Spatial Modeling and Reasoning for Automatic Dimensional Inspection

Aristides A. G. Requicha and Steven N. Spitz

University of Southern California, Los Angeles, CA
Proficiency Ltd., Jerusalem, Israel

Abstract: This paper reports on a system that bridges the gap between computer aided design and metrology activities in the life cycle of a mechanical product. It completely automates the process of programming coordinate measuring machines. Accessibility of surface features to probes is efficiently computed by using computer graphics hardware. This information is used by a high-level planner based on constraint satisfaction techniques to determine setups, probes and probe orientations. High-level plans are refined so as to produce operation sequences and probe paths by using a combination of probabilistic roadmaps and travelling salesperson techniques. Experimental results are presented for industrial parts.

Key words: Coordinate Measuring Machines, CMM, quality control, process control, tolerances, accessibility, direction cones, constraint satisfaction, setup planning, path planning

1. INTRODUCTION

Dimensional inspection is a product's life-cycle activity that seeks to determine if a manufactured part is within the dimensional tolerances specified by the designer. It serves both for quality and process control: its output may be a yes/no decision on whether a workpiece is acceptable, or information that can be correlated with the parameters that control the manufacturing process. Dimensional inspection tasks in modern production systems are often performed by Coordinate Measuring Machines (CMMs),

which are versatile and well-suited to automatic operation. A CMM is essentially a very precise 3-D digitizer that produces a stream of x,y,z coordinates of points on the bounding surfaces of the object being measured. Today, CMMs are typically programmed manually, by teaching or with the help of interactive offline tools. Both methods tend to be time-consuming and error prone. The work reported in this paper automates the entire process of planning and programming CMM operations. Given a toleranced solid model of a part, plus a description of the task (as a set of surface features and associated tolerances to be checked), the system described here produces all the information required to perform the task: setups, probes, probe orientations, and probe paths. Inspection planning problems have been addressed by several researchers—see e.g. (Brown 1990, Keowon 1998, Lim 1994, Limaiem 1997, Merat 1992) and the references in (Spitz 1999)—but the planners developed thus far have been incomplete and impractical for objects of industrial complexity.

In the following sections we discuss the architecture of our planner, accessibity analysis, high-level planning, and low-level planning. Then we present experimental results, and draw conclusions.

2. ARCHITECTURE

We define an inspection task to be performed by a *measurement graph*. The root node corresponds to the task. Below the root are *tolerance* nodes, then surface *features* and finally *faces*. A path from the root to a leaf is called a *measurement*. Thus, a measurement is a triplet (tolerance, feature, face number) or, symbolically M = (T, F, #). Here, T might be a flatness tolerance, F a planar feature, and # the face number corresponding to the feature. (A feature may be composed of several faces; for simplicity of exposition we will assume that each feature has a single face associated with it, and ignore face nodes in the sequel.) The input to our planner is a model of a CMM, a set of probe models, a measurement graph, and a solid model of the workpiece, including tolerancing information equivalent to that prescribed in the ANSI Y14 standards.

We divide the planning task into two, high- and low-level planning. The high-level inspection plan (HLIP) contains information on how to setup the part on the machine, which probes to use, and which features to inspect. The HLIP is refined into a low-level plan (LLIP) in which the points to be measured are identified and sequenced, and the probe paths are computed. The two planners are described below, in Sections 4 and 5. But first we look at accessibility analysis, which is a crucial geometric computation that drives all the planning.

3. ACCESSIBILITY ANALYSIS

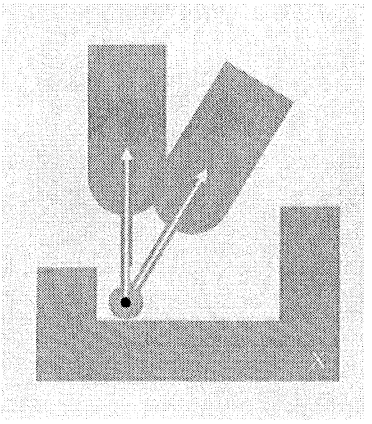

Fig. 1 – Accessibility

We say that a point on the surface of a workpiece is *accessible* to a probe if the probe tip can contact the point without otherwise colliding with the part. A surface feature is accessible if *all* of its points are accessible. Figure 1 shows a probe oriented along two different directions accessing a point on the part's surface. The set of all accessible directions constitutes a *direction cone* called a *global accessibility cone* or *GAC*. Accessibility is a necessary condition for a probe to be able to inspect a feature. However, it is not sufficient, as shown by the 2-D example of Figure 2. The probe can touch the vertical surface without colliding with the part, but it is impossible to move the probe into the desired position without collisions. In other words, the point to be inspected is not *approachable*. Accessibility does not guarantee approachability, but it can be computed fast, while approachability requires full-fledged path-planning algorithms.

We distinguish between *straight* probes, i.e., probes in which the orientation of the stylus (the thin part in Figure 1) is fixed with respect to the ram (the thicker, upper part), and orientable or *bent* probes, as in Figure 2. It is easier to compute the GAC of a point if we replace a straight probe with a semi-infinite line. Project the entire workpiece on a unit sphere centered at the point. The complement of this projection corresponds to the set of directions along which the semi-infinite line does not intersect the workpiece, i.e., to the GAC. In the actual computation, we replace the unit sphere with a cube and use standard computer graphics hardware, invoked through the OpenGL interface, to compute the projections on the six faces of the cube.

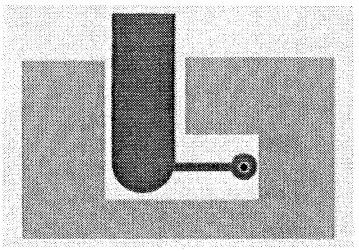

Fig. 2 - Approachability

For bent probes, accessibility computations are considerably more complicated. We need to find two inter-related GACs, one for the stylus, which can be approximated by a finite line segment, and another for the ram, which can be abstracted as a semi-infinite line. It turns out that it is possible to carry out these computations also by using graphics hardware, including z buffers. Details are given in (Spitz 2000b).

Line abstractions for probes are optimistic, because they ignore the finite thickness of the probes. Non-zero thickness can be taken into account by growing the part by the radius of the stylus (or the ram). This is an instance of the familiar robotics procedure of growing the obstacles and shrinking the robots. This growth operation is also known as a solid offset or as a Minkowski sum. Computing the boundary of the result of a Minkowski sum is expensive, but in our case we do not need to know the exact boundary. To find the directions along which there are collisions with the grown obstacle it suffices to project (polyhedral aproximations for) spheres located at the object's vertices, cylinders centered at the edges, and offset faces. As before, the complement of these directions is the desired GAC, and the computations can be done in hardware.

Fig. 3 – Example of GAC

Figure 3 provides an example of a GAC computation. Our work on accessibility analysis is described in more detail in (Spyridi 1990, 1991, 1993, 1994a, 1994b, Spitz 1999a, 1999c, 2000b).

4. HIGH-LEVEL PLANNING

We represent (a set of) high-level plans as trees, with the following levels.

Level 0: Root.
Level 1: Setups, i.e., how to orient the part on the machine table.
Level 2: Probes, i.e., which of the available physical probes to use.
Level 3: Probe orientation.
Level 4: Surface feature to be inspected.
Level 5: Measurement, i.e., (Tolerance, Feature) tuple.

A specific high-level plan corresponds to a complete traversal of the plan tree, e.g., by proceeding depth-first. We cast the planning activity into a constraint satisfaction problem (CSP). The associated variables are the measurements which constitute the leaves of the tree. Each measurement has a domain whose elements are triples of the form (s, p, o), where s is a setup, p a probe, and o a probe orientation.

We impose one hard constraint—accessibility, as an approximation for approachability—and several soft constraints that seek to ensure that the

solution is of high quality, in terms of both *efficiency* of plan execution, and *accuracy* of the measured data. These are called *same* constraints because they prescribe that measurements be made in the *same-setup*, or with the *same-probe*, or in the *same-orientation* for the probe. These constraints reflect the fact that the major sources of inaccuracies and of wasted time are changes of setup, probe and probe orientation. The *same* constraints are arranged hierarchically, and the planner attempts to satisfy the most important of them, when not all can be satisfied.

For simplicity, we assume here that the probes are straight. (How to deal with bent probes is explained in (Spitz 1999a, 1999b).) This assumption implies that the probe orientation and the setup orientation coincide, and we can represent a measurement domain by a list of (physical) probes, each with an associated direction cone containing setup directions. Planning proceeds in two conceptual phases: knowledge acquisition and plan extraction. We acquire knowledge by computing values for the variable domains, primarily through accessibility analysis. This can be done once for all features, but it is an expensive procedure, and our planner architecture allows for lazy evaluation, in which GACs are computed when needed. Given initial values for the domains, we extract an HLIP by using clustering techniques. To understand why clustering is needed, observe that two features can be inspected in the same setup with the same straight probe only if their GACs have a common direction, i.e., if they have a non-empty intersection. Typically, it is impossible to find a common direction for all GACs, and thus we look for groups of GACs that have non-empty intersections. Each group corresponds to a possible setup, which can be any direction in the group's intersection. The clustering and plan extraction operations are relatively complicated; details are given in (Spitz 1999a, 1999b).

The overall activity of the high-level planner can be summarized as follows.

Initialize the knowledge base (with probe information, GACs, etc.).
Loop until done
  Extract a plan, primarily by using clustering to solve the CSP.
  If the plan is valid, report success.
  If not, perform incremental knowledge acquisition and keep looping.

Validation is required because we make numerous approximations to ensure low computation times, especially in accessibility analysis, and some of these approximations are optimistic and may introduce false solutions. We validate the plan by using collision detection software, which executes swiftly because we (pessimistically) replace the probe and ram by grown lines (i.e., cylinders capped by hemispheres) that totally enclose the actual probe and ram.

5. LOW-LEVEL PLANNING

Construction of a low-level inspection plan (LLIP) begins with linearization of a HLIP to obtain a sequence of setups, in which several measurements are to be performed. For each feature, a set of points to be contacted by the probe are selected, and then path planning begins. (Point selection is currently done in a naïve, random and uniformly distributed pattern, but more sophisticated approaches are easy to incorporate in the planner.) The goal in path planning for each setup is to visit each of the selected points without undesirable collisions with the workpiece and as rapidly as possible. We solve this problem by combining path planning techniques from robotics with travelling salesperson (TSP) algorithms from the optimization field.

We construct a *roadmap*, which is a graph whose nodes are points to be visited, and whose arcs are collision-free paths. This is done by using an *insert-node* operator that invokes a local planner and attempts to connect by a collision-free path the node to be inserted with all the previous nodes within a specified neighborhood. The local planner may simply test line segments between nodes for collisions (using the same approach as the validator discussed earlier), or it may incorporate CMM-specific heuristics. It does not have to be powerful, but it does have to be fast. After all the desired points are inserted in the roadmap, we check to see if it is a connected graph, which implies that there exists a path through the graph that solves the problem. If not, we insert nodes randomly, and continue until we have a connected graph or no solution is found within the allotted planning time.

Once a connected roadmap is found, we use a TSP algorithm to find the shortest-distance path that visits all the points. The result is close to optimal for the specific roadmap (but not necessarily across all possible roadmaps).

6. IMPLEMENTATION AND RESULTS

The planner was implemented in C++ on a SUN Ultra 1 with Creator 3D graphics hardware, Solaris 2.5 and 128 MB of memory. For collision detection we used the RAPID system (Gottschalk 1996). The parts were modeled with the ACIS solid modeler and then approximated with polyhedra by using the ACIS mesher. The TSP was solved by using greedy algorithms discussed in (Cormen 1990).

The GAC shown in Figure 3 was found in less than 100 msec for a half-line abstraction. For a more realistic thick probe abstraction the GAC computation time increased to approximately 1 sec. A high level plan for the

part shown in Figure 4 took on the order of a minute to compute for straight probes and approximately twice as long for orientable (bent) probes. Note that the orientable probe plan executes faster, but takes longer to obtain. Low-level plans involving on the order of 100 points to be visited typically are found in about 30 seconds.

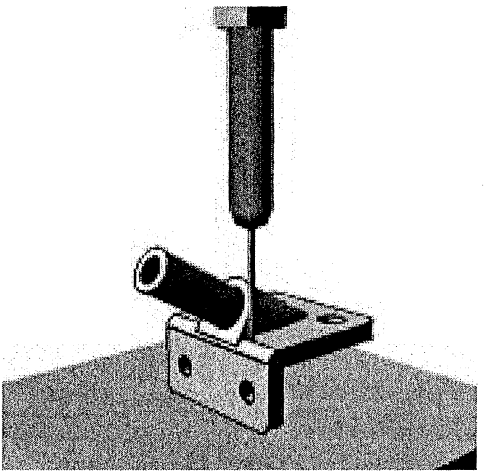

Figure 4 – Part and probe simulation

7. CONCLUSIONS

We presented an overview of a novel planner that derives automatically all the required information for inspecting a part with a CMM from a toleranced solid model of the part plus a task specified by surface features and associated tolerances to be inspected. The planner is reliable and swift, and was tested with industrial parts. Insofar as we know, it is the most powerful CMM planner ever built.

Perhaps most importantly, this research represents a shift from the traditional paradigm of geometric modeling applications, in which one attempts to solve the required geometric problems correctly and independently of their subsequent use. Here we are willing to accept approximate and even incorrect geometric solutions provided that they are obtained fast. This is possible because every plan is eventually verified and incorrect results are eliminated. In essence we are using a sophisticated version of the generate-and-test method, in which the generator is very powerful but not always right. Our experimental results suggest that this new approach is well-worth investigating for problems beyond CMM planning.

ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation under grants DMI-96-34727 and DDM-87-15404.

REFERENCES

Brown, C. W., and Gyorog, D. A. (1990). Generative inspection process planner for integrated production. In Cohen, P. H., and Joshi, S. B. (eds.): *Advances in Integrated Product Design and Manufacturing, Proceedings of the ASME Winter Annual Mtg., PED-47,* pp. 151-162.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms*, MIT Press, Cambridge, MA.

Gottschalk, S., Lin, M. C., and Manocha, D. (1996). OBB-Tree: A hierarchical structure for rapid interference detection. In Rushmeier, H. (ed.): *Proceedings ACM SIGGRAPH '96*, pp. 171-180.

Kweon, S., and Medeiros, D. J. (1998). Part orientations for CMM inspection using dimensioned visibility maps. *Computer-Aided Design*, Vol. 30, No. 9, pp. 741-749.

Lim, C. P., and Menq, C. H. (1994). CMM feature acessibility and path generation. *Robotics and Computer Integrated Manufacturing*, Vol. 32, No. 3, pp. 597-618.

Limaiem, A., and ElMaraghy, H. A. (1997). A general method for accessibility analysis. In *Proc. IEEE Int'l Conf. on Robotics and Automation,* pp. 2346-2351.

Merat, F. L., and Radack, G. M. (1992). Automatic inspection planning with a feature-based CAD system. *Robotics and Computer Integrated Manufacturing*, Vol. 9, No. 1, pp. 61-69.

Spitz, S. N. (1999a). Dimensional inspection planning for coordinate measuring machines. Ph. D. Dissertation, Computer Science Department, University of Southern California.

Spitz, S. N., and Requicha, A. A. G. (1999b). Hierarchical constraint satisfaction for high-level dimensional inspection planning. In *Proc. IEEE Int'l Symp. on Assembly & Task Planning (ISATP '99)*, pp. 374-380.

Spitz, S. N., Spyridi, A. J., and Requicha, A. A. G. (1999c). Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. *IEEE Trans. Robotics & Automation*, Vol. 15, No. 4, pp. 714-727.

Spitz, S. N., and Requicha, A. A. G. (2000a). Multiple-goals path planning for coordinate measuring machines. In *Proc. IEEE Int'l Conf. on Robotics & Automation*, pp. 2322-2327.

Spitz, S. N., and Requicha, A. A. G. (2000b). Accessibility analysis using computer graphics hardware. *IEEE Trans. Visualization & Computer Graphics*, Vol. 6, No. 3, in press.

Spyridi, A. J., and Requicha, A. A. G. (1990). Accessibility analysis for the automatic inspection of mechanical parts by coordinate measuring machines. In *Proc. IEEE Int'l Conf. on Robotics & Automation*, pp. 1284-1289.

Spyridi, A. J., and Requicha, A. A. G. (1991). Accessibility analysis for polyhedral objects. In S. G. Tzafestas (ed.): *Engineering Systems with Intelligence: Concepts, Tools and Applications*, Kluwer Academic Publishers, Inc., Dordrecht, Holland, pp. 317-324.

Spyridi, A. J., and Requicha, A. A. G. (1993). Automatic planning for dimensional inspection. *ASME Manufacturing Review*, Vol. 6, No. 4, pp. 314-319.

Spyridi, A. J. (1994a) Automatic generation of high-level inspection plans for Coordinate Measuring Machines. Ph. D. Dissertation, Computer Science Department, University of Southern California.

Spyridi, A. J., and Requicha, A. A. G. (1994b). Automatic programming of coordinate measuring machines. In *Proc. IEEE Int'l Conf. on Robotics & Automation*, pp. 1107-1112.