is no more a single 3D point, but a 3D polyhedron that contains all the possible solutions given a bounded perturbation of the conjugate points. Thanks to Interval Analysis, this solution set can be rigorously enclosed with a box in a very simple way.

The width of the solution boxes is then reduced by propagating geometrical constraints using CLP. The final solution is composed by points that are good approximations of the initial problems. Experiments show that in a reasonable time we can achieve a (practically) pointwise solution.

Future work will aim at improving the constraint propagation phase, by automating the sequential approach described in Sec. 5 and/or implementing an *ad-hoc* outer constraint solver.

Acknowledgements Arrigo Benedetti co-authored some papers in the past and introduced the authors to IA. This work have been partially supported by the LIMA3D project (PRIN) and by GNCS2005 project on constraints and their applications.

References

1. K. R. Apt: Principles of Constraint Programming, Cambridge, 2003.
2. A. M. Cheadle, W. Harvey, A. J. Sadler, J. Schimpf, K. Shen and M. G. Wallace: ECLiPSe: An Introduction, IC-Parc, Imperial College London, 2003, Technical Report 03–1.
3. O. Faugeras: What can be seen in three dimensions with an uncalibrated stereo rig, in Proceedings of the European Conference on Computer Vision, 1992, pp 563–578, S. Margherita Ligure.
4. O. Faugeras: Three-Dimensional Computer vision: a geometric viewpoint, 1993, MIT Press, Cambridge, MA.
5. R. M. Haralick: Propagating covariance in Computer Vision, in Workshop on Performance Characteristics of Vision Algorithms, 1996, pp 1-12, Cambridge,UK.
6. R. Hartley and F. Schaffalitzky: L_∞ minimization in geometric reconstruction problems, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2004, pp 504-509, Washington, D.C., USA.
7. R. Hartley and P. Sturm: Triangulation, in Computer Vision and Image Understanding, 1997, November, vol 68, no 2, pp 146-157.
8. J. Jaffar and M. J. Maher: Constraint Logic Programming: A Survey, in Journal of Logic Programming, vol 19-20, pp 503–581, 1994.
9. K. Kanatani: Geometric Computation for Geometric Vision, 1993, Oxford University Press.
10. R. B. Kearfott: Rigorous Global Search: Continuos Problems, 1996, Kluwer.
11. R. B. Kearfott, M. T. Nakao, A. Neumaier, S. M. Rump, S. P. Shary and P. van Hentenryck: Standardized notation in interval analysis, Submitted to Reliable Computing.
12. R. E. Moore: Interval Analysis, Prentice-Hall, 1966.
13. S. M. Rump: INTLAB-INTerval LABoratory, Development in Reliable Computing, Kluwer, 1999, ed T.Csendes.
14. Z. Zhang: Determining the Epipolar Geometry and its Uncertainty: A Review, in International Journal of Computer Vision, 1998, March/April, vol 27, no 2, pp 161-195.