Experiments with a Network-Based Geometric Reasoning Engine

Robert B. Fisher  
Mark J. L. Orr  
Dept. of Artificial Intelligence, University of Edinburgh, Scotland  
5 Forrest Hill, Edinburgh EH1 2QL, Scotland, United Kingdom

Abstract
Fisher and Orr (1988) described a geometric reasoning engine based on a value passing constraint network. This paper reports on more recent results from this research: several experiments on detecting inconsistency between constraints, how to bind degrees of freedom in a revolute joint, some results on propagation of error in the networks, and performance on a large three dimensional reasoning problem involving three revolute joints.

Keywords: geometric reasoning, three dimensional scene analysis

1 Introduction
Fisher and Orr (1988) described a geometric reasoning engine based on value passing constraint networks. The networks are created based on an underlying algebraic description of three-dimensional geometric relations and transformations. These networks implement the five key visual geometric reasoning functions: locate, predict, transform, inverse and merge (Orr and Fisher 1987). The networks also produce tighter bounds than simply using symbolic algebra and are have greatly improved efficiency (over, e.g. ACRONYM (Brooks 1981)).

Moreover, we observed that the forms of the algebraic constraints tended to be few and repeated often, and hence standard subnetwork modules could be developed, with instances allocating as new geometric constraints were identified during model matching. For example, network modules exist for constraints like: "a model vector is assigned to a data vector" and "a data boundary must lie inside a transformed model boundary".

This paper describes and discusses some recent experiments and results obtained using the networks. In particular, we describe:

- experiments with detecting inconsistency between constraints,
- how one can bind degrees of freedom in a revolute joint,
- some results on propagation of error in the networks, and
- performance of a large network with three revolute joints.

These are described individually in sections 4 through 7. We start with a brief review of the main concepts behind the network-based geometric reasoner:

1. evaluating algebraic constraints in a value-passing network,
2. specifying the geometric problem as algebraic constraints and
3. partitioning the networks into prototypical modules.

Acknowledgements
This work was funded by the University of Edinburgh and Alvey Grant GR/D/17403. I'd like to thank J. Hallam and M. Cameron-Jones for help with the text and work.

2 Numerical Constraint Networks
The computation implemented by an algebraic network is given by a set of algebraic equalities and inequalities (determined by the problem being solved). Because we are interested in estimating upper (SUP) and lower (INF) bounds on all variables, the arithmetic relationships are reformulated into interval form (Alefeld and Herzberger 1983).

We must then estimate the bounds on the variables, given the algebraic relationships in which they play a part. ACRONYM's constraint manipulation system (Brooks 1981) used a symbolic algebra technique to estimate bounds on all quantities. Because this symbolic algebra method was slow and did not always give tight bounds, a new network approach was designed. The construction and evaluation of these networks was described in Fisher and Orr (1988), but a quick overview is given now.

The basis of the network was the propagation of updated bounds, through functional units linked according to the algebraic problem specification. A simple example is based on the inequality:

\[ A \leq B - C \]

By the SUP/INF calculus (e.g. Brooks 1981), the upper bound of A is constrained by:

\[ SUP(A) \leq SUP(B) - INF(C) \]

as are the lower bound of B and upper bound of C:

\[ INF(B) \geq INF(A) + INF(C) \]

Thus, one can use the value of SUP(B) - INF(C) as an estimated upper bound for A, etc. These relationships are used to create the network for this single inequality, which is shown in Figure 1. As new bounds on B are computed, perhaps from other relationships, they propagate through the network to help compute new bounds on A and C.

There are two advantages to the network structure. First, because values propagate, local improvements in estimates propagate to help constrain other values elsewhere. Even for just local problems, continued propagation until convergence produces better results than the symbolic methods of ACRONYM. Second, these networks have a natural wide-scale parallel structure (e.g. 1000+) that might eventually lead to extremely fast evaluation. One disadvantage is that all bounding relationships must be pre-computed as the network is compiled, whereas the symbolic approach can be opportunistic when a fortuitous set of constraints is encountered.

For a given problem, the networks can be complicated, particularly since there may be both exact and heuristic bounding relationships.
3 Geometric Reasoning Networks

The key geometric reasoning data type for high-level computer vision is the position, that represents the relative spatial relationship between two visual features (e.g. world-to-camera, camera-to-model, or model-to-subcomponent). A position consists of a 3-vector representing relative translation and a unit quaternion (4-vector) representing relative orientation (of the form \( \cos(\theta/2),\sin(\theta/2) \mathbf{u} \)) for a rotation of \( \theta \) about the axis \( \mathbf{u} \).

![Figure 1: Small Constraint Network Example](image)

The key geometric relationships concern relative position and have two forms - exact and partially constrained. An example of an exact form is: let object A be at global position \( T_{A-B} = (r_{A-B}, q_{A-B}) \) (rotation \( r_{A-B} \) and translation \( t_{A-B} \)) and object B be at \( T_{A-B} = (r_{A-B}, t_{A-B}) \) relative to A. Then, the global position of B is:

\[
T_{A-B} = T_{A-B} \ast T_A = (r_{A-B} \ast r_A, t_A + t_{A-B})
\]

where \( \ast \) is the quaternion multiplication operator and \( \ast \) is the quaternion inverse operator.

A partially constrained position is given by an inequality constraint:

\[
i_{A-B} \geq 50
\]

which means that the \( i \) component of \( A \)'s global position is at least 50.

We now include a simple example of network use. Suppose subcomponents B and C are rigidly connected to form object A. Given the estimated positions of the subcomponents in the global coordinate systems \( T_{A-B} \) and \( T_{A-C} \), and the transformations between the object and subcomponent coordinate systems, \( T_{A-B} \) and \( T_{A-C} \), these can be used to estimate the global object position, \( T_{A-B} \), using two instances of the "TT" module listed above. Figure 2 shows this network. Notice that each subcomponent gives an independent estimate of \( T_{A-B} \), so that the network keeps the tightest bounds on each component of the position. Any tighter resulting bounds then propagate back through the modules to refine the subcomponent position estimates.

Rigid Subcomponent Hierarchy

![Figure 2: A Simple Geometric Reasoning Network](image)

4 Binding Degrees of Freedom

Binding degrees of freedom is a typical visual reasoning problem, such as finding the positional parameters, as above. Another example is estimating an intrinsic linear degree of freedom, as in a prismatic robot joint. Its underconstrained position can be modeled by using a variable in the reference frame translation. Then, bounds on this variable can be estimated by using the network method to compute its reference frame relationship relative to its main object.

The more difficult case is a rotational degree of freedom, as in a robot revolute joint, which we now consider in more detail. The problem with this case is that there is no element of the rotation specification that exactly corresponds to the degree of freedom (unlike the translation case). Fortunately, the \( \cos(\theta/2) \) component of a quaternion can be related to a joint's rotation. Figure 3 shows how to exploit this \( \cos(\theta/2) \) relationship in a subnetwork that solves the problem. The key subcomponents are B and D, with their estimated global positions \( T_{B-D} \) and \( T_{D-B} \). From the model, we know which vector in each of their local reference frames is the axis of rotation (azixb and azixd). Also, from the model we have reference vectors, \( r_f \) and \( r_f \), such that when they are aligned, the rotation is zero. These two
positions and four vectors are inputs into the joint angle subnetwork, at the top of figure 3.

The two TV2 modules at the top map each pair of model vectors to a corresponding pair of scene vectors. Two of these map to the common axis vector. From these we can estimate the joint rotation in quaternion rotation form. This is done by the third TV2 module in the middle of the network. It rotates the global outrefb vector to the global outrefd vector, while maintaining the direction of the global axis joints.

Finally, to extract the joint angle itself, we use a QWF module that maps a rotation quaternion to a $\langle \theta, \delta \rangle$ axis-angle form. Here, vector $\delta$ is also the same as the global rotation axis joints, so this helps the module. (The angular output is given in the form $\cos(\theta)$).

The following is an example of the joint network:

$$\text{axisd} = \text{axisd} = (0, -1, 0)$$

$$r_{efb} = r_{refd} = (1, 0, 0)$$

$$\theta_{efb} = \{(0, 0), (0, 0), (0, 0), (0, 0)\}$$

$$\theta_{efd} = \{(0, 0), (0, 0), (0, 0), (0, 0)\}$$

with very small intervals (0.00001 width) correctly gives an estimated rotation: $\cos(\theta) = [0.866, 0.864]$.}

5 Detecting Inconsistency

The bound estimation algorithm is an incomplete decision procedure. If a network is inconsistent ($\text{INF}(V) > \text{SUP}(V)$ for any variable $V$) then there is definitely no solution. However, a consistent network does not guarantee that a solution exists. This is a problem because the network’s ability to detect inconsistent sets of constraints relates to the vision program’s ability to reject incorrect hypotheses. We would therefore like to investigate two questions about sets of constraints that have no solution but that nevertheless lead to consistent networks: (1) how likely are such problems and (2) typically “how close to being solvable” are they.

As our test case we take the problem of finding the rotation that maps a pair of 3D model direction vectors into a corresponding pair of 3D image vectors. This is not only a typical problem in geometric reasoning but also a basic one in that other constraints involving multiple matched pairs of directions and locations can be solved by decomposing them into one or more problems of this sort. For our present purposes we assume that the model vectors are known precisely while the image vectors are known only within certain errors. We further assume the errors are isotropic, that is, a nominal vector $\vec{v}$ is enclosed within an error cone of internal angle $\epsilon$ to yield the interval vector $<\vec{v}, \epsilon>$.

A solution exists only if the model vectors can be rotated to lie within the error cones of their corresponding image vectors. Let $\vec{v}_1$ and $\vec{v}_2$ be the (exact) model vectors and $<\vec{v}_1', \epsilon>$ and $<\vec{v}_2', \epsilon>$ be the (inexact) image vectors with isotropic error $\epsilon$ radians. Then a solution exists if:

$$\cos^{-1}(\epsilon) - 2\epsilon < \cos^{-1}(\epsilon) - 2\epsilon < \cos^{-1}(\epsilon) + 2\epsilon$$

For different values of $\epsilon$ we estimate the probabilities that randomly chosen vectors, will result in a solvable problem and consistent network by performing many trials. The results are shown in the graph of figure 4. The squares show the probability of network convergence while the dots show the probability of generating a solvable problem. The latter can be calculated analytically and is shown by the curve in figure 4 which is in the function:

$$p(\epsilon) = \frac{1}{4}(2 - 2\cos(2\epsilon) + (\epsilon - 2\epsilon)\sin(2\epsilon))$$

The probability of the network being inconsistent given that the problem has a solution is zero for any value of $\epsilon$, but the probability of the network being consistent when no solution exists is significant as figure 4 shows.

We now define:

$$s = \frac{\|\theta - \theta'\|}{2\epsilon}$$

as a measure of “distance from solution”. The problem is solvable if $s \leq 1$. For values of $s$ larger than but close to 1 the problem is “only just” insoluble. By again performing many trials (for fixed $\epsilon = 0.1$) we compiled histograms of values for $s$ in trials that were predicted by the above to have no solution and led to a network state that was (a) inconsistent or (b) consistent. The results are shown in figure 5.

Figure 5 part (a) shows that there is a wide distribution of $s$ values between $s = 1$ and $s = 12$ for insoluble problems correctly detected by the network. On the other hand, part (b) shows that insoluble problems that the network could not detect are concentrated mostly between $s = 1$ and $s = 2$ and are consequently close to being solvable. We therefore conclude that the network will not give misleading results when problems are far from solution and that in practice the incompleteness problem has only a limited effect.

6 Propagation of Uncertainty

Because each numerical quantity is represented as an interval, and because most geometric calculations involve several arithmetic functions, it is possible for error intervals to grow as they propagate through the network. Hence, we examine some of the numerical performance of the network modules. This section discusses results for two key modules, TT and TP, and the joint angle estimation network.
show that the output error varies approximately linearly with input error and has magnitude about 5.5 times as big. The results were generated by perturbing each component of }T_{R-B}\text{ and }T_{P-D}\text{ by an error }\epsilon\text{ for a randomly chosen rotation, and then looking at the output error of the }\text{cot}(\theta)\text{ rotation estimate. The results also show that the error reaches a maximum when the input error reaches about }\epsilon = 0.18,\text{ so the input positions need to be somewhat constrained for joint angle estimation to work.}

Figure 6 also show that there is some variability in the results (besides the underlying trend) which is undesirable. One contributor to the problem is several special cases to the TV2 module that solve for the rotation using a different set of bounds from the main case, and hence have different numerical properties.

7 A Large Network Example

Figure 8 shows the full network generated for analyzing the position of a robot in a test scene. As before, the boxes represent transformations, but there are more types used here. The TP boxes stand for a instances of a TP module. The circular "I" boxes represent three identical instances of subnetworks allocated for transformations involving joint angles, which are omitted to simplify the diagram (each contains 7 network modules). The core of the subnetworks was shown in figure 3, but some additional modules are added to align reference frames. The relative positions of objects are given by the T structures, such as }T_{C-B}\text{, which represents the position of the robot in the global reference frame. These are linked by the various transformations. Links to model or data vectors or points are represented by the unconnected segments exiting from some boxes.

The top position }T_{C-P}\text{ is the position of the camera in the global coordinate system, and the subnetwork to the left and below relates features in the camera frame to corresponding ones in the global coordinate system. Below and right is the position }T_{R-B}\text{ of the robot in the global coordinate system and the position }T_{P-D}\text{ of the robot in the camera coordinate system, all linked by a }T_{T}^\text{TP}\text{ position transformation module. Next, to the bottom left is the subnetwork for the cylindrical robot body }T_{R-B}\text{. The }"J1"\text{ node connects the robot position to the rest ("link") on the right, whose position is }T_{E-L-K}\text{. Its left subcomponent is the rigid shoulder ASSEMBLY (SH) with its subcomponents, the shoulder body (SB) and the small shoulder joint (SOC). To the right, the }"J2"\text{ node connects to the "armarm" ASSEMBLY (UA), linking the upper arm (U) to the lower arm (L), again via another joint angle (J3). At the bottom are the modules that link model vectors and points to observed surface normals, cylindrical axis vectors, and central points, etc. Altogether, there are 61 network modules containing about 96,000 function nodes.

The network structure closely resembles the model subcomponent hierarchy, and only the bottom level is data-dependent. There, new nodes are added whenever new model-to-data pairings are made, producing new constraints on feature positions.

Evaluating the complete network from the raw data requires about 1,000,000 node evaluations in 800 "clock-periods" (thus implying over 1000-way parallelism). Given the simplicity of operations in a node evaluation, a future machine should be able to support easily a 1 microsecond cycle time. This suggests that an approximate answer to this complicated problem could be achieved in about one millisecond.

Because the resulting intervals are not tight, confidence that the mean interval value is the best estimate is reduced, though the bounds are correct, and the mean interval values provide useful position estimates. To tighten estimates, a post-processing phase progressively shrinks the bounds on the intervals. Each position variable was examined to see if its interval was tight. If not, it was reduced in size by 10% and the new bounds were then allowed to propagate through the network. For the robot example, this required an additional 12,000 cycles, implying a total solution time of about 13 milliseconds on our hypothetical parallel machine.

Using the geometric reasoning network, the numerical results for the whole robot in the test scene are summarized in Table 1. Here, the values are given in the global reference frame rather than in the camera reference frame.
Table 1: Measured And Estimated Spatial Parameters

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>MEASURED</th>
<th>ESTIMATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>488 (cm)</td>
<td>487 (cm)</td>
</tr>
<tr>
<td>Y</td>
<td>89 (cm)</td>
<td>87 (cm)</td>
</tr>
<tr>
<td>Z</td>
<td>554 (cm)</td>
<td>550 (cm)</td>
</tr>
<tr>
<td>Rotation</td>
<td>0.0 (rad)</td>
<td>0.038 (rad)</td>
</tr>
<tr>
<td>Slant</td>
<td>0.703 (rad)</td>
<td>0.702 (rad)</td>
</tr>
<tr>
<td>Tilt</td>
<td>3.14 (rad)</td>
<td>2.97 (rad)</td>
</tr>
<tr>
<td>Joint 1</td>
<td>2.24 (rad)</td>
<td>2.21 (rad)</td>
</tr>
<tr>
<td>Joint 2</td>
<td>2.02 (rad)</td>
<td>2.00 (rad)</td>
</tr>
<tr>
<td>Joint 3</td>
<td>4.94 (rad)</td>
<td>4.57 (rad)</td>
</tr>
</tbody>
</table>

Figure 7: Recognized Complete Robot Using Network Method

![Diagram of robot kinematics](image)

Figure 8: Robot Scene Geometric Reasoning Network
The results of the position estimation can be seen more clearly if we look at Figure 7, that shows the estimated robot position overlaying the original scene. Now, we are using the camera position of the whole robot assembly plus the estimated joint angles from above. While the positions of the individual components are reasonable, the accumulated errors in the position and joint angle estimates cause the predicted position of the gripper to drift somewhat from the true position. The iterative bounds tightening procedure described above produced this result, which was slightly better than the one-pass method.

8 Discussion

We have seen how the network methods can detect inconsistency and estimate degrees-of-freedom. From the examples given, it is obvious that this network approach can solve complex geometric reasoning problems, has an incremental modular structure and has potential for wide-scale parallelism. Unfortunately, after experience with the use of the network approach, it has become obvious that there are also problems, particularly when working with noisy data.

One inherent problem concerns the ambiguity in the quaternion rotation representation, as the rotation \((q_0, q_1, q_2, q_3)\) is equivalent to the rotation \((-q_0, -q_1, -q_2, -q_3)\). When coupled with the interval arithmetic, sometimes the bounds on terms cannot choose between alternatives, producing unnecessarily large bounds. For example, the \(q_0\) term may acquire a bound \([-a - \epsilon, -a + \epsilon]\), when the bound \([-a - \epsilon, -a + \epsilon]\) may be equally satisfactory. It is not possible to require that \(q_0 \geq 0\) always, because the composition of some rotations satisfying this condition may produce a rotation that violates this (using the standard quaternion formulation). A related problem also occurs with a tolerance \(\epsilon\) about \(q_0\) terms near one, creating intervals \([1 - \epsilon, 1]\) and \([-1, -1 + \epsilon]\).

These problems suggest that the model-based reasoning program must handle the results and occasionally split or introduce alternative cases, given an understanding of the representation problems. While the error analysis given in section 6 showed that errors usually do not grow too quickly, when we look at the robot network (Figure 8), we still find a problem. In particular, it turns out that the chain of reference frame transformations from the robot in the global reference frame \((T_{-x})\) to the data vectors involved in the lower arm position (below \(T_{-x}\)) involves two \(TV2\) and three \(TT\) modules. Assuming each has an output interval width of 1.7 times the input interval width means that the combined contribution of the lower arm evidence to the robot has an error interval width of \((1.7)^2 = 1000\) times the input error interval. As the maximum output interval can be only \([1, 1]\) for the rotation, this implies that the input uncertainty can be at most about 0.01, which is unreasonably strict.

However, this error analysis must also consider that many of the reference frame transformations are the identity transformation, which create little error (and need not even be introduced into the network). Further, though the estimates may not propagate all the way to the robot global position node, they will help constrain more closely linked structures.

It should be noted that geometric reasoning involving a multiple joint articulated object is a genuinely complicated problem, whereas most object recognition problems have a much flatter model hierarchy, involving fewer compound degrees-of-freedom, if any at all.

A final problem posed by the interval approach is that of finding the "best" estimate for a parameter, when the intervals are large, particularly with rotation parameters. While integrating enough randomly distributed errors will cause the widths of the intervals to converge to a "correct" answer, we seldom have more than a few measured values (e.g. surface normals, curvature axes). Consequently, a statistical technique (e.g. Durrant-Whyte 1987) or a final least-squared error position refinement step also seems attractive, though they might have problems handling rotational degrees-of-freedom.

Though research on the efficient use of these networks is continuing, problems overcome by the new technique included the bounding of transformed parameter estimates and partially constrained variables and effective detection of geometric inconsistencies. The network module decomposition means that networks can be constructed incrementally as scene analysis proceeds. The network also has the potential for large scale parallel evaluation. This is important because a considerable portion of the processing time in three dimensional scene analysis is spent doing geometric reasoning.

Bibliography