TR－A－0142

Accurate Reconstruction of 3D Scenes from Multiple Imprecise and Uncertain Data

Philippe Quinio

1992. 4. 9

ATR 視聴覚機構研究所

〒619-02 京都府相楽郡精華町光台 2-2 ☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408

Accurate Reconstruction of 3D Scenes from Multiple Imprecise and Uncertain Data

Philippe Quinio

ATR Auditory and Visual Perception Research Laboratories
Advanced Telecommunications Research Institute
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

CONTENTS

Accurate Reconstruction of 3D Scenes from Multiple Imprecise and Uncertain Data

Philippe Quinio

ATR Auditory and Visual Perception Research Laboratories

Advanced Telecommunications Research Institute

2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan

1. CONTEXT AND OBJECTIVE

Three dimensional (3D) maps are often of the utmost importance in robotics, and their accuracy is crucial to the realization of many robotic tasks, such as object recognition, object pose estimation and navigation. The automatic generation of such maps with the sole use of vision is therefore of considerable interest to the field of robotics in general and has great potential for industrial applications.

The map generation problem has been solved *in principle* by the various "structure from X" paradigms (structure from motion, from stereo, from shading etc). We say "in principle" because nearly all the existing methods assume that the parameters of the vision system are known with arbitrary precision and that all information are perfectly certain. As it became obvious that one cannot expect from a user/experimenter/measuring instrument to provide such arbitrarily precise and certain information about the system and its relations to its surrounding world, researchers have started to tackle the issue of *uncertainty representation and management*[4,14].

In this work, we selected binocular stereoscopy as our sensor of 3D information, because it is relatively cheap in hardware, easy to handle and still reasonably accurate due to the fixed architecture of the 2 cameras. Monoscopy or trinocular stereoscopy could be used as well and the former is being investigated by a researcher at ATR (Shinjiro Kawato).

As is well-known in stereo-vision, the map obtained from any single stereo-pair is typically of poor reliability and incomplete due to the limited visual field. The "quality" of the 3D information extracted from a single pair depends on the various parameters (baseline distance, distance from the cameras to the scene etc) and the precision with which they are known, but no matter how well we calibrate our system, there is a limit to the precision we can obtain and there are sources of error that cannot be controlled or easily removed (the sensor noise in the cameras, for instance). Therefore, a method is needed that exploits this noisy information in a certain optimal way without forcing the user/experimenter to provide arbitrarily precise or accurate information regarding the hardware.

It has been widely acknowledged that a key to robustness and precision is to *integrate* the information contained in several image pairs taken at various locations in the scene[1] (see fig.1). Difficulties arise when one realizes that the 3D information extracted from the stereo-pairs is in general both redundant and complementary at the same time: this excludes all set-theoretic integration schemes, such as intersection or union.

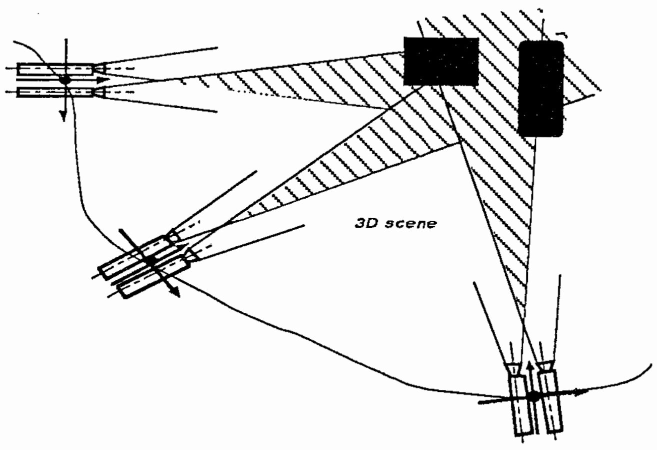


Fig.1. Integration of multiple stereo pairs of images of a 3D scene

We believe that insufficient insight into the sources of error in stereo-vision and the lack of a sufficiently powerful mathematical formalism to model these errors have hampered progress in this direction. We propose here a new method and we show how it leads to increased accuracy of the 3D map of the scene in spite of all the sources of error.

2. THE "RANDOM CLOSED SET" APPROACH

2.1 Imprecision, uncertainty and Random Closed Sets (RACS)

Let us take a closer look at the sources of error in our stereoscopic vision system described in fig.2.

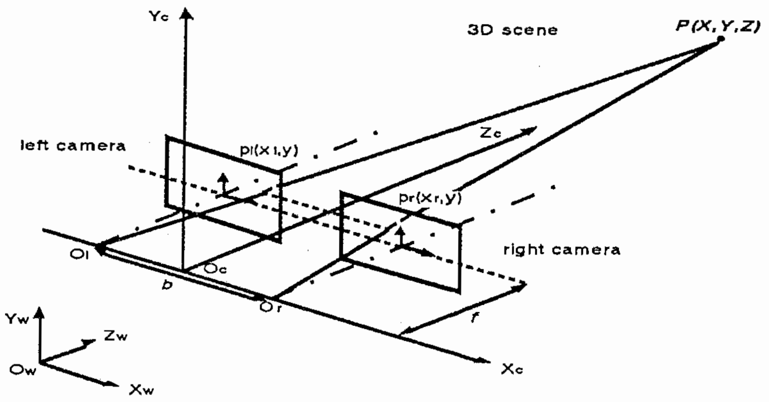


Fig.2. Stereoscopic vision system

One obvious source of error comes from the fact that we work with *digital* images: due to the limited resolution of the cameras, one point in (3D) space does not project onto an infinitely accurate point on the image planes, but onto a *picture element* (pixel), assuming the cameras are focused (if not, the Point Spread Function may spread over adjacent pixels as well).

Another source of error is our limited knowledge about the system geometry: all the parameters are measured using physical instruments and are only known to within a certain precision. Calibration procedures are generally used to measure these parameters, such as the focal length of each camera, the baseline distance between them, the orientation of the two optical axes with respect to world coordinate axes, the vergence angle etc. But no matter how well calibration is performed, errors will not disappear completely. Also, these parameters may deviate or fluctuate from their estimated value as the cameras move around in the scene. For instance, the pitch angle may be affected by small irregularities on the ground, the cameras may shake slightly etc.

The location of the cameras with respect to a fixed origin is also difficult to measure without heavy and costly equipment. In most practical applications, one cannot reasonably expect to measure the location with a precision better than a few millimeters.

All the above errors are *deterministic* in that they are entirely predictable and that one can actually give boundaries inside which the parameters are certain to lie. For instance, one can be certain to know the focal length of each camera to within 10%, or the location to within 1cm. We call these errors *imprecision*, and it is clear from the examples above that imprecision is a set-theoretic concept, i.e. it can be modeled by set theory. It is in fact a generalization of the classical "error intervals" in physics where the imprecision sets need not be intervals of the real line but can be any (closed) subset of a suitable representation space $\mathcal{U}$.

Given a left-right match, i.e. a disparity vector at a certain feature point in the images, one can actually calculate the combined influence of all these sources of imprecision. The resulting 3D closed set quantifies the final imprecision of 3D point locations in the scene (fig.3).

In contrast, some errors in stereo-vision are random in nature: the sensor noise in the cameras, the errors in matching features from the left and right images... We call such errors *uncertainty* because one cannot be certain that the information is not erroneous. Uncertainty is a *probabilistic concept*, i.e. it is modeled within the framework of general Probability Measure theory.

Almost all approaches to uncertainty in computer vision up to now have used random *point* variables, i.e. measurable mappings taking value in the representation space $\mathcal{U}$ itself[1,13]. The reasons are both historical and practical (computational cost) but by no means theoretical: general Probability Measure theory can handle random variables valued in any abstract measurable space.

And indeed, since stereo-vision is both inherently imprecise *and* uncertain, we argue that a *Random Set* theory is needed, i.e. we must work with random variables valued in the power set $\mathcal{P}(\mathcal{U})$ of the representation space. Furthermore, we argued[11] that a notion of *topology* is needed so as to distinguish the experimentally accessible sets from abstract entities of general Set theory whenever the representation space is uncountably infinite. Indeed, it is not possible to experimentally distinguish between a closed interval $[a,b]$ of $\Re$, for instance, and the open interval $]a,b[$ obtained by removing the two boundary values a and b. And although mathematics allows us to construct such abstract entities as a sphere containing only half its boundary while the other half belongs to its complement, we

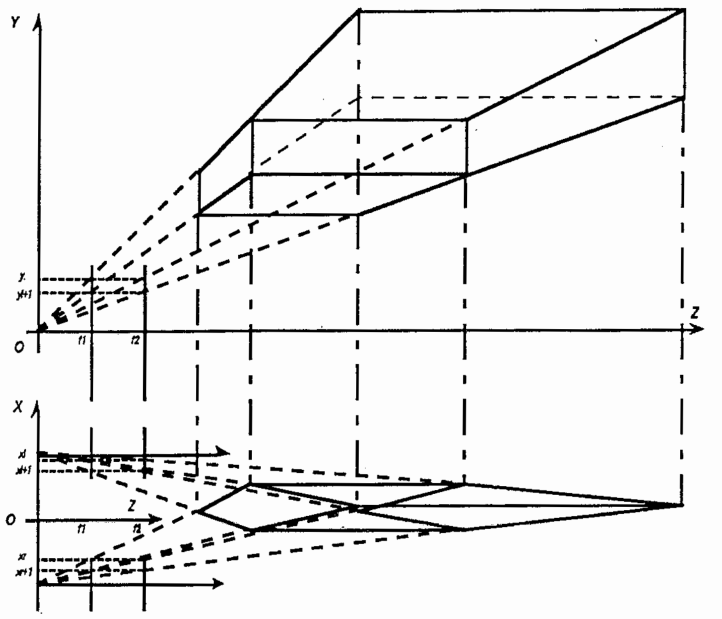


Fig.3. 3D imprecision set resulting from digitization and focal length imprecision ($f_1 \leq f \leq f_2$)

argue that experimenters have no use of such "monsters" and that one should discard them from the set of useful entities. One can explicitly discard all the open sets of a topological space to keep only the closed sets, for instance. What we end up with is precisely the *Random Closed Set* (RACS) theory[5,15,2]. We proposed[11] to use this theory as the general mathematical framework for solving problems that involve both imprecision and uncertainty.

Although Random Closed Sets X are defined rigorously on a σ-algebra of events of the form "X hits O and misses K" for all opens O and compacts K of $\mathcal{U}$, a fundamental result, known as the Choquet theorem[15], insures that X is in fact entirely determined by its hitting functional T_X : $T_X(K) = \mathbf{Prob}(X \text{ hits } K)$, which is a Choquet capacity of infinite order defined on the *compact* subsets K of $\mathcal{U}$. This is very nice since in practice we do not want to deal with all the open sets, which are not even bounded in general!. What the Choquet theorem tells us is that it is *sufficient* to work with the compact subsets of $\mathcal{U}$.

We have investigated[9,11] the connections between the RACS theory and other formalisms commonly used in Artificial Intelligence (AI), and found a very strong link with the Dempster-Shafer theory, also known as the theory of Belief functions. Indeed, the hitting capacity of an almost surely non-empty RACS is a Plausibility functions in the sense of Shafer[16]. This connection will be particularly useful to interpret our results in terms of Plausibility and Belief and to extend the approach to AI problems, including object recognition in the presence of uncertainty.

2.2 Data RACS and visual RACS

For every disparity vector in a stereo-pair (L_i, R_i) obtained at location i, we compute the 3D imprecision set from the known sampling step (pixel size), imprecision in the geometry parameters and in the system location. The union of all these 3D sets for all disparity vectors obtained in stereo-pair i is the Data Set X_i for this pair. Also associated with location i is the *visual field* V_i, i.e. the set of all the points in the scene that are visible from both cameras. As we (randomly) move our system around in the scene, we get a number of Data Sets $\{X_i\}_{i=1}^n$ and corresponding visual fields $\{V_i\}_{i=1}^n$. We assume that the latter are realizations of a *visual* RACS V and the former of a *data* RACS X. Note that $X \subset V$.

2.3 An octree of structuring elements

To estimate the probability distribution of X, the Choquet theorem tells us that it suffices to estimate the probability that it hits (intersects) any compact subset of $\mathcal{U}$. Since the features are 3D points, $\mathcal{U} = \Re^3$ and the compact subsets are the closed bounded subsets in the Euclidean space.

Since in general there are an (uncountably) infinite number of compacts K in $\mathcal{U}$, we cannot expect to determine X entirely in the sense of the Choquet theorem. Instead, we select a finite family of compacts that play the role of *structuring elements* and "probe" our data with these analysis tools, as suggested by Mathematical Morphology[15]. The choice of these structuring elements is an active process that requires the commitment of the experimenter/engineer. Serra[15] classified the main structuring elements of $\Re^3$ according to their convexity and isotropy properties. Closed balls in $\Re^3$ are natural elements to probe a priori isotropic data and are therefore good candidates for our problem. They do not allow efficient calculations on computers though and we preferred cubes to balls, in spite of their anisotropy.

It is natural to include in this family compacts sets of different sizes, as this will allow a *multiscale* analysis of our data. We opted for a tree of 3D closed cubes, called *octree*, as it allows for a very concise representation of the space[3], especially when the scene is "almost empty", which is often the case in practice. Let $\{C_j\}$ be the octree of cubes.

2.4 Statistical estimation

What we directly estimate from X_i and V_i is the *joint probability that X hits C_j and that C_j is visible* from the (random) robot location:

$$\mathbf{Prob}(X \text{ hits } C_j \text{ and } C_j \subset V) \tag{1}$$

However, we are interested in the location of scene features independently of the particular motion of our robot, i.e. we are interested in the probability that X hits a visible C_j: $\mathbf{Prob}(X \text{ hits } C_j | C_j \subset V)$. This is readily obtained by:

$$\mathbf{Prob}(X \text{ hits } C_j | C_j \subset V) = \frac{\mathbf{Prob}(X \text{ hits } C_j \text{ and } C_j \subset V)}{\mathbf{Prob}(C_j \subset V)} \tag{2}$$

which can be statistically estimated by dividing the number of times the Data Sets X_i hit C_j by the number of times C_j is included in V_i.

The resulting estimated distribution is then compared to a threshold τ quantifying the a priori amount of uncertainty, i.e. reflecting the overall quality of the matching procedure. τ is a constant of the vision system and can be determined beforehand independently. Obviously, if matching is perfect, $\tau = 1$ and the previous approach merely extracts the intersection of the data sets (up to the resolution of the octree), hence providing the *strongest integration*. In the extreme case when matching is completely random, $\tau = 0$ and the previous approach yields the union of all the data sets, i.e. their ***weakest integration*** (which is the best we can do). Between these two cases, the data sets are combined in such a way that the influence of any erroneous matching will statistically tend to zero while the sets generated by correct correspondences will tend to accumulate in space.

3. EXISTING APPROACHES TO UNCERTAINTY MANAGEMENT

A recent survey on multisensor integration that tackles the issue of uncertainty representation in intelligent systems can be found in [4].

3.1 Uncertainty ellipsoids

"Set membership" approaches[8,14] model uncertainty using simple convex sets of the Euclidean space, such as hyperellipsoids or strips (the latter being unbounded). The advantage of using those simple shapes is that one can represent them using only a few parameters.

The key assumption is that the actual points lie in their respective "uncertainty set" with ***absolute certainty.*** If this is the case, then a simple ***intersection*** operation should be suitable to integrate the information. However, since the intersection of two ellipsoids or even two strips need not be an ellipsoid nor a strip, computational complexity makes the problem intractable if the number of data sources is large. To avoid this problem, algorithms have been devised that compute a "tight uncertainty bound" providing a reasonable approximation of the intersection.

The connection with our RACS approach is clear: what the authors[14] call "uncertainty" is labeled "imprecision" in our work, and this "set membership method" simply assumes that there is no uncertainty at all, but only imprecision in the data. Hence, the intersection operation. If the information is potentially ***erroneous***, which is always the case in practice, this over-optimistic approach will fail, no matter what algorithm is used to approximate intersection. Some researchers[8,14] have suggested to monitor the intersection procedure and discard any "outlier" that do not intersect with the "consensus" represented by the already integrated information. Even when outliers are scarce and sufficiently conflicting so as to make the discarding decision easy, this method introduces an order-dependency that is not suitable to many problems.

Another difficulty with this method is what is known as the ***registration*** (or correspondence) problem: to integrate the information acquired at time t_1 and t_2 (or location L_1 and L_2), one must explicitly tell which ellipsoid of time t_1 corresponds to which ellipsoid at time t_2. If we extract hundreds of points at each instant or location, all the ellipsoids may overlap each other in space, making this correspondence problem extremely difficult. In contrast, no such registration is necessary in our RACS-based approach.

3.2 Kalman filtering

The Kalman filtering method goes one step further into generalization. Instead of dealing with ellipsoids, we can assume that the errors are random (point) variables of the space $\mathcal{U}$ with Gaussian distribution of given covariance matrices, and additive to the signal. By doing so, the errors are not assumed to lie with certainty within a fixed bounded set anymore. Now if we restrict ourselves to linear combinations of the distributions, Kalman filtering provides the "optimal" fused distribution in the form of the ***minimal variance*** estimate of the fused data (in the context of Bayesian inference). This amounts to combining the uncertain data into a minimal "volume" Gaussian probability distribution. The method has been extended to non-linear filtering[1].

Although not required from a purely theoretical point of view, the Gaussian assumption is crucial in practice as it leads to covariance matrix computations, which are much more tractable than 3-dimensional probability distributions. However, the errors in stereo-vision are not Gaussian and can hardly be approximated by Gaussian distributions, as is clear for zero-disparity points: such points lie in unbounded polyhedra that extend to infinity and trying to fit a Gaussian, which amount to finding a center and a variance, seems simply hopeless...

Furthermore, Kalman filtering approaches do not solve the registration problem mentioned above, as they assume that we are given all time correspondences for the features.

3.3 Certainty grids

Certainty grids[6,7] have become very popular in robotics recently and there is a clear connection with our work. The starting point is the same, namely the desire to avoid the oversimplification of uncertainty typical of the above methods. The basic assumption is that, once all errors are properly modeled, it should be easy to reduce them. In that respect, the RACS approach and the certainty grid method coincide.

However, the latter is fundamentally heuristic: the authors[6] do not recognize the generality of the concept of structuring elements and stick to grid representations. They cannot handle cases where the structuring elements overlap in space. More importantly, they make the mistake of considering each box in the grid as independent from all other boxes, so that the occupancy certainty of all the boxes are statistically independent.

Finally, since they use a standard Bayesian updating scheme, they cannot represented *ignorance* fully: they must assume prior distributions or use maximum entropy considerations. For instance, taking an occupancy certainty of 0.5 for all boxes in the grid as a prior distribution, as suggested[6] when no information *a priori* is known, leads to the following absurd result: if one groups the boxes 2 by 2, hence representing the data at a coarser resolution, one gets a grid which is almost surely occupied everywhere although we know nothing a priori!

In contrast, we do not need prior information in our RACS approach and ignorance can be fully represented by means of the hitting capacity.

4. AN IMPLEMENTATION

4.1 Overview of the system

The RACS approach described above was implemented at the Visual Perception Department of ATR on the following hardware:

- an SIMD massively parallel super-computer, the Connection Machine CM-2 (Thinking Machines Corp.), with 2 sequencers of 8K processors each, which gives a total number of 16384 processing units. To each processor is attached a memory of 256K bits. A front-end workstation provides the programming/debugging environment (C language augmented with parallel instructions) and the user interface.
- a Unix workstation SPARC-2 (Sun Microsystems Corp.), on which runs the software package AVS (Application Visualization System) that provides a "user friendly" interface and 3D rendering capabilities.

- a computer-controlled rotating table CPC-3DN (Chuo Seiki Corp.)
- 2 CCD cameras (Panasonic).
- an image grabbing board and software Videopix (Sun Microsystems Corp.).

The system is composed of following programs:

- a program *stereo.c* implementing a stereo-vision algorithm. This program extracts a number of ***feature points*** in both the left and right images using either a Sobel or a Moravec filter, then matches those points using a standard window-based normalized cross-correlation technique. The disparity (vector) field originating from the left image is merged with that computed from the right image and the result is checked for consistency (the information coming from both images should be compatible wherever they overlap). Finally each disparity vector is used to produce a 3D ***data set*** represented as a 14 face polyhedron. For each stereo-pair i of images, a file $data_i$ is produced that contains the visual field at location i and all the data sets (polyhedra) computed from all the vectors in the disparity field of this pair. The output of program *stereo.c* is as many files as there are stereo image pairs. See appendix A for a detailed description of how the data sets are computed and represented in computer memory. This program runs entirely on the Connection Machine and its front end.
- a program *tree.c* implementing an octree algorithm. The inputs are the files produced by *stereo.c*, i.e. polyhedra in space. The output is a file containing the octree, i.e. a list of cubes of various locations and sizes along with 2 (integer) counters: a ***visibility counter***, i.e. the number of times the cube was visible from the 2 cameras (lied inside the visual field), and a ***hit counter***, i.e. the number of times the cube was hit (intersected) by the data set polyhedra. Note that ***hit counter*** $\leq$ ***visibility counter***. If the depth of the octree is set to 7, we get a maximum number of 2 Mega virtual processors (nodes) at the deepest layer, if the tree is full. In practice, only about 10%-30% of this number is necessary to represent typical scenes in the octree. This is because indoor or outdoor scenes usually contain a lot of empty space. This program runs entirely on the Connection Machine and its front end.
- a program *octree_to_geom.c* that converts the octree into an AVS "geometry" data type. The resulting (3D) geometry can then be given as an input to the AVS geometry renderer that performs HSR (hidden surface removal) and complete rendering, and which enables the user to interactively manipulate the octree in space with the mouse/keyboard. The resulting rendered (2D) image can then be processed using any of the image processing modules available in the AVS standard library. For example, a "pixmap" can be produced which can be dithered and written into Postscript format for printing.
- finally, an ***AVS network*** that connects all the above programs. This network is shown in fig.4. Refer to appendix B for a short introduction to programming in the AVS environment.

4.2 Experimental setting

Sequences of stereo image pairs were obtained by regularly rotating 3D objects in front of the two (fixed) CCD cameras and by grabbing the left and right images at various angles (fig.5).

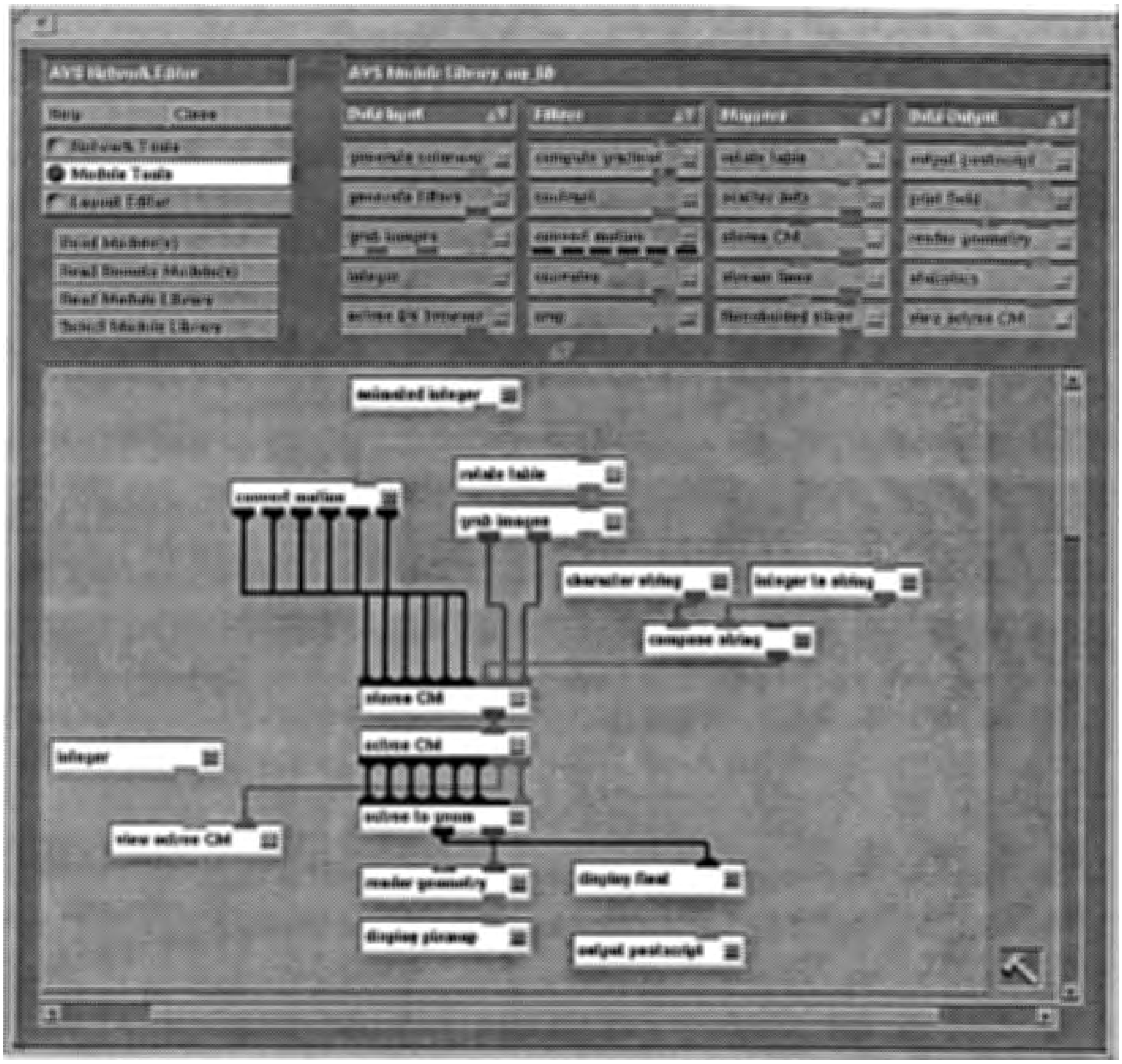


fig.4. AVS network

For the DIAMOND sequence presented below, we rotated a simple block 360 degrees and grabbed image pairs every 12 degrees, which produced a total number of 30 pairs, *diamond.0* to *diamond.29*.

The *stereo.c* program was designed for a *fixed scene/mobile robot* setup since our system will eventually be used for mobile robot navigation purposes. The inputs are the parameters of the motion of the 2 cameras, i.e. translation and orientation angles (yaw, pitch), as well as the camera parameters (focal length, baseline), and the *precision* in all those measurements. In our experimental *mobile scene/fixed robot* setup, inputs are the distance between the rotation axis and the cameras, the angle of rotation on the rotating table and the camera parameters as well as their precision. The *convert motion* AVS module handles the conversion between those two (equivalent) types of motion.

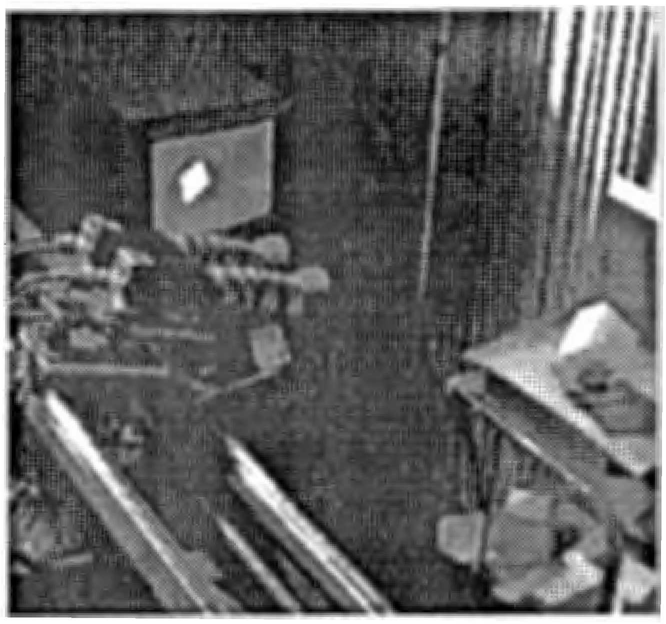


fig.5. Experimental setting for sequence DIAMOND

Since our purpose is to show that we can increase the quality of the 3D information by integrating the various data in a suitable way, we did not make great efforts to calibrate our cameras and measure all the geometrical parameters involved in our system. Only the *relative improvement* of the 3D map is meaningful here, not its absolute quality.

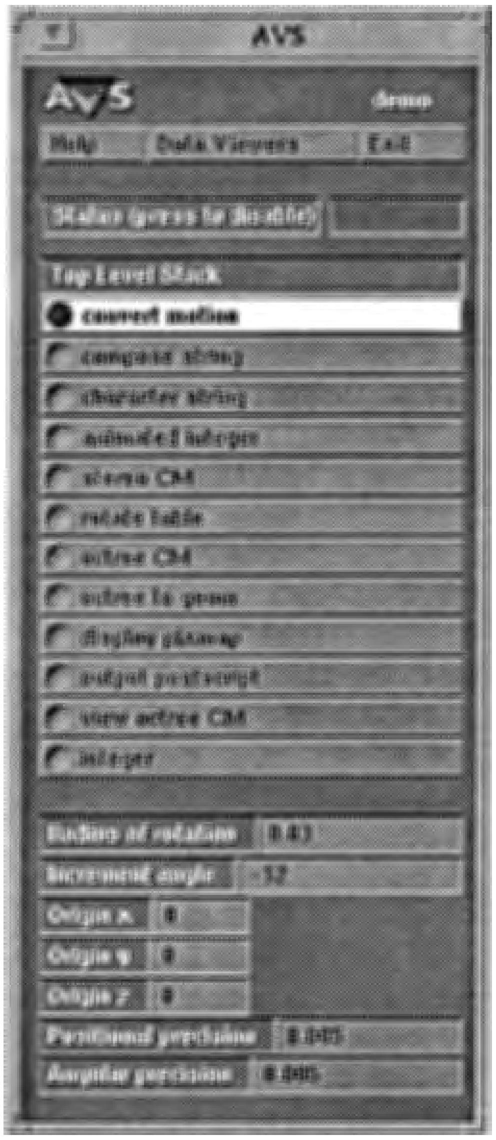
Figure 6 shows the parameter values for modules *convert motion* and *stereo CM*. The focal length was known to within 7%, the yaw with a precision of 0.005 degrees, the baseline 0.5mm and the 3D location of the system with a precision of 5mm.

Figures 7 and 8 show respectively the 1st (*diamond.0*) and 6th (*diamond.5*) image pairs of sequence DIAMOND.

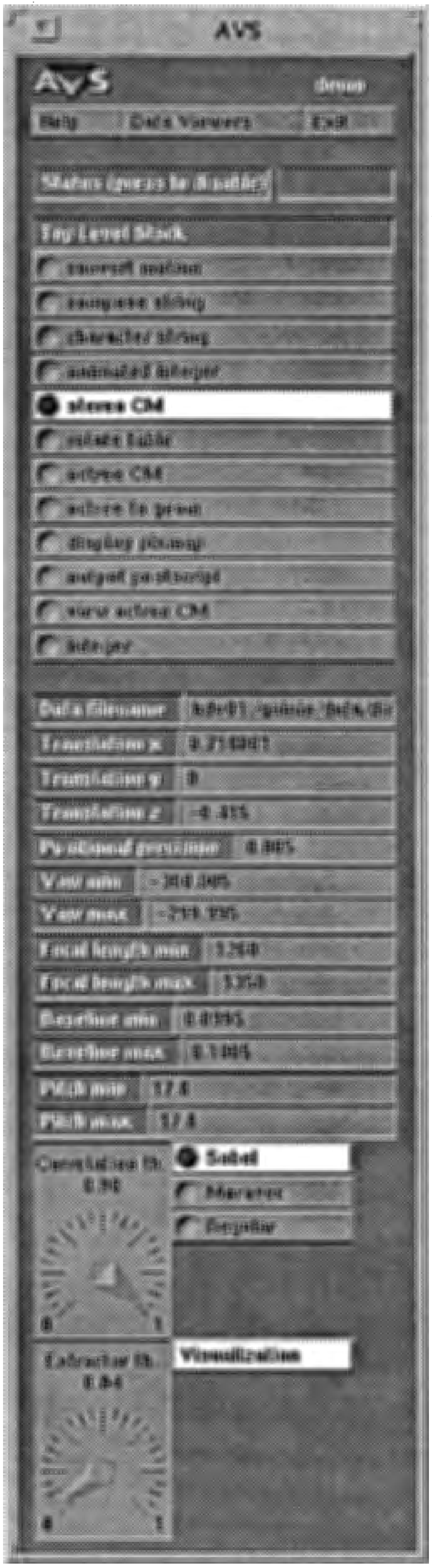
Figures 9 to 12 show various views of the distribution of the hitting Choquet capacities estimated in the octree. Each cube C of the octree has a "color" that represents the probability that the Data RACS X hits C given that C is visible from the 2 cameras. To give a better idea of this 3D distribution, we either rotate it in 3D or threshold it at various levels between 0.0 to 1.0. Note that computer graphics rendering (shading and hidden surface removal) was performed so that we can better see the individual cubes that form the octree.

Figure 9 shows various views of the hitting capacity obtained by using *one image pair only* (1st image pair *diamond.0*). The only "color" here is 1.0 since the cubes are either hit once or missed. As expected, there is no information at the "back" of the block since we cannot see it from location 0. Also, notice the amount of imprecision and noise in the 3D map.

Figure 10 shows various views of the hitting capacity obtained by using 15 image pairs (image pair *diamond.0* to *diamond.14*), i.e. 180 degrees around the block. Observe that we improved the map compared to figure 9, since both



Radius of rotation 0.43
Increment angle -12
Origin x 0
Origin y 0
Origin z 0
Positional precision 0.005
Angular precision 0.005



Data Filename [illegible]
Translation x 0.210001
Translation y 0
Translation z -0.435
Positional precision 0.005
Yaw min -300.005
Yaw max -299.995
Focal length min 1.260
Focal length max 1.350
Baseline min 0.0995
Baseline max 0.1005
Pitch min 17.8
Pitch max 17.8
Correlation th. 0.90
● Sobel
Marr
Regular
Extraction th. 0.84
Visualization

Fig. 6. Parameter values for modules *convert motion* and *stereo CM*

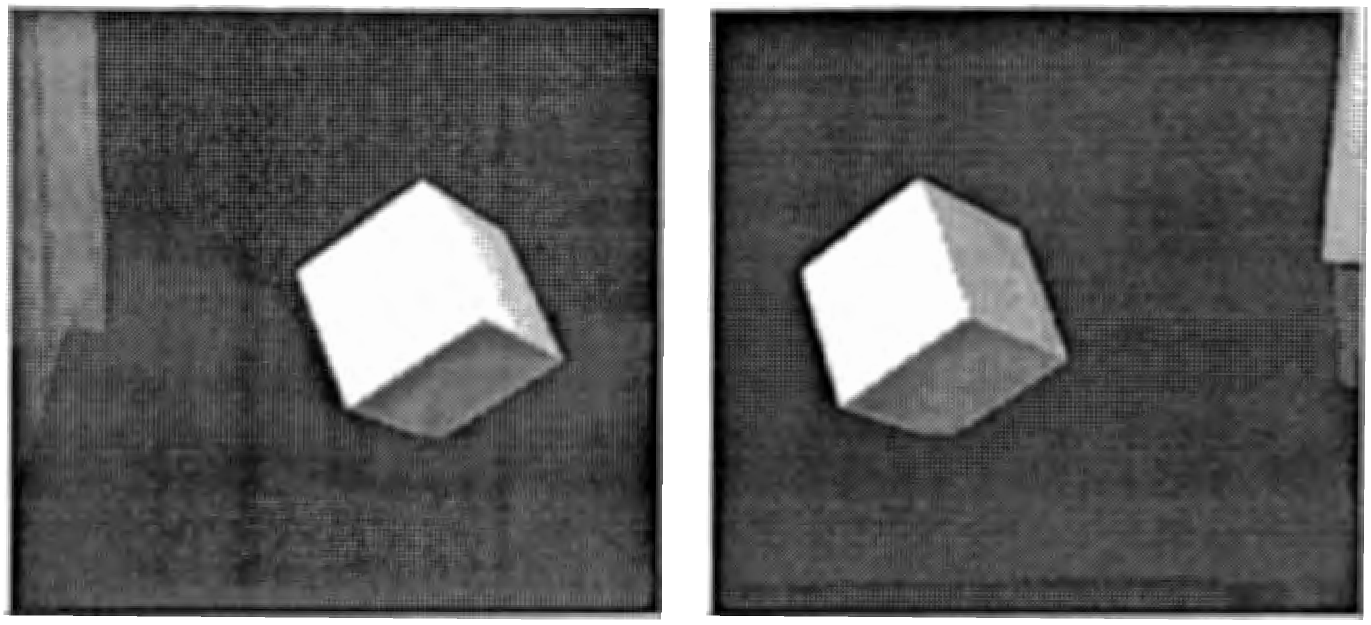


fig.7. Sequence DIAMOND: pair 0, left and right images

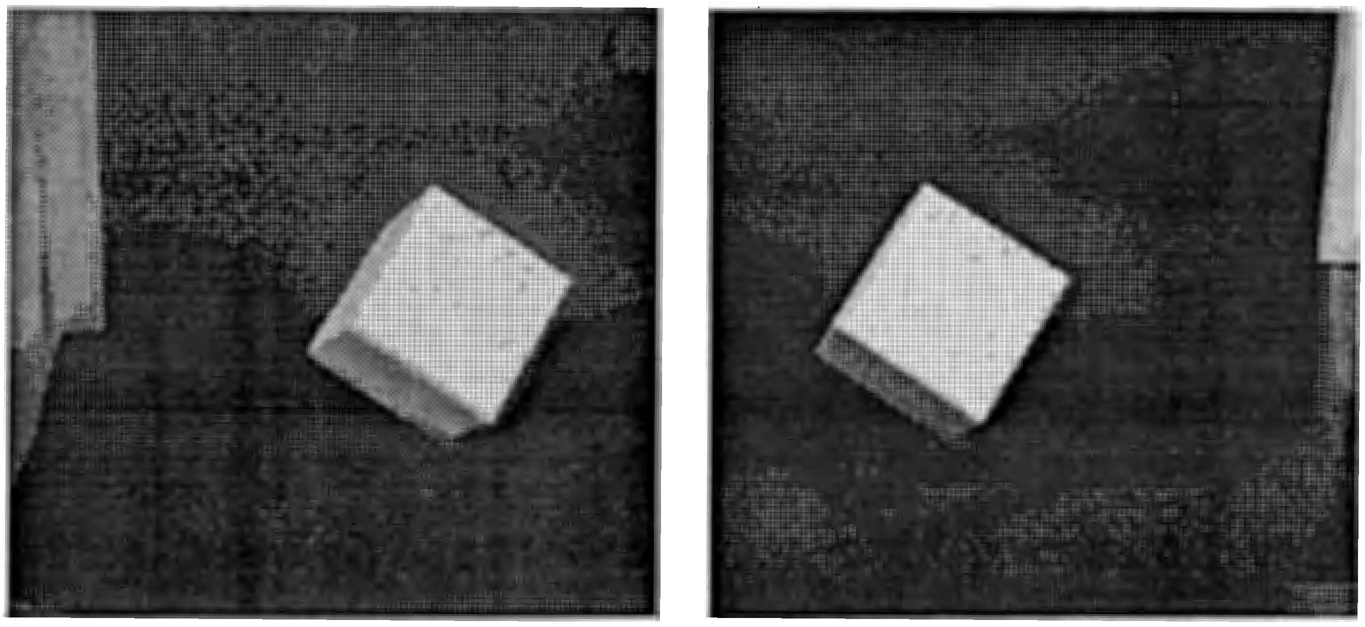


fig.8. Sequence DIAMOND: pair 5, left and right images

completeness and precision increased. As can be expected, only the half of the block is present since the rotation is not complete yet.

Figures 11 and 12 show various views of the hitting capacity obtained by using 30 image pairs (image pair *diamond.0* to *diamond.14*), i.e. 360 degrees around the block scene. Observe that we improved the map compared to figure 10, since both completeness and precision increased.

By comparing figures 9, 10 and 11-12, it is clear that integration succeeded in decreasing imprecision (the

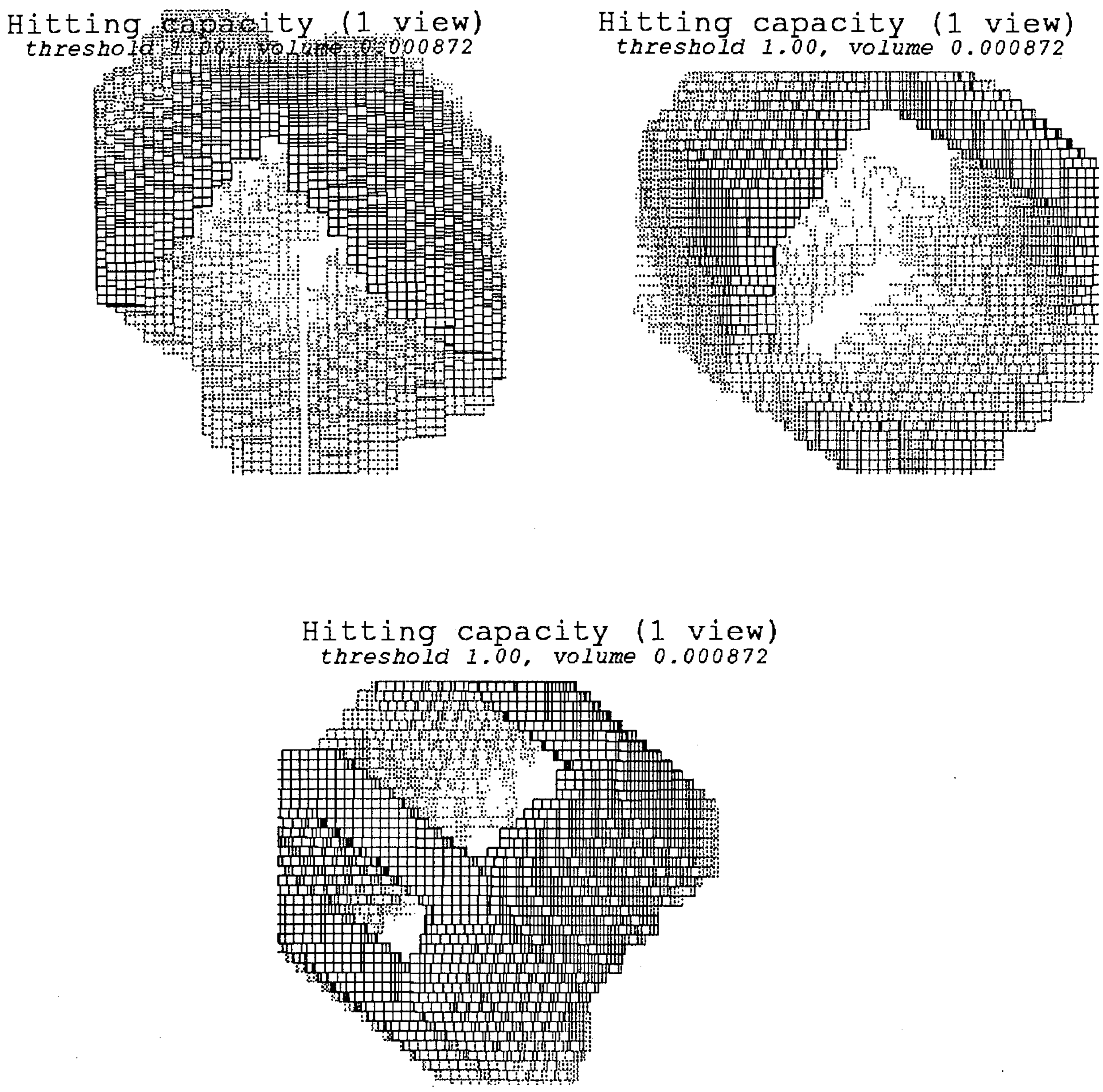

Hitting capacity (1 view)
threshold 1.00, volume 0.000872
Hitting capacity (1 view)
threshold 1.00, volume 0.000872
Hitting capacity (1 view)
threshold 1.00, volume 0.000872

fig.9. Hitting capacity using only the image pair of fig.7

apparent volume of the distribution), removing errors (noise in 3D map) and increasing the completeness of the 3D map.

This is more apparent on figure 13, which shows the volume of the distribution thresholded at τ plotted against the threshold τ. We selected 6 subsets of image pairs from the original 30 pairs *diamond.0*, ..., *diamond.29*, in a way that guarantees constant redundancy of the information: every feature in the scene is seen with equal frequency in all the 6 experiments. Hence, the curve marked *"2 views"* uses pair 0 and pair 14; the curve marked *"3 views"* uses pairs 0, 9, 19; the curve marked 5 views uses pairs 0, 6, 12, 18 and 24 etc. The curve marked *"30 views"* uses all image pairs. The curves are constructed by thresholding each distribution at various τ, measuring the volume of the

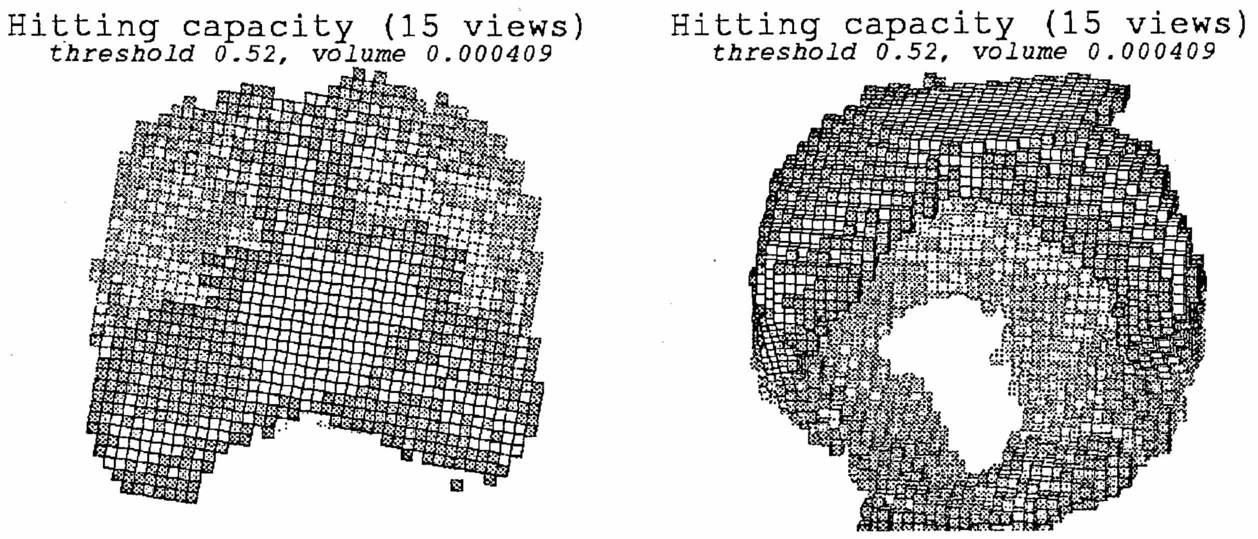

Hitting capacity (15 views)
threshold 0.52, volume 0.000409
Hitting capacity (15 views)
threshold 0.52, volume 0.000409

fig.10-1. Hitting capacity using 15 image pairs (180 deg)

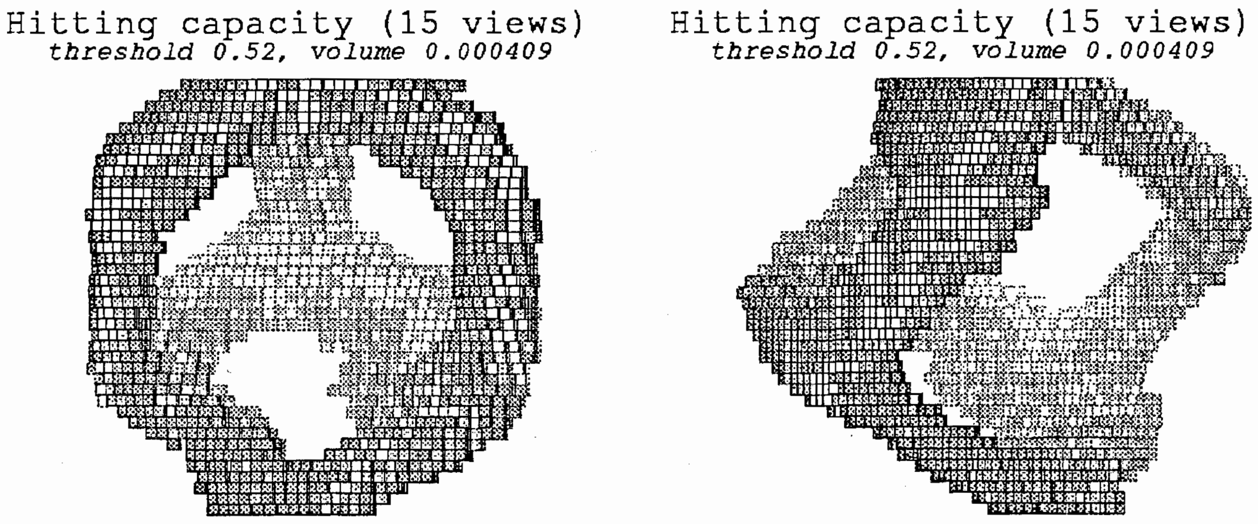

Hitting capacity (15 views)
threshold 0.52, volume 0.000409
Hitting capacity (15 views)
threshold 0.52, volume 0.000409

fig.10-2. Hitting capacity using 15 image pairs (180 deg)

threholded octree and plotting this volume against τ for all 6 experiments.

Figure 13 clearly shows that the more image pairs we use, the better the precision becomes, as can be expected intuitively. The figure also shows that the more views we integrate, the smaller the improvement that can be expected from further integrating additional views. In other words, the behavior of our system is asymptotic: the improvement beyond 15 views is negligible especially when compared to the additional amount of computation

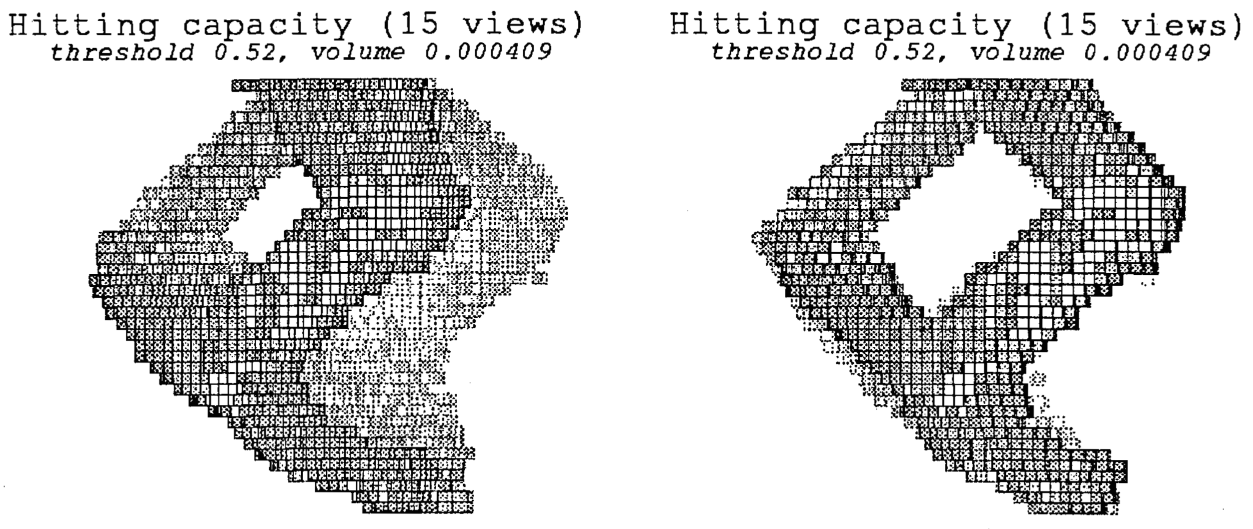

Hitting capacity (15 views)
threshold 0.52, volume 0.000409
Hitting capacity (15 views)
threshold 0.52, volume 0.000409

fig.10-3. Hitting capacity using 15 image pairs (180 deg)

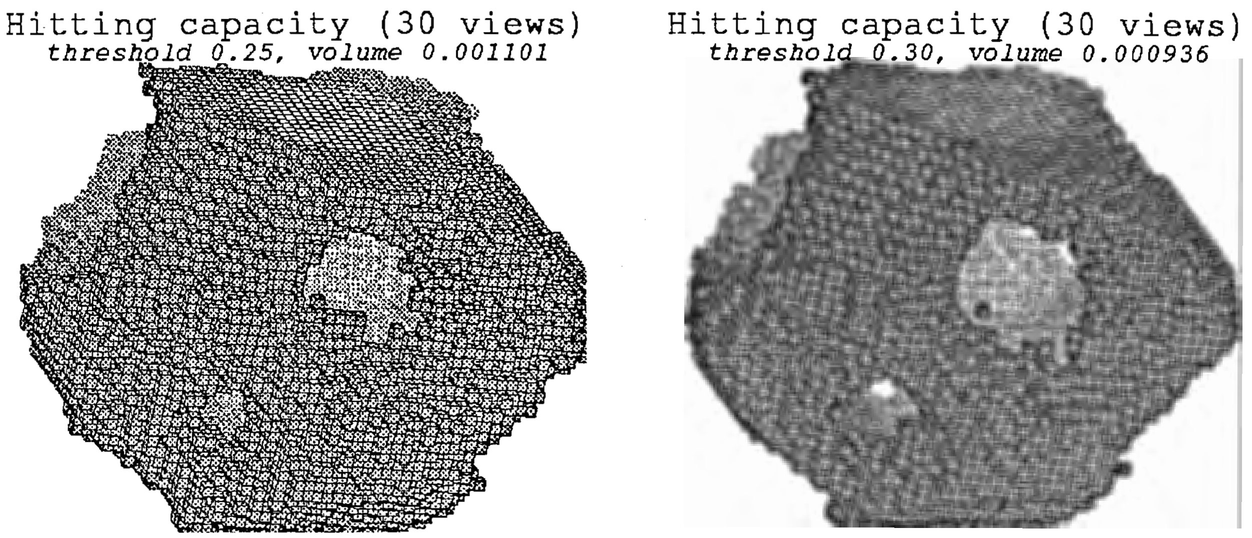

Hitting capacity (30 views)
threshold 0.25, volume 0.001101
Hitting capacity (30 views)
threshold 0.30, volume 0.000936

fig.11-1. Hitting capacity using 30 image pairs (360 deg) at various thresholds

required.

Since our scene is fairly simple, the asymptote is reached early: only a small number of views is sufficient to completely describe the scene and reduce imprecision. However, one might expect that this asymptote will be higher for a more complex scene.

5. CONCLUSION AND FUTURE WORK

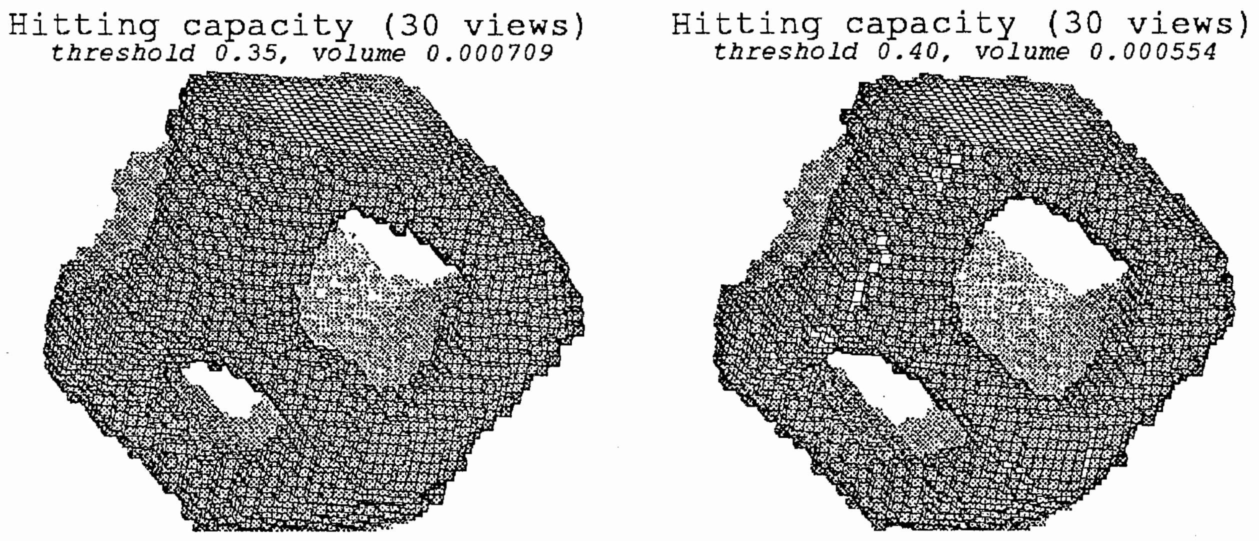

Hitting capacity (30 views)
threshold 0.35, volume 0.000709
Hitting capacity (30 views)
threshold 0.40, volume 0.000554

fig.11-2. Hitting capacity using 30 image pairs (360 deg) at various thresholds

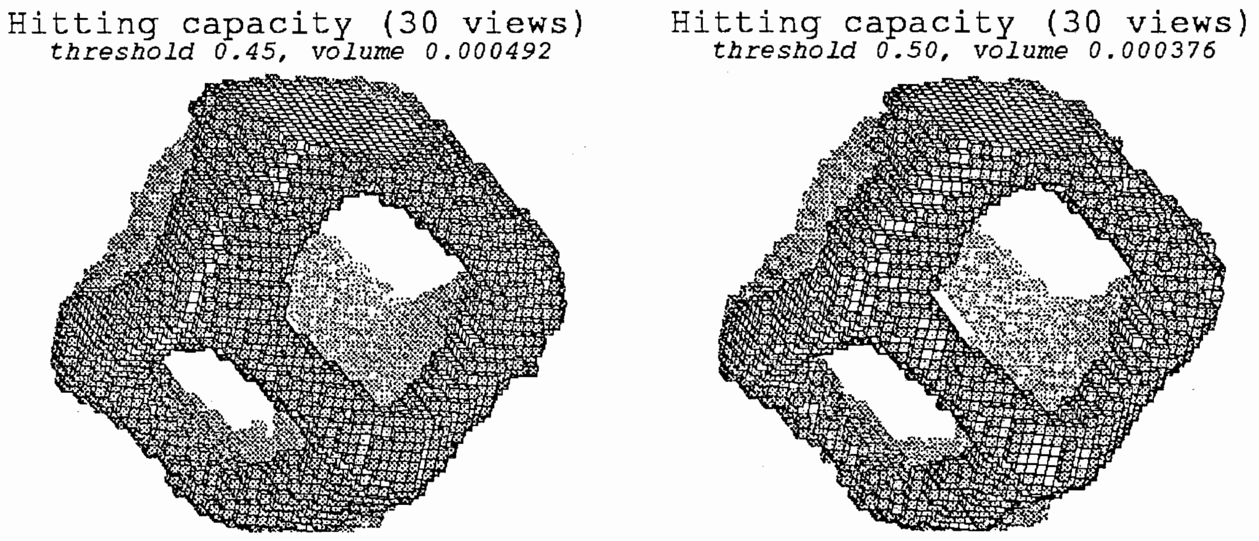

Hitting capacity (30 views)
threshold 0.45, volume 0.000492
Hitting capacity (30 views)
threshold 0.50, volume 0.000376

fig.11-3. Hitting capacity using 30 image pairs (360 deg) at various thresholds

We have proposed a new approach to the reconstruction of 3 dimensional scenes from multiple data. The information extracted from any stereoscopic image pair is inherently imprecise and uncertain and therefore a mathematical framework that can model both imprecision and uncertainty is required. Due to the limitations of the random point approach, most of the methods proposed so far cannot fully represent imprecision and uncertainty. These 2 forms of error get mixed into a (3D) probability density function and most methods are compelled to make unrealistic assumptions about the prior probability distribution of disparity vectors or even of the 3D scene features.

We have proposed to use the Random Closed Set theory for this purpose. The resulting integration method

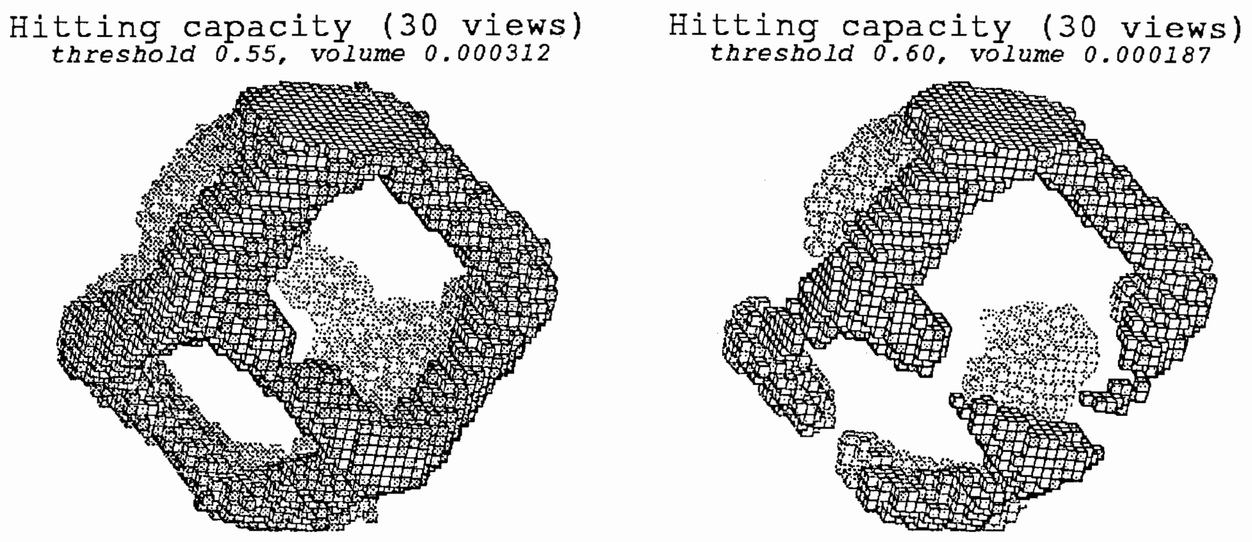

Hitting capacity (30 views)
threshold 0.55, volume 0.000312
Hitting capacity (30 views)
threshold 0.60, volume 0.000187

fig.11-4. Hitting capacity using 30 image pairs (360 deg) at various thresholds

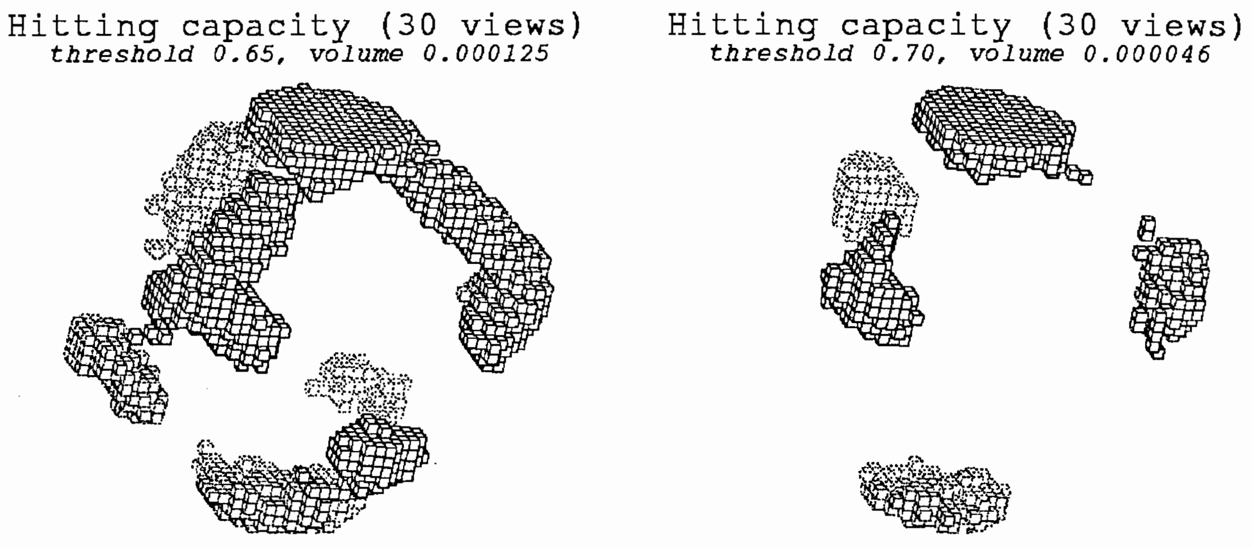

Hitting capacity (30 views)
threshold 0.65, volume 0.000125
Hitting capacity (30 views)
threshold 0.70, volume 0.000046

fig.11-5. Hitting capacity using 30 image pairs (360 deg) at various thresholds

makes no assumption whatsoever about the noise/errors and does not require prior registration of all the feature points in time. Experimental results have confirmed that one can decrease both imprecision and uncertainty at the same time while increasing completeness in the final map, provided that all sources of error are duly investigated and suitably represented.

Further theoretical and experimental work needs to be done so as to quantify more precisely the performance of the proposed method, and in particular to determine how and in what extent the complexity of a scene determines the optimal number of 2D views that must be processed in order to describe it accurately.

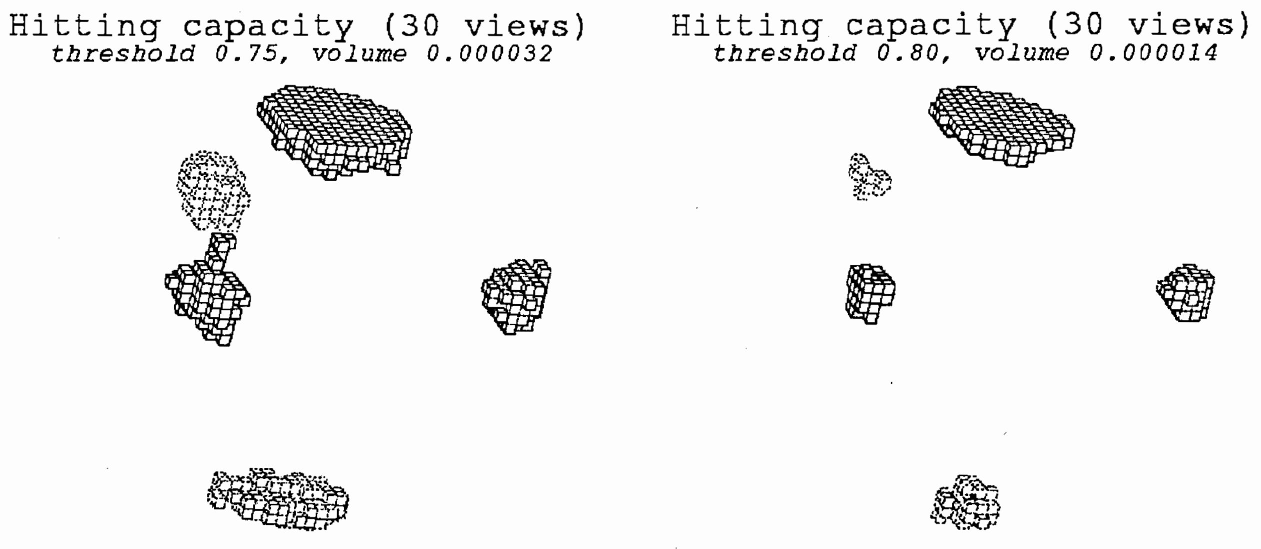

Hitting capacity (30 views)
threshold 0.75, volume 0.000032
Hitting capacity (30 views)
threshold 0.80, volume 0.000014

fig.11-6. Hitting capacity using 30 image pairs (360 deg) at various thresholds

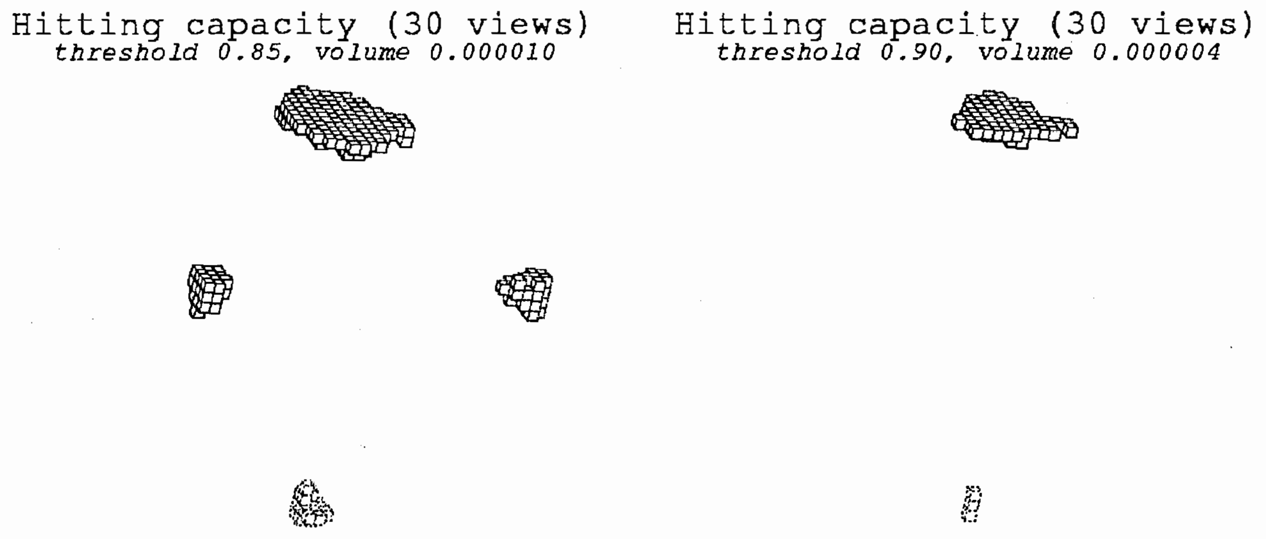

Hitting capacity (30 views)
threshold 0.85, volume 0.000010
Hitting capacity (30 views)
threshold 0.90, volume 0.000004

fig.11-7. Hitting capacity using 30 image pairs (360 deg) at various thresholds

6. ACKNOWLEDGEMENTS

The author wishes to thank T. Matsuyama, Ed Gamble and K. Arimura for their encouragement and fruitful discussions throughout this work, and E. Yodogawa and K. Ueno for their support.

7. REFERENCES

1. N. Ayache and O. Faugeras, "Building a Consistent 3D Representation of a Mobile Robot Environment by

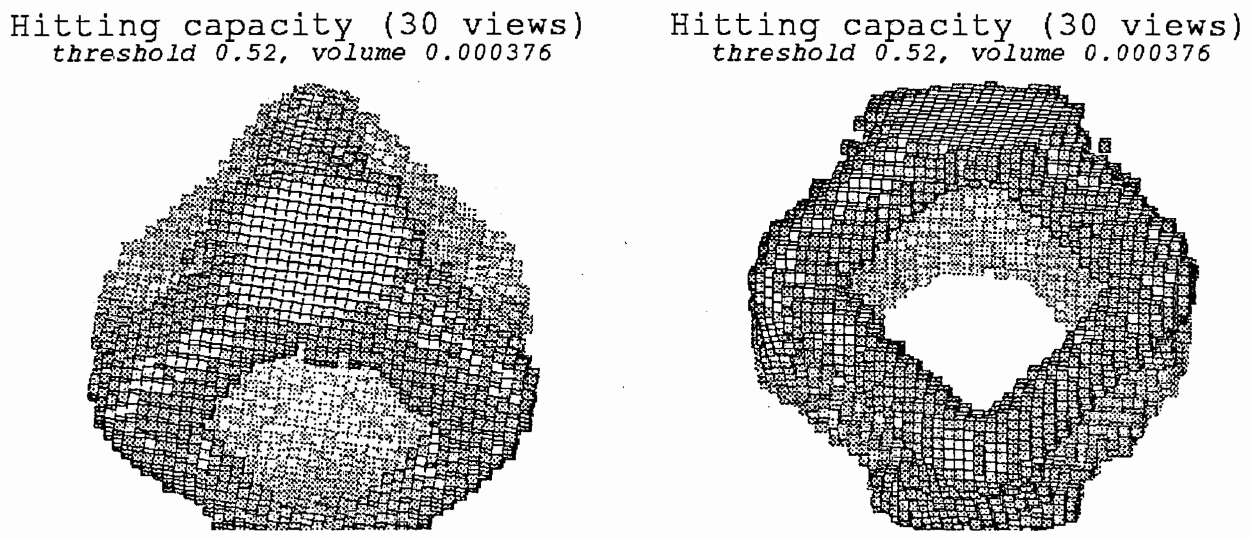

Hitting capacity (30 views)
threshold 0.52, volume 0.000376
Hitting capacity (30 views)
threshold 0.52, volume 0.000376

fig.12-1. Hitting capacity using 30 image pairs (360 deg) at various angles

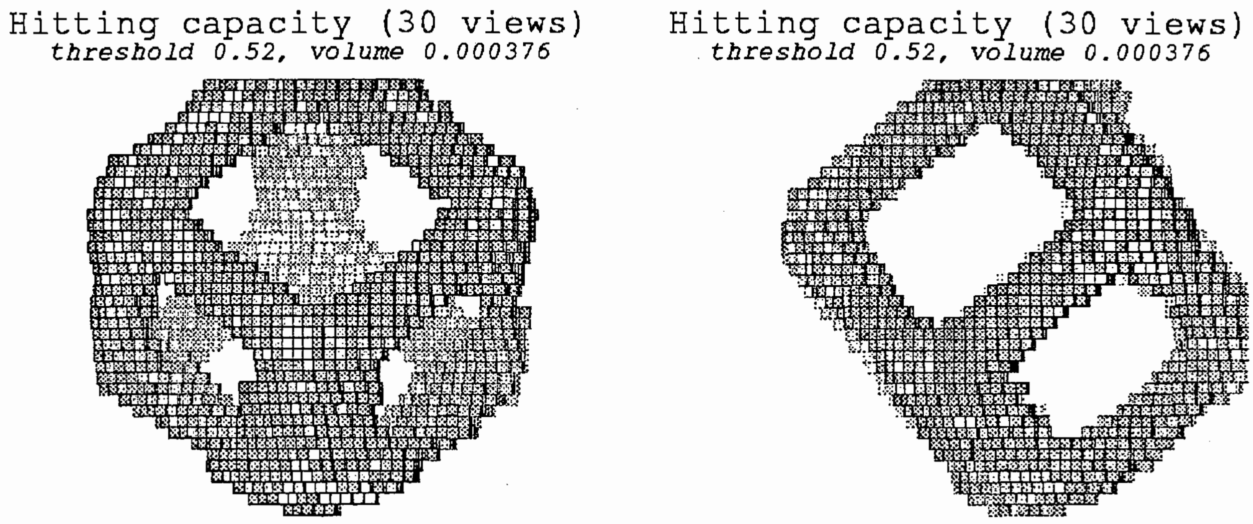

Hitting capacity (30 views)
threshold 0.52, volume 0.000376
Hitting capacity (30 views)
threshold 0.52, volume 0.000376

fig.12-2. Hitting capacity using 30 image pairs (360 deg) at various angles

Combining Multiple Stereo Views", *Proc. IJCAI'87*, pp. 808-810.

2. N. Cressie, *Statistics for Spatial Data*, Wiley, 1991.
3. C.L. Jackins and S.L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects", *Comput. Graphics Image Processing*, vol.14, 1980.
4. R.C. Luo and M.G. Kay, "Multisensor Integration and Fusion in intelligent Systems", *IEEE Trans. Systems,*

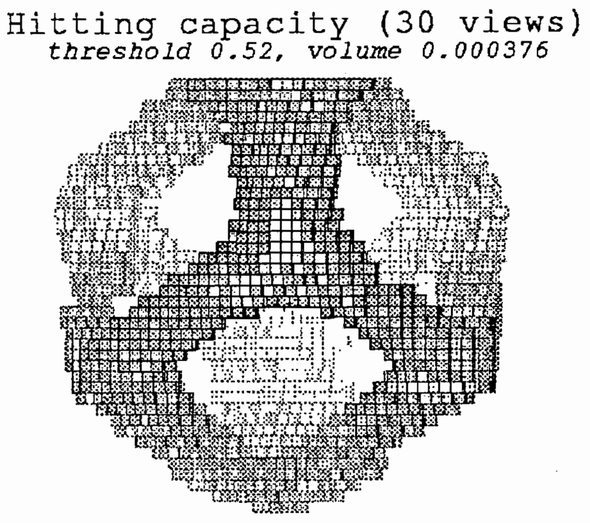


fig.12-3. Hitting capacity using 30 image pairs (360 deg) at various angles

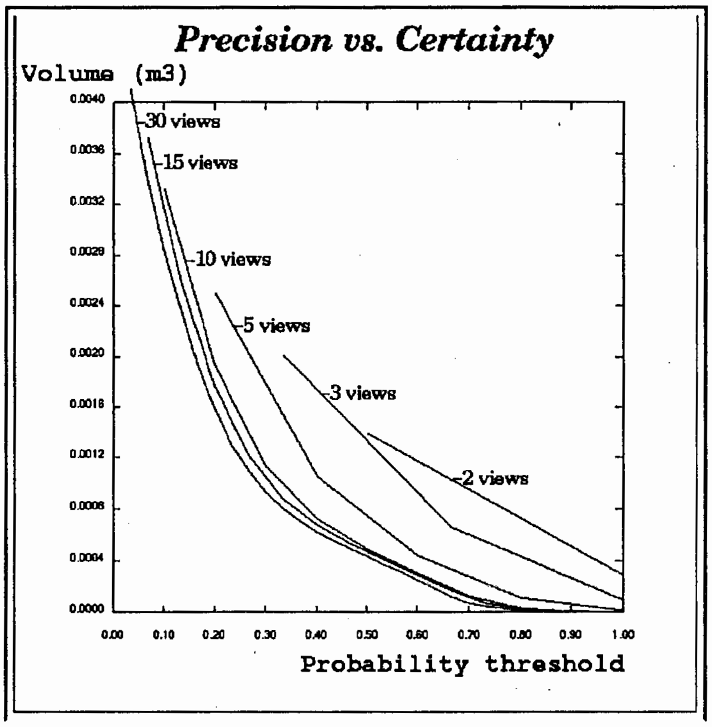


fig.13. Volume of the distribution thresholded at τ plotted against τ

(the 6 experiments were performed under the same information redundancy)

Man, Cybernetics, vol.19, no.5, 1989.

5. G. Matheron, *Random Sets and Integral Geometry*, Wiley, New York, 1975.

6. L. Matthies and A. Elfes, "Integration of Sonar and Stereo Range Data Using a Grid-based representation",

Proc. IEEE Int. Conf. Robotics and Automation, pp.727-733, Philadelphia, PA, April 1988.

7. H.P. Moravec, "Sensor Fusion in certainty grids for mobile robots", *A.I. Magazine*, vol.9, no.2, pp.61-74, 1988.
8. A. Preciado, D. Meizel et al., "Fusion of Multi-Sensor Data: a Geometric Approach", *Proc. IEEE Inter. Conf. Robotics Automation*, pp.2806-2811, Sacramento, CA, April 1991.
9. Ph. Quinio, "Mathematical Connections between the Probability, Fuzzy Set, Possibility and Dempster-Shafer Theories", *ATR technical report no. TR-A-0112*, July 1991.
10. Ph. Quinio, "Robust Stereo-Vision: a New Approach based on the Random Closed Set theory", *Proc. of the annual convention of the Institute of Television Engineers (Japan)*, Tokyo, July 1991.
11. Ph. Quinio, "Random Closed Sets: a Unified Approach to the Representation of Imprecision and Uncertainty", *Proc. of the European Conference on the Symbolic and Quantitative Approaches to Uncertainty*, in *Lecture Notes in Computer Science*, Springer-Verlag, October 1991.
12. Ph. Quinio, "A Random Set Approach to 3D Scene Reconstruction by Stereoscopic Vision", *Proc. of the 4th Symposium on Electronic Imaging: Science and Technology, Stereoscopic Displays and Applications Conference*, San Jose, USA, 12-13 February 1992.
13. J.J. Rodriguez and J.K. Aggarwal, "Stochastic Analysis of Stereo Quantization Error", *IEEE Pattern Anal. Machine Intell.* vol. PAMI12, May 1990.
14. A. Sabater and F. Thomas, "Set Membership Approach to the Propagation of Uncertain Geometric Information", *Proc. IEEE Inter. Conf. Robotics Automation*, pp.2718-2723, Sacramento, CA, April 1991.
15. J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, 1982.
16. G. Shafer, *A mathematical Theory of Evidence*, Princeton University Press, 1976.

APPENDIX A

Representation and construction of the Data and Visual sets

A.1 Representation of the Sets in computer memory

The Data Sets $X_{i,j}$ for time t_i and at image location j, and the Visual Sets V_i for time t_i are approximated by 3-dimensional convex polyhedra: 14-face polyhedra for the Data Sets $X_{i,j}$ and 8-face polyhedra for the Visual Fields V_i. Thus, they can be represented as the finite intersection of *half-spaces.* These half-spaces are defined by their boundary plane and an arbitrary rule for selecting one of the 2 half-spaces defined by the same plane.

We chose a computationally efficient representation for half-spaces: a pair consisting of a (3D) normal vector $\vec{n}$ and a 3D point P. This pair defines a *positive half-space* Σ^+ and a *negative half-space* Σ^-, by:

$$\Sigma^+ = \{M \in \mathcal{U}, \quad \vec{PM} \cdot \vec{n} \geq 0\}$$

$$\Sigma^- = \{M \in \mathcal{U}, \quad \vec{PM} \cdot \vec{n} \leq 0\}$$

Thus we must add to our representation 1 bit of information to specify the sign of the half-space defined by $\vec{n}$ and P (this is computationally cheaper that keeping only the positive half-spaces at the cost of changing the sign of $\vec{n}$).

Therefore a Data set $X_{i,j}$ is represented by a set of (14) triplets $(\vec{n}_{i,j}, P_{i,j}, \sigma_{i,j})$:

$$X_{i,j} = \bigcap \Sigma^{\sigma_{i,j}}(\vec{n}_{i,j}, P_{i,j})$$

The *global* Data Set X_i at time t_i is the union of all the polyhedra at different image location j: $X_i = \bigcup_j X_{i,j}$.

A.2 Construction of the Data Sets

A.2.1 Construction of the generic Data Sets

The generic Data Sets are the data sets obtained when the only source of imprecision is sampling (digitization). Note that $x_l \geq x_r$ since the optical axes of the 2 cameras are assumed parallel (no vergence).

Let $x_l, x_r, \delta = x_l - x_r > 0, y, f, b, \alpha, \beta, L$ be the horizontal coordinate on the left and right image and their difference (disparity), the vertical coordinate, focal length of the 2 cameras, the baseline, the yaw, the pitch angles and camera location respectively. Then a generic Data Set is defined by the following 6 faces (in Camera Coordinates):

Face 0: **furthest left** vertical plane (defined only when $\delta > 0$).

$$(\quad \vec{n}(f, 0, -x_l), \quad P(-b/2, 0, 0), \quad + \quad)$$

Face 1: **closest right** vertical plane.

$$(\quad \vec{n}(f, 0, -x_l - 1), \quad P(-b/2, 0, 0), \quad - \quad)$$

Face 2: **closest left** vertical plane.

$$(\quad \vec{n}(f, 0, -x_r), \quad P(b/2, 0, 0), \quad + \quad)$$

Face 3: **furthest right** vertical plane (defined only when $\delta > 0$).

$$(\quad \vec{n}(f, 0, -x_r - 1), \quad P(b/2, 0, 0), \quad - \quad)$$

Face 4: plane **below** set passing through L_i.

$$(\quad \vec{n}(0, f, -y), \quad P(0, 0, 0), \quad + \quad)$$

Face 5: plane **above** set passing through L_i.

$$(\quad \vec{n}(0, f, -y - 1), \quad P(0, 0, 0), \quad - \quad)$$

To obtain a World Coordinate representation, we must rotate $\vec{n}$ and P of every face by α and β and translate the result by $\vec{0L_i}$. Note that translation only applies to P (not $\vec{n}$).

A.2.2 Construction of the actual Data Sets

The actual Data Set must take into account the imprecision in the parameters f, b, α, L:

focal length :	$f_1 \leq f \leq f_2$
baseline :	$b_1 \leq b \leq b_2$
yaw :	$\alpha_1 \leq \alpha \leq \alpha_2$
camera location :	$L_i \in S(r_i, L_i)$ (sphere of radius r_i and centered at L_i)

Face 0: **furthest left** vertical plane (defined only when $\delta > 0$).

- **if** $x_l > 0$: ($\vec{n}(f_2, 0, -x_l)$, $P(-b_2/2, 0, 0)$, $+$)
- **if** $x_l \leq 0$: ($\vec{n}(f_1, 0, -x_l)$, $P(-b_2/2, 0, 0)$, $+$)
- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 1: **closest right** vertical plane.

- **if** $x_l > -1$: ($\vec{n}(f_1, 0, -x_l - 1)$, $P(-b_1/2, 0, 0)$, $-$)
- **if** $x_l \leq -1$: ($\vec{n}(f_2, 0, -x_l - 1)$, $P(-b_1/2, 0, 0)$, $-$)
- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 2: **closest left** vertical plane.

- **if** $x_r > 0$: ($\vec{n}(f_2, 0, -x_r)$, $P(b_1/2, 0, 0)$, $+$)
- **if** $x_r \leq 0$: ($\vec{n}(f_1, 0, -x_r)$, $P(b_1/2, 0, 0)$, $+$)
- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 3: **furthest right** vertical plane (defined only when $\delta > 0$).

- **if** $x_r > -1$: ($\vec{n}(f_1, 0, -x_r - 1)$, $P(b_2/2, 0, 0)$, $-$)
- **if** $x_r \leq -1$: ($\vec{n}(f_2, 0, -x_r - 1)$, $P(b_2/2, 0, 0)$, $-$)
- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 4: plane **below** set passing through L_i (approximation).

- **if** $y > 0$: ($\vec{n}$, $P(0, 0, 0)$, $+$) where $\vec{n}$ is the cross product of $\vec{n}_1(\frac{x_l+x_r}{2}, y, f_2)$ rotated by α_1, β and $\vec{n}_2(\frac{x_l+x_r}{2} + 1, y, f_2)$ rotated by α_2, β.
- **if** $y \leq 0$: ($\vec{n}$, $P(0, 0, 0)$, $+$) where $\vec{n}$ is the cross product of $\vec{n}_1(\frac{x_l+x_r}{2}, y, f_1)$ rotated by α_1, β and $\vec{n}_2(\frac{x_l+x_r}{2} + 1, y, f_1)$ rotated by α_2, β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 5: plane **above** set passing through L_i (approximation).

- **if** $y > -1$: ($\vec{n}$, $P(0, 0, 0)$, $-$) where $\vec{n}$ is the cross product of $\vec{n}_1(\frac{x_l+x_r}{2}, y + 1, f_1)$ rotated by α_1, β and $\vec{n}_2(\frac{x_l+x_r}{2} + 1, y + 1, f_2)$ rotated by α_2, β.
- **if** $y \leq -1$: ($\vec{n}$, $P(0, 0, 0)$, $-$) where $\vec{n}$ is the cross product of $\vec{n}_1(\frac{x_l+x_r}{2}, y + 1, f_2)$ rotated by α_1, β and $\vec{n}_2(\frac{x_l+x_r}{2} + 1, y + 1, f_1)$ rotated by α_2, β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 6: vertical plane perpendicular to (L_i, x) at the **left** of the set.

- **if** $x_r > 0$: ($\vec{n}(1, 0, 0)$, $P(b_1 \cdot (\frac{x_l+1}{\delta+1} - \frac{1}{2}), 0, 0)$, $+$)
- **if** $x_l \leq 0$ (defined only when $\delta > 1$): ($\vec{n}(1, 0, 0)$, $P(b_2 \cdot (\frac{x_l}{\delta-1} - \frac{1}{2}), 0, 0)$, $+$)
- **if** $x_r \leq 0$ **and** $x_l > 0$ **and** $x_l + x_r > 0$ (defined only when $\delta > 0$): ($\vec{n}(1, 0, 0)$, $P(b_1 \cdot (\frac{x_l}{\delta} - \frac{1}{2}), 0, 0)$, $+$)

- **if** $x_r \leq 0$ **and** $x_l > 0$ **and** $x_l + x_r \leq 0$ (defined only when $\delta > 0$): ($\vec{n}(1,0,0)$, $P(b_2 \cdot (\frac{x_l}{\delta} - \frac{1}{2}), 0, 0)$, $+$)
- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 7: vertical plane perpendicular to (L_i, x) at the **right** of the set.

- **if** $x_r > -1$ **and** $x_l + x_r > 0$ (defined only when $\delta > 1$): ($\vec{n}(1,0,0)$, $P(b_2 \cdot (\frac{x_l}{\delta-1} - \frac{1}{2}), 0, 0)$, $-$)
- **if** $x_r > -1$ **and** $x_l + x_r \leq 0$ (defined only when $\delta > 1$): ($\vec{n}(1,0,0)$, $P(b_1 \cdot (\frac{x_l}{\delta-1} - \frac{1}{2}), 0, 0)$, $-$)
- **if** $x_l \leq -1$: ($\vec{n}(1,0,0)$, $P(b_1 \cdot (\frac{x_l+1}{\delta+1} - \frac{1}{2}), 0, 0)$, $-$)
- **if** $x_r \leq -1$ **and** $x_l > -1$ **and** $x_l + x_r > 0$ (defined only when $\delta > 0$): ($\vec{n}(1,0,0)$, $P(b_2 \cdot (\frac{x_l+1}{\delta} - \frac{1}{2}), 0, 0)$, $-$)
- **if** $x_r \leq -1$ **and** $x_l > -1$ **and** $x_l + x_r \leq 0$ (defined only when $\delta > 0$): ($\vec{n}(1,0,0)$, $P(b_1 \cdot (\frac{x_l+1}{\delta} - \frac{1}{2}), 0, 0)$, $-$)
- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 8: horizontal plane **below** set.

- **if** $y > 0$: ($\vec{n}(0,1,0)$, $P(0, b_1 \cdot \frac{y}{\delta+1}, 0)$, $+$)
- **if** $y \leq 0$ (defined only when $\delta > 1$): ($\vec{n}(0,1,0)$, $P(0, b_2 \cdot \frac{y}{\delta-1}, 0)$, $+$)
- Rotation by β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 9 : horizontal plane **above** set (defined only when $\delta > 1$).

- **if** $y > -1$: ($\vec{n}(0,1,0)$, $P(0, b_2 \cdot \frac{y+1}{\delta-1}, 0)$, $-$)
- **if** $y \leq -1$: ($\vec{n}(0,1,0)$, $P(0, b_1 \cdot \frac{y+1}{\delta+1}, 0)$, $-$)
- Rotation by β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 10: vertical plane at **left** of object passing through L_i (only defined when $\delta > 0$).

- **if** $x_r > 0$: ($\vec{n}(f_2, 0, -\frac{x_l+x_r}{2})$, $P(0,0,0)$, $+$)
- **if** $x_r \leq 0$: ($\vec{n}(f_1, 0, -\frac{x_l+x_r}{2})$, $P(0,0,0)$, $+$)
- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 11: vertical plane at **right** of object passing through L_i (only defined when $\delta > 0$).

- **if** $x_l > -1$: ($\vec{n}(f_1, 0, -\frac{x_l+x_r}{2} - 1)$, $P(0,0,0)$, $-$)
- **if** $x_l \leq -1$: ($\vec{n}(f_2, 0, -\frac{x_l+x_r}{2} - 1)$, $P(0,0,0)$, $-$)
- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 12: **closest** vertical plane approximating $\alpha_1 - \alpha_2$ curve (defined only when $\alpha_2 \neq \alpha 1$).

($\vec{n}$, P_1, $+$) where $\vec{n}$ is the cross product of $\vec{n}_1(0,1,0)$ with $\vec{P_2P_1}$ and P_1 is $P'(b_1 \cdot (\frac{x_l+1}{\delta+1} - \frac{1}{2}), b_1 \cdot \frac{y}{\delta+1}, b_1 \cdot \frac{f_1}{\delta+1})$ rotated by α_1 and β, and P_2 is P' rotated by α_2 and β.

- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

Face 13: **furthest** vertical plane approximating $\alpha_1 - \alpha_2$ curve (defined only when $\alpha_2 \neq \alpha 1$ and $\delta > 1$).

($\vec{n}$, P_1, $-$) where $\vec{n}$ is the cross product of $\vec{n}_1(0,1,0)$ with $\vec{P_2P_1}$ and P_1 is $P'(b_2 \cdot (\frac{x_l}{\delta-1} - \frac{1}{2}), b_2 \cdot \frac{y}{\delta-1}, b_2 \cdot \frac{f_2}{\delta-1})$ rotated by α_1 and β, and P_2 is P' rotated by α_2 and β.

- Translation by $\vec{OL_i}$. Dilation by sphere $S(r_i, O)$.

A.3 Construction of the Visual Fields

A.3.1 Construction of the generic Visual Fields

The generic Visual Fields are the visual fields obtained when the only source of imprecision is sampling (digitization). They are determined by the location L_i and orientation α, β of the cameras and are *data-independent* (they do not depend on x_l, x_r, y and δ of the image features).

Face 0: (closest) **right** vertical plane.

($\vec{n}(f, 0, -(Size/2 - 1 + 1))$, $P(-b/2, 0, 0)$, $-$)

Face 1: (closest) **left** vertical plane.

($\vec{n}(f, 0, -(-Size/2 + 1))$, $P(b/2, 0, 0)$, $+$)

Face 2: plane **below** set passing through L_i.

($\vec{n}(0, f, -(-Size/2 + 1))$, $P(0,0,0)$, $+$)

Face 3: plane **above** set passing through L_i.

($\vec{n}(0, f, -(Size/2 - 1 + 1))$, $P(0,0,0)$, $-$)

Face 4: closest vertical plane perpendicular to (L_i, z).

($\vec{n}(0,0,1)$, $P(0, 0, \frac{f \cdot b}{\delta_{MAX}+1}))$, $+$)

To obtain a World Coordinate representation, we must rotate $\vec{n}$ and P of every face by α and β and translate the result by $\vec{0L_i}$. Typical values for *size* and δ_{MAX} are 512, and 64 or 128 respectively.

A.3.2 Construction of the actual Visual Fields

The actual Visual Field must take into account the imprecision in the parameters f, b, α, L.

Face 0: (closest) **right** vertical plane.

($\vec{n}(f_2, 0, -(Size/2 - 1 + 1))$, $P(-b_2/2, 0, 0)$, $-$)

- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 1: (closest) **left** vertical plane.

($\vec{n}(f_2, 0, -(-Size/2 + 1))$, $P(b_2/2, 0, 0)$, $+$)

- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 2: plane **below** set passing through L_i (yaw α_2).

($\vec{n}(0, f_2, -(-Size/2 + 1))$, $P(0,0,0)$, $+$)

- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 3: plane **below** set passing through L_i (yaw α_1).

($\vec{n}(0, f_2, -(-Size/2 + 1))$, $P(0,0,0)$, $+$)

- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 4: plane **above** set passing through L_i (yaw α_2).

($\vec{n}(0, f_2, -(Size/2 - 1 + 1))$, $P(0,0,0)$, $-$)

- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 5: plane **above** set passing through L_i (yaw α_1).

($\vec{n}(0, f_2, -(Size/2 - 1 + 1))$, $P(0,0,0)$, $-$)

- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 6: closest vertical plane perpendicular to (L_i, z) (yaw α_2).

($\vec{n}(0,0,1)$, $P(0,0,\frac{f_2 \cdot b_2}{\delta_{MAX}+1}))$, $+$)

- Rotation by α_2 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Face 7: closest vertical plane perpendicular to (L_i, z) (yaw α_1).

($\vec{n}(0,0,1)$, $P(0,0,\frac{f_2 \cdot b_2}{\delta_{MAX}+1}))$, $+$)

- Rotation by α_1 and β.
- Translation by $\vec{OL_i}$. Erosion by sphere $S(r_i, O)$.

Although very simple and easy to read and understand, the above representation of polydra by half-spaces defined by ($\vec{n}$, $P(x,y,z)$, $+/-$) is not optimal in terms of memory space: there are 7 degrees of freedom whereas the optimal number is 4. Thus we make the representation more compact by imposing $\vec{OP} = \vec{n}$, i.e. by making $\vec{OP}$ perpendicular to the plane and making the modulus of $\vec{n}$ be the distance from the plane to the origin. This representation is not complete however and there are cases when it cannot be used: whenever the origin O belongs to the plane, i.e. when the origin O lies at the boundary of the polyhedron. In such cases, a warning is issued by the program and the user is asked to move the origin to another (arbitrary) location.

APPENDIX B
Programming in the AVS environment

The basic computing unit in AVS is the *module*. Modules are dragged from their library to the network workspace using the mouse and their inputs/outputs are connected by pressing the appropriate button on the mouse. Fairly complicated networks can be built in this manner, including incrementations, loops, animations and so on.

In addition to the standard library of modules provided with the AVS package, the user can create new modules written in C (or Fortran) and add them into his own library. An attractive feature of AVS is that modules can be executed on *remote machines*. For example, one can easily create modules that are run on the Connection Machine (and its front-end computer), modules that use special hardware installed on remote machines (e.g. CD cameras, frame grabber, computer-controlled tables etc), and *integrate* all inputs/outputs into a single AVS network. Refer to the *AVS User's guide* and the *AVS Developer's guide* for more information on how to use AVS and how to write new modules.

```
/******************************************************************************
 *                                                                            *
 *                          An example AVS module                             *
 *                        integer to string conversion                        *
 *                                                                            *
 ******************************************************************************/

/* include files */
#include <stdio.h>
#include <avs/avs.h>
#include <avs/field.h>
#include <avs/flow.h>
#include <my_functions_and_macros.h>

/* Used for AVSmessage() */
static char file_version[] = "@(#)integer_to_string.c   Quinio 92/02/24";

/******************************************************************************/
AVSinit_modules()
{
    int integer_to_string();

    AVSmodule_from_desc( integer_to_string );
}
/******************************************************************************/
static
integer_to_string()
{
  int param;
  int integer_to_string_init();
  int integer_to_string_compute();
  int integer_to_string_destroy();

  /* Set the module name */
  AVSset_module_name("integer to string", /* type of module */ MODULE_MAPPER );

  /* Create the input port */
  AVScreate_input_port( /* port name */ "integer", /* AVS type */ "integer",
                        /* specify is required or optional */ REQUIRED );

  /* Add parameters as needed, and connect them to widgets */
  /*
    param = AVSadd_parameter( "parameter name", "string", string, "", "" );
    AVSconnect_widget( param, "typein" );
  */

  /* Create the output port */
  AVScreate_output_port( /* output port name */ "string", /* AVS type */ "string");

  /* Set the (optional) init procedure */
  AVSset_init_proc( integer_to_string_init );

  /* Set the compute procedure */
  AVSset_compute_proc( integer_to_string_compute );

  /* Set the (optional) destroy procedure */
  AVSset_destroy_proc( integer_to_string_destroy );

}
/******************************************************************************/
static
integer_to_string_compute( integer, string )
     int integer;
     char **string;
{
  /* this is the 'compute' function of the module; it is run everytime an
     input or parameter is changed */

  char the_string[ MISC_BUFFER_SIZE ];

  sprintf( the_string, "%d", integer );
  *string = the_string;

  /* Indicate success */
  return(1);
}
/******************************************************************************/
static
integer_to_string_init()
{
  /* this is the 'init' function of the module; it is run once when the
     module is dragged onto the network workspace */

}
/******************************************************************************/
static
integer_to_string_destroy()
{
  /* this is the 'destroy' function of the module; it is run once when the
     module is "hammered" (destroyed) */

}
```

fig.10: an example AVS module written in C