The Efficient Computation of Uncertainty Spaces for Sensor-based Robot Planning

R.H. Taylor and V.T. Rajan

Manufacturing Research Department
IBM T. J. Watson Research Center
Yorktown Heights, N.Y. 10598

Abstract

This paper describes methods for propogating systems of constrained variables, which may represent geometric uncertainties, sensing errors, disturbance forces, or other variations, through equations describing coordinate transformations in the task domain and for projecting the resulting large linear system onto a lower-dimensional space representing specific variations of interest for a particular problem. We have implemented a system based on these methods. We describe the mathematical representation, briefly describe two projection algorithms, and present a number of examples applying our implementation to robot task planning problems.

Introduction and Problem Statement

Many problems in robot task planning involve propagation of systems of constrained variables through equations describing coordinate transformations in the task domain {Lozano-Perez and Taylor 1988}. Examples include modelling of measurement and motion uncertainty, part tolerances, free motion analysis and stability of subassemblies in the presence of disturbing forces. Similar examples may be found in other geometric domains (such as mechanical design automation) as well.

Characteristically, the constrained variables correspond to small values and the problems are readily linearized. The constraints themselves may be expressed as statistical covariances {e.g., Smith and Cheeseman 1986, Durrant-White 1987} or as inequalities {Taylor 1976, Brooks 1982}. The principal advantages of covariance methods is the relative simplicity of the rules for combining multiple constrained variables and their consistency with standard gaussian models of uncertainties. On the other hand, these methods are less successful if the actual probability distributions of the variations are not well characterized by first and second moments. Furthermore, they cannot easily represent non-probabilistic constraints on the variations, such as the fact that two bodies cannot interpenetrate. This paper follows {Taylor 1976} in using inequality constraints to model physical uncertainties and sensory information. Inequalities provide a straightforward means of modelling non-probabilistic constraints and do not depend on knowledge of probability distributions. They are also compatible with geometric methods for constructing configuration spaces. Although planners that use such "worst case" analysis tend to be more conservative than those that rely on covariances, conservatism may be warranted if the objective is to produce robust strategies.

The total number of residual degrees-of-freedom is usually quite small relative to the total number of free variables in the system. For example, consider a subassembly consisting of a stack of five parts. Each part is subject to small manufacturing perturbations affecting how it is attached to the part below, which can be modelled by six free scalar variables, with inequalities expressing the possible range of variation. The *total* possible variation between the base of the stack and its top corresponds to six-dimensional polytope, which is the projection of a 30-dimensional polytope corresponding to the individual free variables. If only, say, positional displacements are required, the we project onto a three dimensional polyhedron.

In many cases, it is very desirable for the planning software to analyze this lower dimension polytope corresponding to the constrained region. In other cases, it is sufficient to solve a (non-linear) optimization problem over the region. However, if many such problems must be solved, it is sometimes more efficient to enumerate the constraints corresponding to the lower dimensional polytope only once and then solve the problems in a lower dimensional space.

Representation of Variations

Following {Taylor 1976}, we treat a coordinate transformation, T, as a nominal transformation ${}^0\mathbf{T}$, and a perturbation $\Delta\mathbf{T}$

$$\begin{aligned} \mathbf{T} &= {}^0\mathbf{T} \bullet \Delta\mathbf{T} \\ &= \mathit{trans}({}^0\mathbf{R}, {}^0\vec{\mathbf{p}}) \bullet \mathit{trans}(\Delta\mathbf{R}, \Delta\vec{\mathbf{p}}) \end{aligned}$$

Thus, if $\vec{\mathbf{v}}_a$ is a vector in coordinate system a, then the same vector in coordinate system b is given by

$$\begin{aligned} \vec{\mathbf{v}}_b &= \mathbf{T}_{ab} \bullet \vec{\mathbf{v}}_a \\ &= {}^0\mathbf{R}_{ab} \bullet (\Delta\mathbf{R}_{ab} \bullet \vec{\mathbf{v}}_a + \Delta\vec{\mathbf{p}}) + \vec{\mathbf{p}}_{ab} \end{aligned}$$

For a reasonably small $\Delta\mathbf{R}$, we can approximate $\Delta\mathbf{R} \bullet \vec{\mathbf{v}}$ by $\vec{\mathbf{v}} + \vec{\alpha} \times \vec{\mathbf{v}}$, where $\vec{\alpha}$ is a small vector. Similarly, we represent $\mathbf{R}\,\Delta\vec{\mathbf{p}}$ by a vector, $\vec{\epsilon}$, giving

$$\vec{\mathbf{v}}_b \cong {}^0\mathbf{R}_{ab} \bullet \vec{\mathbf{v}}_b + {}^0\vec{\mathbf{p}}_{ab} + \vec{\alpha}_{ab} \times \vec{\mathbf{v}}_a + \vec{\epsilon}_{ab}$$

By doing some algebra and dropping second order terms in $\vec{\alpha}$ and $\vec{\epsilon}$, we derive expressions for the product of two transformations

$$\begin{aligned} \mathbf{T}_{ac} &= \mathbf{T}_{ab} \bullet \mathbf{T}_{bc} \\ &= \mathit{trans}({}^0\mathbf{T}_{ac}, {}^0\vec{\mathbf{p}}_{ac}, \vec{\alpha}_{ac}, \vec{\epsilon}_{ac}) \\ {}^0\mathbf{R}_{ac} &= {}^0\mathbf{R}_{ab} \bullet {}^0\mathbf{R}_{bc} \\ {}^0\vec{\mathbf{p}}_{ac} &= {}^0\mathbf{R}_{ab} \bullet {}^0\vec{\mathbf{p}}_{bc} + {}^0\vec{\mathbf{p}}_{ab} \\ \vec{\alpha}_{ac} &\cong {}^0\mathbf{R}_{bc}^{-1} \bullet \vec{\alpha}_{ab} + \vec{\alpha}_{bc} \\ \vec{\epsilon}_{ac} &\cong {}^0\mathbf{R}_{ab} \bullet (\vec{\alpha}_{ab} \times {}^0\vec{\mathbf{p}}_{bc} + \vec{\epsilon}_{bc}) + \vec{\epsilon}_{ab} \\ &= ({}^0\vec{\mathbf{p}}_{bc} \times {}^0\mathbf{R}_{ab}) \bullet \vec{\alpha}_{ab} + {}^0\mathbf{R}_{ab} \bullet \vec{\epsilon}_{bc} + \vec{\epsilon}_{ab} \end{aligned} \tag{1}$$

where the vector-matrix cross product is defined by

$$\vec{\mathbf{v}} \times \mathbf{R} = [v_2\mathbf{R}_z - v_3\mathbf{R}_y \mid v_3\mathbf{R}_x - v_1\mathbf{R}_z \mid v_1\mathbf{R}_y - v_2\mathbf{R}_x]$$

and for the inverse of a transformation

$$\begin{aligned}
\mathbf{T}_{inv} &= \mathbf{T}^{-1} = \Delta\mathbf{T}^{-1} \bullet {}^{0}T^{-1} \\
&= trans({}^{0}\mathbf{T}_{inv}, {}^{0}\vec{\mathbf{p}}_{inv}, \vec{\alpha}_{inv}, \vec{\epsilon}_{inv}) \\
{}^{0}\mathbf{R}_{inv} &= {}^{0}\mathbf{R}^{-1} \\
{}^{0}\vec{\mathbf{p}}_{inv} &= -\,{}^{0}\mathbf{R}^{-1} \bullet \vec{\mathbf{p}} \\
\vec{\alpha}_{inv} &= -\,({}^{0}\mathbf{R} \bullet \vec{\alpha}) \\
\vec{\epsilon}_{inv} &= \vec{\alpha} \times ({}^{0}\mathbf{R}^{-1} \bullet {}^{0}\vec{\mathbf{p}}) - {}^{0}\mathbf{R}^{-1} \bullet \vec{\epsilon}
\end{aligned} \tag{2}$$

Generally, we represent $\vec{\alpha}$ and $\vec{\epsilon}$ as sums of vectors multiplied by free scalar variables corresponding to small degrees-of-freedom:

$$\vec{\alpha} = \sum_{i=1}^{n_b} \alpha_i \vec{\mathbf{a}}_i \qquad \vec{\epsilon} = \sum_{j=1}^{n} \varepsilon_j \vec{\mathbf{e}}_j \tag{3}$$

Thus, for example, we compute $\vec{\alpha}_{ac}$ in (1) as

$$\vec{\alpha}_{ac} \cong \sum_{i=1}^{n_{b_{ab}}} \alpha_{abi}\, {}^{0}\mathbf{R}_{bc}^{-1} \bullet \vec{\mathbf{a}}_{abi} + \sum_{j=1}^{n_{b_{bc}}} \alpha_{bcj} \vec{\mathbf{a}}_{bcj}$$

The specific interpretation of the α_i and ε_j varies depending on the situation. Typical interpretations include manufacturing tolerances on parts, possible small motions between loosely fitting parts, and sensor uncertainty.

Constraints

Following {Taylor 1976}, we use linear constraints on the scalar variables α_i and ε_j to express bounds and relationships between them. Often, simple intervals suffice. For instance, if the robot joint positioning error in (4) , below, is $\pm$ 0.005°, we get

$$-0.005\frac{\pi}{180} \le \Delta\theta_j \le 0.005\frac{\pi}{180}$$

For a more complex example, consider the situation in Figure 1. If $\mathbf{T}_{fp}$ is the transformation between a feed tray, f and a small part p, then the constraint that the upper right hand corner, $\mathbf{c}_1$ of p must lie within the feeder nest is expressed by the pair of constraints

$$(\mathbf{T}_{fp} \bullet \mathbf{c}_1) \bullet \mathbf{x} \le w_f/2 \qquad (\mathbf{T}_{fp} \bullet \mathbf{c}_1) \bullet \mathbf{y} \le h_f/2$$

For ${}^{0}\mathbf{T}_{fp} = trans(\mathbf{I}, \mathbf{0}, \vec{\alpha}_{fp}, \vec{\epsilon}_{fp})$ this resolves to

$$\begin{aligned}
(\vec{\alpha}_{fp} \times \mathbf{c}_1 + \vec{\epsilon}_{fp}) \bullet \mathbf{x} &\le w_f/2 - \mathbf{c}_1 \bullet \mathbf{x} \\
(\vec{\alpha}_{fp} \times \mathbf{c}_1 + \vec{\epsilon}_{fp}) \bullet \mathbf{y} &\le h_f/2 - \mathbf{c}_1 \bullet \mathbf{y}
\end{aligned}$$

Similar constraints may be applied to the other three corners of the part.

Kinematic chains

Expressions for the variation in open-loop kinematic chains are easily computed from repeated application of formulas (1) and (2) . For example, consider the planar-type manipulator shown in Figure 2. Each revolute axis of this robot is characterized by

$$\begin{aligned}
\mathbf{T}_j &= trans(\mathbf{I}, \mathbf{0}, \vec{\alpha}_{axj}, \mathbf{0}) \bullet trans({}^{0}\mathbf{R}_j, \mathbf{0}, \Delta\theta_j\vec{\mathbf{z}}, \mathbf{0}) \\
&\qquad \bullet trans(\mathbf{I}, d_j\vec{\mathbf{x}}, \mathbf{0}, \vec{\epsilon}_j) \\
&= trans({}^{0}\mathbf{R}_j, d_j\mathbf{x}, \vec{\alpha}_j, \vec{\epsilon}_j) \\
\vec{\alpha}_j &= {}^{0}\mathbf{R}_j^{-1} \vec{\alpha}_{axj} + \Delta\theta_j\vec{\mathbf{z}} \\
\vec{\epsilon}_j &= {}^{0}\mathbf{R}_j \bullet ({}^{0}\mathbf{R}_j^{-1} \vec{\alpha}_{axj} + \Delta\theta_j \times d_j\mathbf{x} + \vec{\epsilon}_j)
\end{aligned} \tag{4}$$

where ${}^{0}\mathbf{R}_j = Rot(\vec{\mathbf{z}}, \theta_j)$, $\Delta\theta_j$ is the uncertainty in the joint rotation angle θ_j, d_j is the nominal link length, $\vec{\alpha}_{axj}$ is the link "twist" uncertainty, and $\vec{\epsilon}_j$ is the link displacement uncertainty. The position of the end-effector, relative to the robot's base, is given by

$$trans(\mathbf{I}, h_0\vec{\mathbf{z}}) \bullet \mathbf{T}_1 \bullet \mathbf{T}_2 \bullet \mathbf{T}_3 \bullet trans(\mathbf{I}, -d_4\vec{\mathbf{z}}, \mathbf{0}, \Delta d_4\vec{\mathbf{z}})$$

where h_0, d_4 and Δd_4 are as shown in Figure 2. Multiplying this out gives us linear expressions for $\vec{\alpha}$ and $\vec{\epsilon}$. The resulting expressions for a typical joint configuration of the robot are shown in the figure, along with inequalities bounding all the free variables for a typical design.

Graphs

More generally, we are concerned with kinematic *graphs* which may contain cycles and redundent paths. In general, we compute T_{ab} as follows:

1. Compute each acyclic path from a to b.
2. For each such path compute the corresponding open-chain transformation

   $${}_{k}\mathbf{T}_{ab} = \prod_{j=1}^{n_k} {}_{k}\mathbf{T}_j$$

3. For each $k > 1$, we create equality constraints

   $${}_{1}\mathbf{T}_{ab}^{-1} \bullet {}_{k}\mathbf{T}_{ab} = trans(\mathbf{I}, \mathbf{0}, \mathbf{0}, \mathbf{0})$$

   If ${}^{0}_{k}\mathbf{R}_{ab}$ and ${}^{0}_{k}\vec{\mathbf{p}}_{ab}$ are identical for all k, we can use the simpler constraints:

   $$\begin{aligned}
   {}_{1}\vec{\alpha}_{ab} - {}_{k}\vec{\alpha}_{ab} &= \mathbf{0} \\
   {}_{1}\vec{\epsilon}_{ab} - {}_{k}\vec{\alpha} &= \mathbf{0}
   \end{aligned}$$

4. For each free variable α_i or ε_i in any ${}_{k}\mathbf{T}_j$, add any additional constraints affecting that free variable. These constraints may be explicitly stated (see "**Constraints**", above) or may arise from cycles in the kinematic graph.
5. Add any addional constraints affecting any new free variables ξ_i appearing in constraints introduced in the previous step. Repeat the process until no new free variables are introduced.

The resulting linear system may then be solved to find all feasible values of $\vec{\alpha}_{ab}$ and $\vec{\epsilon}_{ab}$. The set of such solutions is a six dimensional polytope corresponding to the projection of all the constraints onto ($\vec{\alpha}_{ab}$, $\vec{\epsilon}_{ab}$)

Polytope Projection Algorithms

We have developed two efficient algorithms for computing the projection of a high dimensional polytope defined by inequality constraints into a lower dimensional space {Rajan & Taylor 1987}. The efficiency is important because as the dimensionality increases the complexity of the polytopes increases dramatically. A polytope built from n constraints in d-dimensional space will have $O(n^{d/2})$ vertices. Hence, a brute-force algorithm that projects all the vertices of the higher dimensional polytope can be very slow.

The first algorithm is inspired in part by earlier work in Robotics on determination of the shape of a convex object by repeated probes by a Robot. It has been shown that the number of probes required to completely characterize a polytope is the sum of the number of its vertices and faces {Cole and Yap 1983 ,Dobkins et. al. 1986}. For our case, the solution of a linear programming problem provides us with a plane which touches the projected polytope. Each probe reveals a vertex on the polytope and provides a linear half-space which bounds the polytope. The convex hull of the vertices lies within the projected polytope. Further probing is done parallel to the faces of this convex hull. Each additional probe either discovers a new vertex of the projected polytope or confirms a face of the polytope and when all faces are confirmed, we have the polytope we seek. In addition, the projected polytope lies within the intersection of the half-spaces from the probe. If we only want an approximation to the projected polytope, we can stop when the inscribing polytope formed by the convex hull of vertices and the circumscribing polytope formed by the intersection of half-spaces are close enough.

The second algorithm uses a dual of the standard gift wrapping method for convex hulls to determine the polytope in the lower dimensional space. It projects only those edges and facets of the higher dimensional polytope which lie on the boundary of the projection. We have developed a criterion for determining if a particular facet lies on the boundary. The complexity of this algorithm is also proportional to the number of vertices of the lower dimensional polytope.

The facets and vertices of the projected polytope (once it is computed) can be related straightforwarldy to the constraints defining the higher dimensional polytope. This information may be used by a planning system in a variety of ways. For example, it may identify particular tolerances that must be tightened in order to meet a functional requirement.

Implementation

We have implemented a complete system for representing large graphs of uncertain spatial relationships, for setting up and manipulating the corresponding systems of linear equations, and for computing projected polytopes in two dimensions, using the first algorithm described above. We are presently extending the implementation to three and higher dimensions. The polytope projection algorithm is implemented in C++ {Stroustrup, 1987} and the rest of the system is implemented in AML/X {Nackman et al., 1986}. The examples below illustrate typical experiments we have performed with the system.

Example: Robot kinematic error

Figure 3 shows projections of the kinematic uncertainty of the robot and joint configuration from Figure 2 onto $(\varepsilon_x, \varepsilon_y)$ and $(\alpha_z, \varepsilon_x)$. Similarly, Figure 4 shows projections of the kinematic repeatability onto $(\varepsilon_x, \varepsilon_y)$ and $(\alpha_z, \varepsilon_x)$. The repeatability is computed by computing $\mathbf{T}_{lab,gripper}$ for two different motion commands to the same spot, and then computing

$$\mathbf{T}_{rpt} = {}_1\mathbf{T}_{lab,gripper}^{-1} \cdot {}_2\mathbf{T}_{lab,gripper}$$

Since only the $\Delta\theta_i$ and Δd_4 are different in ${}_1\mathbf{T}_{lab,gripper}$ and ${}_2\mathbf{T}_{lab,gripper}$, the contributions of the $\vec{\alpha}_{axj}$ and the $\vec{\epsilon}_j$ cancel out. Figure 4(d) shows the AML/X code fragment used to generate the polytope projections and is intended to convey some of the flavor of the user interface to the implemented system. Essentially, this code works by attaching a "marker" coordinate system to the robot, moving it to a test point, releasing it and then making another move. The standard graph search algorithm is then called to find the relation between the marker and the robot gripper.

The solutions to the linear programming subproblems provide useful information in addition to the projected polytopes. For example, the point of maximum radial error in Figure 3(a) is (0.222, – 0.647) . By examining the corresponding vertex of the 22-variable linear programming problem, we can learn the relative sensitivity of this radial error to each variable at this point, and (possibly) use the information as input to a planner or design optimization system. In this case, the radial error is most sensitive to changes in $\alpha_{ax1,x}$ and $\alpha_{ax2,y}$, which correspond to twists in the robot links.

Example: Grasping Uncertainty

Consider the situation shown in Figure 5, in which the robot has just picked up part p from the feeder shown in Figure 1 with a small suction device at the end of its last joint. The grasping affixment -- i.e., the relation between the package and the suction gripper -- is given by:

$$\mathbf{T}_{gp} = \mathbf{T}_{lg}^{-1} \cdot \mathbf{T}_{lf} \cdot \mathbf{T}_{fp}$$

where $\mathbf{T}_{lg}$ is the robot kinematic transformation relative to the laboratory coordinate system, $\mathbf{T}_{lf}$ gives the feeder location relative to the lab, and $\mathbf{T}_{fp}$ locates the part relative to the feeder. Assuming that the feeder is nominally located at *trans*(**I**, *vector*(400,0,0)) , and the part is 5 mm thick, then the equations and constraints for $\mathbf{T}_{fp}$ and $\mathbf{T}_{lg}$ are given in Figure 1 and Figure 2 , respectively. Assume that the feeder is guaranteed to lie flat on the table, but that its position on the table may vary by $\pm 20\ \mu$m and that its orientation about the $\hat{\mathbf{z}}$ axis may vary by as much as $\pm\ 1^\circ$ Thus,

$$\mathbf{T}_{lf} = trans(\mathbf{I}, vector(400,0,0), \alpha_{lf,z}\vec{\mathbf{z}}, \vec{\epsilon}_{lf})$$

where $|\varepsilon_{lf,x}| \leq 0.020$, $|\varepsilon_{lf,y}| \leq 0.020$, $|\varepsilon_{lf,z}| \leq 0.020$, and $|\alpha_{lf,z}| \leq 1^\circ$. If we multiply everything out, we get a system with 30 bounded scalar variables, together with the eight additional constraints shown in Figure 1. Figure 5 shows the projected polytopes for $(\varepsilon_{gp,x}, \varepsilon_{gp,y})$ and $(\alpha_{gp,z}, \varepsilon_{gp,y})$. An analysis of the linear programming solutions shows that most of this error comes from the relatively loose fit of the part in the feeder tray.

Example: Sensing uncertainty

Consider the situation illustrated in Figure 6, in which the robot has moved over a viewing station, where a vision algorithm locates two corners of the part. If the relation between the part and the vision station is given by

$$\mathbf{T}_{vp} = trans(\mathbf{I}, h\hat{\mathbf{z}}, \vec{\alpha}_{vp}, \epsilon_{vp})$$

then we can model the vision step with constraints of the form:

$$|(\mathbf{T}_{vp} \cdot \vec{\mathbf{c}}_k) \cdot \vec{\mathbf{x}} - c_{k,x}| \leq \delta_{vis}$$
$$|(\mathbf{T}_{vp} \cdot \vec{\mathbf{c}}_k) \cdot \vec{\mathbf{y}} - c_{k,y}| \leq \delta_{vis}$$

for each corner k, where δ_{vis} is a property of the vision algorithm. Multiplying this out gives

$$|(\alpha_{vp} \times \vec{c}_k + \vec{\epsilon}_{vp}) \cdot \vec{x} - c_{k,x}| \leq \delta_{vis}$$
$$|(\alpha_{vp} \times \vec{c}_k + \vec{\epsilon}_{vp}) \cdot \vec{y} - c_{k,y}| \leq \delta_{vis}$$

Figure 7(a) shows the corresponding polytopes for $(\epsilon_{vp,x}, \epsilon_{vp,y})$ and $(\alpha_{vp,z}, \epsilon_{vp,y})$. Figure 7(b) shows an updated projection of $(\epsilon_{gp,x}, \epsilon_{gp,y})$. Now, the principal sources of grasping uncertainty are uncertainties in the robot's positioning, in the position of the vision station, and in the precision of the vision algorithm, rather than the loose fit of the part in its feeder. Furthermore, the subproblem solutions can tell us how sensitive the remaining uncertainty is to improvements in, say, δ_{vis}.

Example: Placement uncertainty after sensing

Finally, we move the robot to place the part on a workpiece clamped to the same holder that was used to hold the camera, as shown in Figure 6. The relation between the part and the workpiece is given by

$$\mathbf{T}_{pw} = \mathbf{T}_{gp}^{-1} \bullet \mathbf{T}_{lg}^{-1} \bullet \mathbf{T}_{lh} \bullet \mathbf{T}_{hw}$$

This relation gives a system of 49 bounded variables, with 16 additional constraints corresponding to the feeder-part relationship and the vision step. Projections of this system onto $(\epsilon_{gp,x}, \epsilon_{gp,y})$ and $(\alpha_{gp,z}, \epsilon_{gp,y})$ are shown in Figure 7(c).

Example: Frictional stability

During the planning of a robot assembly task we would like to know how stable a particular assembly of rigid bodies is. If the contact between the bodies is described by the laws of friction, we can determine the stable region from the laws of statics. {Rajan et. al. 1987, M. E. Erdmann 1984}. As the assembly becomes more complex, the task of determining if it is stable becomes more difficult, and can be done using Linear Programming {Boneschanger et al 1988}. However, for task planning it is useful have the full region of stability.

Consider an assembly of two dimensional TV-tray resting on a a rigid stand, as shown in Figure 8(a). We would like to know what external force $\vec{\mathbf{f}}_{ext} = (f_{ext,x}, f_{ext,y})$ without disturbing the stability of the structure. When the assembly becomes unstable we would like to know what kind of motion it will manifest. The four points of contact have frictional contact, and the reaction forces must lie within the cone of friction. If the reaction force at point of contact i, located at $\vec{\mathbf{r}}_i$, is $\vec{\mathbf{f}}_i = (f_{ix}, f_{iy})$ then we have following set of constraints:

$$|f_{ix}| \leq \mu f_{iy}$$

In addition, the normal reaction has to be positive. This gives a set of inequality constraints:

$$f_{iy} \geq 0$$

When the system is stable, the total force and torque on each rigid body in the assembly has to be zero. This gives a set of equality constraints:

$$\vec{\mathbf{f}}_{ext} + \vec{\mathbf{f}}_1 + \vec{\mathbf{f}}_2 + \vec{\mathbf{w}}_t = 0$$
$$\vec{\mathbf{r}}_{ext} \times \vec{\mathbf{f}}_{ext} + \vec{\mathbf{r}}_1 \times \vec{\mathbf{f}}_1 + \vec{\mathbf{r}}_2 \times \vec{\mathbf{f}}_2 + \vec{\mathbf{r}}_w \times \vec{\mathbf{w}}_t = 0$$
$$-\vec{\mathbf{f}}_1 - \vec{\mathbf{f}}_2 + \vec{\mathbf{f}}_3 + \vec{\mathbf{f}}_4 + \vec{\mathbf{w}}_s = 0$$
$$-\vec{\mathbf{r}}_1 \times \vec{\mathbf{f}}_1 - \vec{\mathbf{r}}_2 \times \vec{\mathbf{f}}_2 + \vec{\mathbf{r}}_3 \times \vec{\mathbf{f}}_3 + \vec{\mathbf{r}}_4 \times \vec{\mathbf{f}}_4 + \vec{\mathbf{r}}_s \times \vec{\mathbf{w}}_s = 0$$

where $\vec{\mathbf{w}}_t$, $\vec{\mathbf{w}}_s$ denote the gravitational forces on the tray and the stand respectively, and $\vec{\mathbf{r}}_t$ and $\vec{\mathbf{r}}_s$ denote their centers of gravity. In Figure 8(b) we show the stable region for this assembly together with the boundary showing the kinds of instability it manifests at each boundary. The nature of instability at each boundary can be inferred from the constraints which become relevant at each point on the boundary. For example, the sensitivity analysis shows that the only relevant constraint corresponding to the top edge of Figure 8(b) is $f_{2y} \geq 0$, revealing that this boundary corresponds to the kind of instability shown in the figure.

Conclusions and Future Work

We are presently debugging a three-dimensional version of the polytope projection algorithm and expect to implement higher dimensional versions as well. So far we have run the two-dimensional projection system on a moderate number of problems similar to the examples shown here. The implementation works surprisingly well even for large numbers of free variables and constriants. The largest example from this paper runs in about two minutes on an IBM RT/PC, despite considerable debugging trace output. We are considering modifications to further improve efficiency. One such improvement would be stopping when a sufficiently good approximation to the actual polytope has been found.

When we have used the system interactively, we have found that display of the full polytope is quite useful in giving a good intuition of the variations associated with a robotic task. In this regard, the sensitivity analysis coming from the linear programming subproblems is quite valuable in identifying relevant sources of error and their relative importance. Indeed, this analysis led to the discovery of several "bugs" in earlier versions of the robotic assembly example used above, when the predicted placement precision was demonstrated to be very different than one of us (Taylor) had expected intuitively. Other potential uses of the system include mechanical design and analysis of tolerences, robot task planning, sensor "fusion", subassembly stability, and similar engineering analysis tasks.

Acknowledgement

We would like to thank Steve Klinkner for his excellent implementation work on the polytope projection algorithm and on the graphics subroutines used to generate many of the figures used in this paper.

References.

1. N. Boneschanger, H. van der Drift, S. Buckley, R. Taylor "Subassembly Stability", *Proc AAAI Conference*, 1988.
2. R.A. Brooks, "Symbolic error analysis and robot planning", *International Journal of Robotics Research*, 1(4), 1982.
3. R. Cole and C. Yap, "Shape from probing", NYU-Courant Institute, Robotics Lab. No. 15, 1983.
4. D. Dobkin, H.Edelsbrunner, C.K. Yap,"Probing convex polytpes" *ACM Symposium on Theory of Computing*, p. 424 1986.
5. H. F. Durrant-White, "Consistent integration and propogation of disparate sensor observations", *International Journal of Robotics Research*, 6(3):3-24, Fall 1987.
6. M.E. Erdmann 1984. *On Motion Planning with Uncertainty*, M.S. Thesis, M.I.T., Cambridge, Mass.

7. T. Lozano-Perez and R. H. Taylor, "Geometric Issues in Planning Robot Tasks", chapter in *SDF Benchmark Symposium on Robotics* (title uncertain), MIT Press, 1988 (pending publication).

8. T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-motion Strategies for Robots", *International Journal of Robotics Research*, 3(1):3-24, 1984.

9. L. R. Nackman, M. A. Lavin, R. H. Taylor, W. C. Dietrich, "AML/X Users' Manual", IBM Research Report RA 175, IBM T. J. Watson Research Center, Yorktown Heights NY, 1986.

10. V.T. Rajan and R.H. Taylor "Geometric Tolerances and Mathematical Programing", presented at the SIAM conference on Geometric Modelling, July 1987, Albany, N.Y.

11. V.T. Rajan, R. Burridge, J.T. Schwartz "Dynamics of a rigid body in frictional contact with rigid walls, motion in two dimensions.", *Proceedings of the IEEE Conference on Robotics and Automation*, Raleigh, North Carolina, March 1987, pp.671-677.

12. R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty" *International Journal of Robotics Research*, 5(4):56-68, Winter 1986.

13. B. Stroustrup, *The C++ Programming Language*, Addison Wesley, 1987.

14. R. H. Taylor, *The Synthesis of Manipulator Control Programs from Task-level Specifications.* , Ph. D. Thesis, Technical Report AIM-282, Stanford University, Artificial Intelligence Laboratory, 1976.

$$
\begin{aligned}
-2 &\le -10\,\alpha_z + \epsilon_x \le 2\\
-1 &\le 10\,\alpha_z + \epsilon_y \le 1\\
-2 &\le -10\,\alpha_z - \epsilon_x \le 2\\
-1 &\le 10\,\alpha_z - \epsilon_y \le 1
\end{aligned}
$$

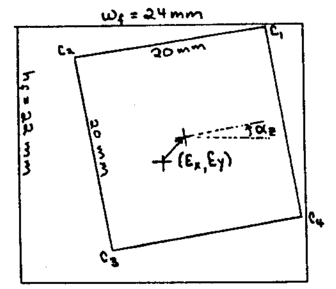
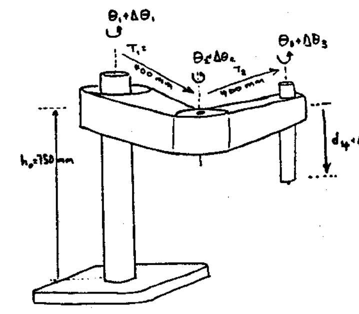


Figure 1. Constraint of a small part within a feed tray



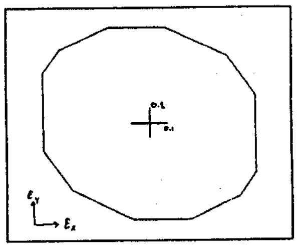
$$
\begin{aligned}
\alpha_x &= \alpha_{ax1,x} + .5\,\alpha_{ax2,x} + .87\,\alpha_{ax2,y} + .5\,\alpha_{ax3,x} - .87\,\alpha_{ax3,y}\\
\alpha_y &= \alpha_{ax1,y} - .87\,\alpha_{ax2,x} + .5\,\alpha_{ax2,y} + .87\,\alpha_{ax3,x} + .5\,\alpha_{ax3,y}\\
\alpha_z &= \alpha_{ax1,z} + \Delta\theta_1 + \alpha_{ax2,z} + \Delta\theta_2 + \alpha_{ax3,z} + \Delta\theta_3
\end{aligned}
$$

$$
\begin{aligned}
\epsilon_x = &-745\,\alpha_{ax1,y} + 645\,\alpha_{ax2,x} - 372\,\alpha_{ax2,y} - 346\,\alpha_{ax2,z} - 346\,\Delta\theta_2\\
&- 645\,\alpha_{ax3,x} - 372\,\alpha_{ax3,y} + 0.5\,\epsilon_{1,x} + .87\,\epsilon_{1,y} + 0.5\,\epsilon_{2,x} - 0.87\,\epsilon_{2,y} + \epsilon_{3,x}\\
\epsilon_y = &\ 745\,\alpha_{ax1,x} + 400\,\alpha_{ax1,z} + 372\,\alpha_{ax2,x} + 645\,\alpha_{ax2,y} + 200\,\alpha_{ax2,z} + 200\,\Delta\theta_2\\
&+ 372\,\alpha_{ax3,x} - 645\,\alpha_{ax3,y} - .87\,\epsilon_{1,x} + 0.5\,\epsilon_{1,y} + .87\,\epsilon_{2,x} + 0.5\,\epsilon_{2,y} + \epsilon_{3,y}\\
\epsilon_z = &-400\,\alpha_{ax1,y} + 346\,\alpha_{ax2,x} + 200\,\alpha_{ax2,y} + \epsilon_{1,z} + \epsilon_{2,z} + \epsilon_{3,z} + \Delta d_4
\end{aligned}
$$

$$
\begin{array}{lll}
|\alpha_{ax1,x}| \le 0.010^\circ & |\alpha_{ax2,x}| \le 0.010^\circ & |\alpha_{ax3,x}| \le 0.010^\circ\\
|\alpha_{ax1,y}| \le 0.010^\circ & |\alpha_{ax2,y}| \le 0.010^\circ & |\alpha_{ax3,y}| \le 0.010^\circ\\
|\alpha_{ax1,z}| \le 0.010^\circ & |\alpha_{ax2,z}| \le 0.010^\circ & |\alpha_{ax3,z}| \le 0.010^\circ\\
|\Delta\theta_1| \le 0.005^\circ & |\Delta\theta_2| \le 0.005^\circ & |\Delta\theta_3| \le 0.005^\circ\\
|\epsilon_{1,x}| \le 0.025mm & |\epsilon_{2,x}| \le 0.025mm & |\epsilon_{3,x}| \le 0.025mm\\
|\epsilon_{1,y}| \le 0.025mm & |\epsilon_{2,y}| \le 0.025mm & |\epsilon_{3,y}| \le 0.025mm\\
|\epsilon_{1,z}| \le 0.025mm & |\epsilon_{2,z}| \le 0.025mm & |\epsilon_{3,z}| \le 0.025mm\\
|\Delta d_4| \le 0.020mm & &
\end{array}
$$

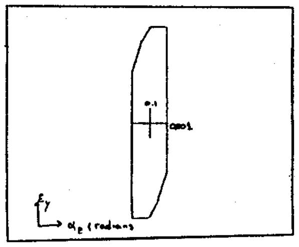
Figure 2. Typical Planar Robot: The equations for $\vec{\alpha}$ and $\vec{\epsilon}$ give the kinematic uncertainties when $\theta_1 = -60$, $\theta_2 = 120$, $\theta_3 = -60$, $d_4 = 745$ and the inequalities give bounds on the various link and control error parameters.

$$
\begin{aligned}
-0.30 &\le 0.52\epsilon_x + 0.00\epsilon_y \le 0.30\\
-0.18 &\le 0.27\epsilon_x + 0.16\epsilon_y \le 0.18\\
-0.24 &\le 0.19\epsilon_x + 0.34\epsilon_y \le 0.24\\
-0.20 &\le 0.00\epsilon_x + 0.31\epsilon_y \le 0.20\\
-0.21 &\le -0.16\epsilon_x + 0.27\epsilon_y \le 0.21\\
-0.12 &\le -0.16\epsilon_x + 0.09\epsilon_y \le 0.12
\end{aligned}
$$



(a)

$$
\begin{aligned}
-0.08 &\le 82.4\alpha_z + 0.00\epsilon_y \le 0.08\\
-0.06 &\le 0.0\alpha_z + 0.09\epsilon_y \le 0.06\\
-0.03 &\le -10.5\alpha_z + 0.05\epsilon_y \le 0.03\\
-0.04 &\le -20.9\alpha_z + 0.05\epsilon_y \le 0.04
\end{aligned}
$$



(b)

Figure 3. Positioning uncertainties for the robot configuration in Figure 2: (a) polytope and projected constraints for (ϵ_x, ϵ_y) ; (b) polytope and projected constraints for (α_z, ϵ_y) (Note that the α_z axis on the plot has been expanded 100 times)

$\alpha_{rpt,x} = 0 \qquad \alpha_{rpt,y} = 0$

$\alpha_{rpt,z} = -\,{}_1\Delta\theta_1 - {}_1\Delta\theta_2 - {}_1\Delta\theta_3 + {}_2\Delta\theta_1 + {}_2\Delta\theta_2 + {}_2\Delta\theta_3$

$\epsilon_{rpt,x} = 346\,{}_1\Delta\theta_2 - 346\,{}_2\Delta\theta_2$

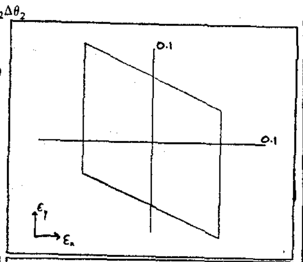
$\epsilon_{rpt,y} = -400\,{}_1\Delta\theta_1 - 200\,{}_1\Delta\theta_2 + 400\,{}_2\Delta\theta_1 + 200\,{}_2\Delta\theta_2$

$\epsilon_{rpt,z} = -\,{}_1\Delta d_4 + {}_2\Delta d_4$

(a)

$-0.004 \le 0.07\,\epsilon_{rpt,x} + 0.00\,\epsilon_{rpt,y} \le 0.004$

$-0.008 \le 0.07\,\epsilon_{rpt,x} + 0.12\,\epsilon_{rpt,y} \le 0.008$



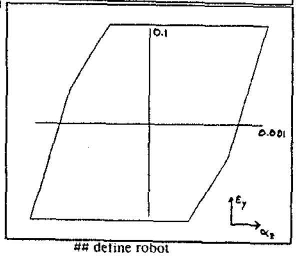
(b)

$-0.015 \le 0.00\,\alpha_{rpt,z} + 0.13\,\epsilon_{rpt,y} \le 0.015$

$-0.015 \le 0.00\,\alpha_{rpt,z} + 0.13\,\epsilon_{rpt,y} \le 0.015$

$-0.006 \le -0.07\,\alpha_{rpt,z} + 0.03\,\epsilon_{rpt,y} \le 0.006$

$-0.011 \le -0.14\,\alpha_{rpt,z} + 0.03\,\epsilon_{rpt,y} \le 0.011$

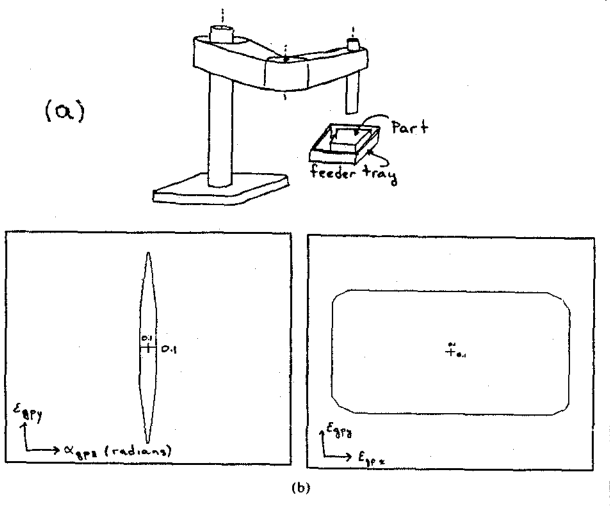


(c)

```
rob: bind scara();                                              ## define robot
marker: bind node();                                            ##
...                                                             ...
  link(marker, gripper, null__xf);                              ## attach marker to robot
  rob.mv(xf(null__rot, vec( < 400,0,5 > )))                     ## carry marker to target
  affix(marker, lab);                                           ## remember where it is
  unfix(marker, gripper);                                       ## drop it
  rob.mv(xf(null__rot, vec( < 400,0,5 > )));                    ## make another move
  compute__projected__polytope(marker, gripper, < 4,5 > );      ## (ε_rpt,x, ε_rpt,y)
  compute__projected__polytope(marker, gripper, < 4,5 > );      ## (α_rpt,z, ε_rpt,y)
```

(d)

Figure 4. Repeatability polytopes for robot configuration in Figure 2: (a) Equations for α_{rpt} and ϵ_{rpt}; (b) polytope and projected constraints for (ϵ_x, ϵ_y) ; (c) polytope and projected constraints for (α_z, ϵ_y) (Note that the α_z axis on the plot has been expanded 100 times); (d) AMLX code fragment used to generate example



(b)

Figure 5. Grasping uncertainty: (a) the robot is picking up the part from the feeder tray; (b) plots of $(\epsilon_{gpx}, \epsilon_{gpy})$ and $(\alpha_{gpz}, \epsilon_{gpx})$

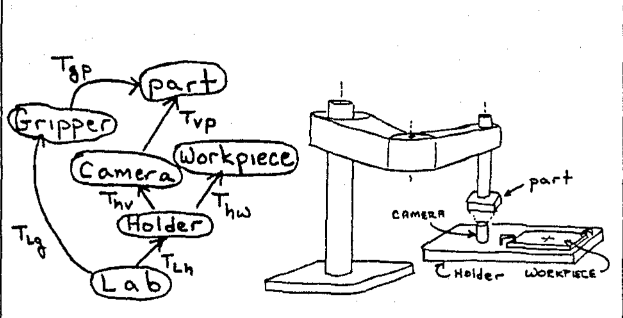


Figure 6. Vision and Assembly Stations: The graph on the left shows the relationship between the various transformations.

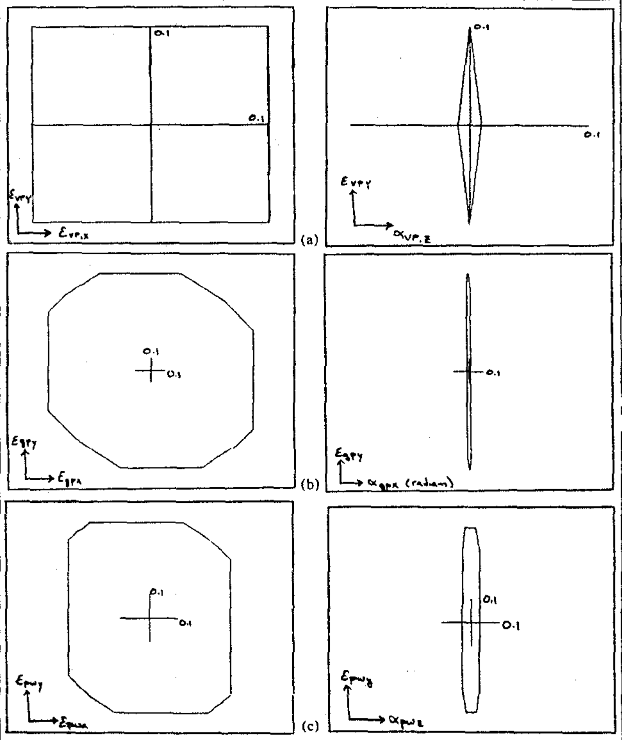


Figure 7. Projected uncertainties after vision: (a) $(\epsilon_{vp,x}, \epsilon_{vp,y})$ and $(\alpha_{vp,z}, \epsilon_{vp,y})$: (b) a updated projection of $(\epsilon_{gp,x}, \epsilon_{gp,y})$.; (c) $(\epsilon_{pw,x}, \epsilon_{pw,y})$ and $(\alpha_{pw,z}, \epsilon_{pw,y})$ after the part is moved to the workpiece.

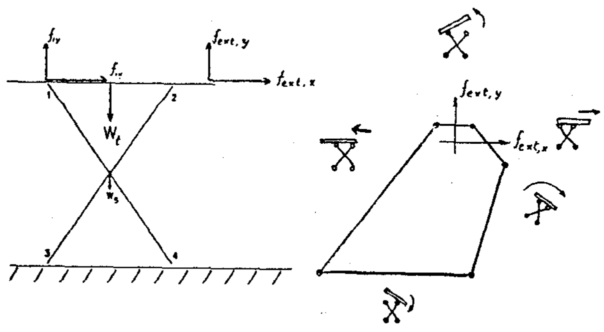


Figure 8. Frictional Stability: (a) forces on a 2D "tv table":(b) stable region for $(f_{ext,x}, f_{ext,y})$. The sketches illustrate what happens when the constraints are violated.