

Accessibility Analysis for Planning of Dimensional Inspection with Coordinate Measuring Machines

Steven N. Spitz Antonia J. Spyridi*
Aristides A. G. Requicha[†]

March 1998

Abstract

Quality and process control activities in a mechanical product's life cycle require that components be measured, or dimensionally inspected. Computer-controlled dimensional inspection is typically performed with Coordinate Measuring Machines (CMMs), which are very precise Cartesian robots that use touch probes to measure the coordinates of points on a workpiece's surfaces. Automatic planning and programming of inspection tasks with a CMM involve spatial reasoning, to determine how to orient the part on the CMM, which probes to use, how to orient the probes, and so on. This paper introduces the notions of accessibility and approachability, which are important for inspection planning, and describes two sets of implemented algorithms for computing accessibility information. One of these sets of algorithms performs exact computations on polyhedral objects and is relatively slow, whereas the other uses discrete approximations and achieves high speed by exploiting standard computer graphics hardware. The discretized algorithm has been tested on real-world parts, and is sufficiently fast for industrial applications.

1 Introduction

Mechanical components must be measured to ensure that they satisfy their design specifications. The measurement task is called *dimensional inspection*, and is part of the *quality control* activity in a product's life cycle. The results of dimensional inspection are also used in *process control*, to adjust the parameters of the manufacturing processes so as to achieve the desired results. Quality and

*Antonia Spyridi (spyridi@engr.sgi.com) is with Silicon Graphics Inc., Advanced Graphics Software Dept., Mountain View, California.

[†]Steven Spitz (spitz@usc.edu) and Ari Requicha (requicha@lipari.usc.edu) are with the Programmable Automation Laboratory, Computer Science Department, University of Southern California, Los Angeles, California, 90089-0781.

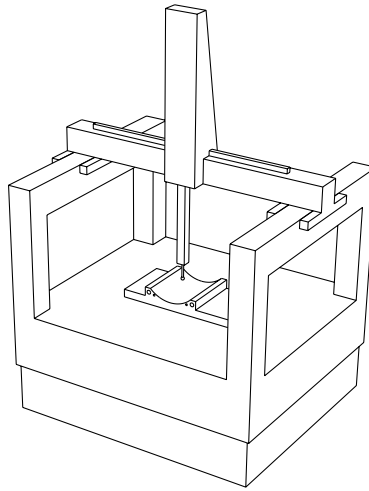


Figure 1: A typical coordinate measuring machine. The vertical component attached to the probe is called the *ram*.

process control are becoming increasingly more important, as industry strives to meet customers' demands for high-quality products.

Coordinate Measuring Machines (CMMs) are very precise three-dimensional (3-D) digitizers, which can perform most of the tasks required for dimensional inspection under computer control. Figure 1 illustrates a typical CMM configuration. The machine is essentially a Cartesian robot, equipped with a touch probe (see Figure 2). Today's CMMs are programmed primarily by *teaching*. A user manually drives the probe around the part, and records the positions to be sensed. This cannot be done before the part is manufactured, is tedious, time-consuming, and ties up expensive equipment. Off-line programming systems avoid some of these drawbacks because they work with a computer model of

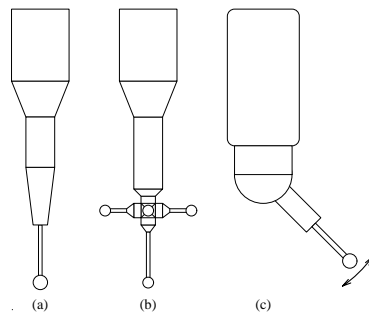


Figure 2: Common CMM probes: (a) straight probe; (b) star probe; and (c) bent, or orientable, probe.

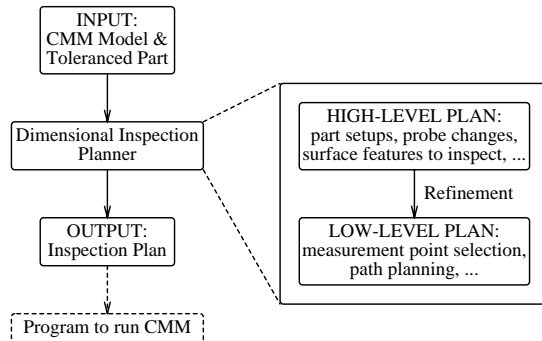


Figure 3: Architecture for dimensional inspection planner.

the part, rather than the physical part itself. But they still involve considerable manual work, and require powerful simulation and visualization capabilities. Automation of planning and programming of CMMs would cut development time and hence time-to-market, ensure consistent inspection results, and lower labor costs.

Automatic inspection is not only practically important but also scientifically interesting, because it involves spatial reasoning in one of its purest forms. Kinematics and dynamics play a minor role in CMM planning and programming, which are dominated by geometric considerations. The major issue is to ensure that a suitable set of points on each of the part’s surfaces to be measured can be reached by the probe without collisions. This is a problem in *accessibility analysis*, which is the topic of this paper.

Planning for CMM inspection can be divided into two sequential processes as shown in Figure 3. High-level planning seeks to determine how to setup the workpiece on the CMM table, which surfaces to inspect in each setup, which probes to use, and how to orient these probes. Low-level planning consists primarily of selecting a set of specific points to be measured in each surface, and generating probe paths to traverse these sets of points and execute the sensing operations. Standard algorithms from the robotics literature can be used for solving the low-level path-planning problem, but not for high-level planning. The goal of this paper is to present spatial-reasoning concepts and algorithms needed to tackle high-level planning for inspection tasks with CMMs.

The remainder of the paper is organized as follows. First we briefly survey prior work. Then we review some of the mathematical notions used in the paper. In Sections 4 and 5 we define mathematically the notions of accessibility and approachability, and introduce accessibility cones as sets of collision-free directions for orienting probes. These notions are specialized in Sections 6 and 7 to the straight and orientable (or bent) probes used for CMM inspection. Two sets of implemented algorithms for computing accessibility cones and other important entities are discussed in Section 8. A final section discusses the various abstractions and approximations used by the algorithms, and presents

conclusions.

2 Related Work

Accessibility analysis has been acknowledged as an important tool for developing automatic dimensional inspection planners. Several attempts have been made in the inspection planning community to compute global accessibility cones (or GACs, defined mathematically in Section 6), but the results are either incomplete or computationally intensive and impractical for real-world parts. This section summarizes these attempts and then points to related research in other fields that tackle similar problems. These fields include: Machining, Assembly Planning, Computer Vision, Computer Graphics and Computational Geometry.

Lim and Menq [13] describe an algorithm to compute discrete global accessibility cones using a ray casting technique, which is very slow. The discrete representation for the direction cone coincides with the 720 orientations of a Renishaw PH9 probe head [19], hence they assume that the part is in a fixed setup. Furthermore, they ignore the fact that the probe is bent, whereas the global accessibility cone is the set of accessible directions for straight probes.

Khoshnevis and Yeh [10] use 2-D accessibility for 3-D inspection planning by analyzing planar slices of the workpiece. This is an incomplete solution and it is unclear where to produce the slices in order to provide good results.

Limaïem and ElMaraghy [14, 15] compute GACs at discrete points using standard solid modelers. The algorithm iteratively slices a sphere about the point into two shells, which are then scaled to overlap and intersected. The algorithm works on arbitrary parts and provides a natural way to deal with GACs of truncated half-lines as well. The main drawback is that the algorithm is very slow, because many Boolean intersections are performed.

Local accessibility analysis studies the problem of computing accessibility directions for a surface feature, when all obstacles in the environment are ignored except for a thin layer of material including the feature [26, 27]. Work on accessibility for machining, has been done mostly in the area of local accessibility [3, 6, 11, 7]. This work deals with convex direction cones from a computational geometry point of view. For inspection, local accessibility is more useful for providing “hints” that help with more complicated computations, rather than as a tool in itself.

Accessibility is related to the concept of visibility, where the objective is to compute the set of points from which a feature is visible. This set is called the visibility region, $V(F)$, of the feature F . Cowan and Kovesi [5] introduced the separating plane concept for computing $V(F)$, and Tarabanis and Tsai [28] developed efficient algorithms that compute $V(F)$, based on this concept. $V(F)$ can be used to approximate the GAC of a feature F , and vice versa, by letting the points of $V(F)$ at infinity define directions in the GAC. The hemi-cube algorithm [4] is a method for computing visibility for graphics applications, and is similar to our discrete GAC algorithm.

Similar accessibility problems are common in Vision/laser inspection systems

[30, 16]. We have not found any system that can compute cones at the speed of those we present in this paper. Furthermore, these systems do not address the issue of bent probes.

Accessibility is also related to some approaches to Assembly Planning, which must generate directions that can be used for disassembling two objects. Wilson [31] has reported an algorithm for computing such directions. His algorithm is based on the same idea as the analytical algorithm presented in this paper, namely, using configuration space obstacles for capturing invalid directions. The algorithms, however, are different, mainly because they use different representations (for both the objects and the cones), and different methods for computing sweeps.

3 Mathematical Background

Here we review briefly a few mathematical concepts that are used in the remainder of the paper.

3.1 Topology

3-D Euclidean space with the natural topology is denoted by R^3 . The *interior*, *closure*, *boundary* and *complement* of an object $A \subseteq R^3$ are denoted by iA , clA , ∂A and A^c respectively [17]. $B(r)$ denotes the *closed ball* centered at the origin with radius r .

S^2 denotes the Gaussian (or unit) sphere. Points in S^2 correspond to vectors of unit length or directions in R^3 . We call any set of directions (i.e., a subset of S^2) a *direction cone*.

3.2 Mathematical Morphology

The *sweep* of object A along object B , also called the *dilation* of A by B or the *Minkowski sum* of A and B , is denoted by $A \oplus B$, and defined as:

$$A \oplus B = \{a + b \mid a \in A, b \in B\}. \quad (1)$$

An equivalent definition illustrates the geometric nature of “sweeping” A along B [8]:

$$A \oplus B = \bigcup_{b \in B} A_b, \quad (2)$$

where A_b denotes the *translation* of A along vector b . In other words:

$$A_b = A \oplus \{b\}. \quad (3)$$

The fundamental properties of the Minkowski sum are [8]:

$$A \oplus B = B \oplus A \quad (4)$$

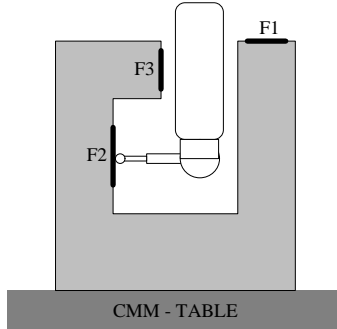


Figure 4: A CMM state with three highlighted features corresponding to three different primitive inspection plans.

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) \quad (5)$$

$$(A \cup B) \oplus C = (A \oplus C) \cup (B \oplus C) \quad (6)$$

$$(A \cap B) \oplus C \subseteq (A \oplus C) \cap (B \oplus C) \quad (7)$$

$$A \subseteq B \Rightarrow A \oplus C \subseteq B \oplus C \quad (8)$$

Growing an object A by a distance r is denoted by $A \uparrow r$, and is the same as sweeping A along a ball of radius r [21],

$$A \uparrow r = A \oplus B(r). \quad (9)$$

The *symmetric* of an object A with respect to the origin is denoted A' , and defined as:

$$A' = \{-a \mid a \in A\}. \quad (10)$$

The following result is important in collision analysis and applies to any three objects A , B and C [8]:

$$(A \oplus B) \cap C = \emptyset \iff A \cap (B' \oplus C) = \emptyset \quad (11)$$

4 The Accessibility Condition

Dimensional inspection planners use a model of the workpiece to produce a complete set of commands to drive the CMM and inspect the part. The initial task of a planner is to produce a high-level inspection plan (see Figure 3), which specifies how to setup the workpiece with respect to the CMM, which probes to use for measuring specific surface features on the boundary of the part, and how to orient the probes. This plan is composed of atomic steps called *primitive inspection plans* (PIPs). A PIP corresponds to the inspection of a *feature* in a specific *CMM state*.

A *CMM state* is a high-level description of the state of the machine. It is composed of three elements: the workpiece setup, the probe attached to the

CMM ram and the probe’s orientation. Note that the actual position of the CMM ram is considered a low-level detail and not included in the specification of a PIP. Figure 4 is an example of a CMM state. Three features are highlighted corresponding to three different PIPs that share the same CMM state.

One of the major tasks of a dimensional inspection planner is to produce PIPs for features on the boundary of the workpiece. A useful PIP with feature F and CMM state S must satisfy the *approachability condition*, i.e. it must be such that the CMM can *inspect* the entire feature F while in state S . To inspect a particular point in the feature, there must exist a collision free path for the CMM in state S , such that the probe is initially at a “safe” distance from the workpiece and in the end one of the probe’s tips comes in contact with the point. A feature is *approachable* in a specific CMM state, if the corresponding PIP satisfies the approachability condition for all the points of the feature.

A PIP that satisfies the approachability condition is called a *valid PIP*. If a planner manages to generate a high-level inspection plan that is composed entirely of valid PIPs, then we are guaranteed that a collision-free low-level refinement of this plan exists.

Unfortunately, the approachability condition is very hard to establish. Testing PIP validity can be reduced to a path planning problem that is harder than the standard FindPath [12] for several reasons: (1) a feature may have an infinite number of points to be approached; (2) the probe may have multiple tips (e.g., the star probe); and (3) the tips have a volume, and may come in contact with the same point in different positions. The last two points will not be addressed in this paper. We will concentrate on probes with single tips and assume that either the tip has no volume or that the center of the tip must be placed at specified locations that ensure contact with the feature.

To alleviate the complexity of the approachability condition we introduce a weaker concept, called accessibility. The *accessibility condition* eliminates the requirement of an approachability path, and only demands that the goal configuration be satisfied. We say that a feature F is *accessible* in a specific CMM state S , if the corresponding PIP satisfies the accessibility condition, i.e., if the tip can be placed in contact with every point of F in the state S .

Consider the three PIPs illustrated in Figure 4. $F1$ is approachable and the corresponding PIP is valid. $F2$ is accessible, but not approachable, because there is no collision free path for the probe to enter the slot from above. $F3$ is not accessible, let alone approachable in this state. Notice that a CMM with a motorized head can enter the slot with the probe oriented vertically and then re-orient it horizontally. Re-orientation during approach is almost never done in industrial CMM programming and we rule it out in this paper to simplify the planning problem. Under the simplifying assumption that the CMM state remains constant during approach, $F2$ is not approachable.

We have shown that accessibility is a weaker condition than approachability. However, it is an excellent approximation of approachability from two perspectives: quality of approximation and computational efficiency. First, in practice only rarely is a surface feature accessible but not approachable. We will see later that for PIPs involving straight probes accessibility implies approachabil-

ity. Second, we have efficient algorithms for generating PIPs that satisfy the accessibility condition, whereas just testing approachability of a single PIP can be as complex as a standard path planning problem.

The following section gives a mathematical definition of accessibility sets for abstract robots, and derives useful properties for these sets. Sections 6 and 7 study the accessibility of straight and bent probes, with the goal of producing PIPs that satisfy the accessibility condition.

5 Accessibility Sets

We now use some standard notions of robot configurations spaces to study accessibility.

The *configuration* of a robot is a specification of the position of every point in the robot relative to a fixed reference frame. The *configuration space* (c-space) of a robot corresponds to the space of all possible configurations. Every obstacle in the robot’s workspace maps to a region of c-space, called a *configuration space obstacle* (c-obstacle), that is defined as [12]:

$$\{q \in \mathcal{C} \mid \mathcal{R}(q) \cap X \neq \emptyset\}, \quad (12)$$

where \mathcal{C} is the c-space of robot \mathcal{R} , $\mathcal{R}(q)$ is the region of the workspace occupied by the robot in configuration q , and X is the region of the workspace occupied by the obstacle.

In accessibility analysis, we are interested in the complement set, also known as the *free space* of the robot.

Definition 5.1 *The accessibility set with respect to an obstacle X and robot \mathcal{R} is denoted by $\alpha(X)$ and defined as:*

$$\alpha(X) = \{q \in \mathcal{C} \mid \mathcal{R}(q) \cap X = \emptyset\}. \quad (13)$$

Consider the CMM ram/probe combination as a robot that has its tip in a fixed location, centered at the origin of the workpiece (i.e., the obstacle) coordinate system. The configuration q of such a robot is defined by the orientation of the ram and the orientation of the probe (see Figure 10 for an example). Such a configuration corresponds to the parameters that define the CMM state. Specifically, the ram direction defines the setup orientation (up to a rotation about the ram), the probe attached to the ram is known, and the probe’s orientation is given in q (up to the same rotation about the ram). Hence, the accessibility set of a CMM with an attached probe corresponds to the PIPs that satisfy the accessibility condition when the feature to be inspected is the origin. Sections 6 and 7 provide concrete examples of accessibility sets for straight and bent probes.

Note that the configuration q of a ram/probe combination does not contain translation parameters (the tip is in a fixed location). For our application in inspection planning we treat translations separately.

We now generalize the concept of accessibility sets to arbitrary features in the workspace. It is useful to visualize the definitions in terms of CMMs, however all the results in this section can be applied to arbitrary robots. From now on we refer to q simply as a configuration.

Definition 5.2 *The accessibility set of a feature F with respect to an obstacle X and robot \mathcal{R} , denoted by $\alpha(X, F)$, is:*

$$\alpha(X, F) = \{q \in \mathcal{C} \mid \forall p \in F, \mathcal{R}(q)_p \cap X = \emptyset\}. \quad (14)$$

$\mathcal{R}(q)_p$ is the result of translating the robot in configuration q by a vector p . For a CMM this implies that the tip is centered at p . Note that $\alpha(X, \{0\}) = \alpha(X)$. We say that a feature F is *accessible* in configuration q iff q is a member of $\alpha(X, F)$. Note that the accessibility set contains precisely those configurations for which *every* point of F is accessible.

Proposition 5.1 *The accessibility set of a feature F with respect to an obstacle X and robot \mathcal{R} is*

$$\alpha(X, F) = \bigcap_{p \in F} \alpha(X, \{p\}) \quad (15)$$

Proof: By definition, $q \in \alpha(X, \{p\}) \iff \mathcal{R}(q)_p \cap X = \emptyset$. Quantify both sides of this expression over all $p \in F$ to get, on the left side, the intersection of $\alpha(X, \{p\})$ and, on the right, the definition of $\alpha(X, F)$. ■

Proposition 5.2 *The accessibility set of a feature F with respect to an obstacle X and robot \mathcal{R} can be expressed in terms of sweep operations:*

$$\alpha(X, F) = \{q \in \mathcal{C} \mid (\mathcal{R}(q) \oplus F) \cap X = \emptyset\} \quad (16)$$

$$\alpha(X, F) = \alpha(F' \oplus X) \quad (17)$$

Proof: From the definition of sweep (Equation 2) and because set intersection distributes over set union, $(\mathcal{R}(q) \oplus F) \cap X = \bigcup_{p \in F} (\mathcal{R}(q)_p \cap X)$. This is the empty set iff for all $p \in F$ the set $\mathcal{R}(q)_p \cap X$ is empty, i.e., F is accessible in configuration q or $q \in \alpha(X, F)$. This proves the first equation. The second equation follows from the first, the fact that $(\mathcal{R}(q) \oplus F) \cap X = \emptyset$ iff $\mathcal{R}(q) \cap (F' \oplus X) = \emptyset$ (see Equation 11), and the definition of $\alpha(X)$ in Equation 13. ■

In particular, if F contains a single point p , then $F' \oplus X = X_{-p}$ and $\alpha(X, \{p\}) = \alpha(X_{-p})$.

Proposition 5.3 *Accessibility sets have the following properties:*

$$\alpha(X, F_1 \cup F_2) = \alpha(X, F_1) \cap \alpha(X, F_2) \quad (18)$$

$$F_1 \subseteq F_2 \Rightarrow \alpha(X, F_1) \supseteq \alpha(X, F_2) \quad (19)$$

$$\alpha(X_1 \cup X_2, F) = \alpha(X_1, F) \cap \alpha(X_2, F) \quad (20)$$

$$X_1 \subseteq X_2 \Rightarrow \alpha(X_1, F) \supseteq \alpha(X_2, F) \quad (21)$$

Proof: The properties can be directly derived from the definition of an accessibility set. ■

These properties can be used to decompose the problem of computing accessibility sets. The features may be segmented and the corresponding accessibility sets combined using Equation 18, or the obstacles in the workspace may be decomposed using Equation 20. If we wish to approximate features or obstacles using appropriate subsets or supersets, then Equations 19 and 21 tell us if the result is an upper bound or a lower bound on the true accessibility set. The former is called an *optimistic approximation* and the latter a *pessimistic* one. A pessimistic approximation discards some of the accessible configurations, whereas an optimistic one produces some inaccessible configurations.

We introduce another concept called weak accessibility. Whereas standard accessibility requires that all points in a feature be accessible, weak accessibility only requires that one point in the feature be accessible.

Definition 5.3 *The weak accessibility set of a feature F with respect to an obstacle X and robot \mathcal{R} , denoted by $\omega(X, F)$, is:*

$$\omega(X, F) = \{q \in \mathcal{C} \mid \exists p \in F, \mathcal{R}(q)_p \cap X = \emptyset\}. \quad (22)$$

Note that $\omega(X, \{0\}) = \alpha(X)$. We say that a feature F is *weakly accessible* in configuration q iff q is a member of $\omega(X, F)$.

Proposition 5.4 *The weak accessibility set of a feature F with respect to an obstacle X and robot \mathcal{R} is:*

$$\omega(X, F) = \bigcup_{p \in F} \alpha(X, \{p\}) \quad (23)$$

Proof: Dual to the proof of Proposition 5.1. The forall quantifier and set intersection operator are replaced by an existential quantifier and set union. ■

We do not know how to express weak accessibility using sweeps as we did for accessibility sets in Proposition 5.2. Therefore, we will see in Section 8 that computing weak accessibility sets is a hard problem.

Proposition 5.5 *Weak accessibility sets have the following properties:*

$$\omega(X, F_1 \cup F_2) = \omega(X, F_1) \cup \omega(X, F_2) \quad (24)$$

$$F_1 \subseteq F_2 \Rightarrow \omega(X, F_1) \subseteq \omega(X, F_2) \quad (25)$$

$$\omega(X_1 \cup X_2, F) \subseteq \omega(X_1, F) \cap \omega(X_2, F) \quad (26)$$

$$X_1 \subseteq X_2 \Rightarrow \omega(X_1, F) \supseteq \omega(X_2, F) \quad (27)$$

Proof: The properties can be directly derived from the definition of a weak accessibility set. ■

Note that Equations 24 and 25 are dual to the first two equations of Proposition 5.3. Interesting is the fact that Equation 26 is an inequality, unlike its accessibility counterpart (Equation 20).

The following two sections give concrete examples of accessibility sets for straight and bent probe abstractions. Section 8 provides algorithms to compute these sets.

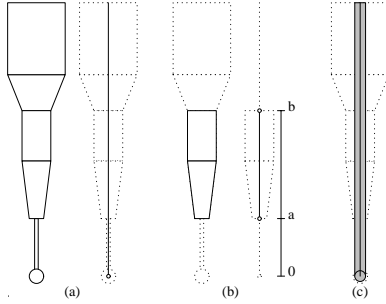


Figure 5: Straight probes are abstracted by the half-line on the main axis of symmetry (a), which is generalized to a truncated half-line that corresponds to a certain component of the probe (b) or a dilated half-line (c).

6 Straight Probe Accessibility

A *straight probe*, shown in Figure 5a, normally is attached to the CMM ram, which is much longer than the probe and aligned with its axis. We consider the whole ram/probe assembly as forming the straight probe and make no distinction between the two. The position and orientation of the probe can be expressed by the direction of the axis plus three translation parameters. Therefore, the region of workspace occupied by the probe can be expressed as $\mathcal{R}(v)_p$, where $v \in S^2$ is a direction and $p \in R^3$ is a position in space.

$\mathcal{R}(v)_p$ is defined such that the center of the tip is at p , therefore the accessibility set $\alpha(X, F)$ corresponds to the directions from which F is accessible and the center of the tip lies on F . We call any set of directions (i.e., a subset of S^2) a *direction cone*, and therefore the accessibility set for straight probes is a direction cone.

The direction of the probe corresponds to the direction of the CMM ram with respect to the workpiece. Therefore, we can use this direction to represent the orientation of the workpiece setup on the CMM table. In other words, the accessibility set of a feature defines a collection of straight probe PIPs that satisfy the accessibility condition.

This section analyzes the accessibility set for different straight probe abstractions. First, we model the probe as a half-line as shown in Figure 5a. Then, we generalize this model to two different cases: truncated half-lines (Figure 5b) and dilated half-lines (Figure 5c). The former is useful to model a component of the probe, and the latter is useful to model the volume of a probe.

6.1 Half-lines

A *half-line* is an open ray with an endpoint at the origin. If we model the probe by a half-line, then $\mathcal{R}(v) = \{tv \mid 0 < t\}$ for any direction $v \in S^2$ (we do not distinguish between a point of S^2 and the corresponding unit vector of R^3). We call the accessibility set of a half-line the *global accessibility cone* (GAC).

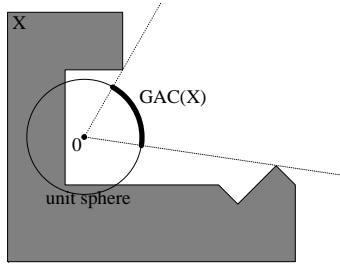


Figure 6: The global accessibility cone of a half-line with respect to an obstacle X is highlighted on the unit sphere.

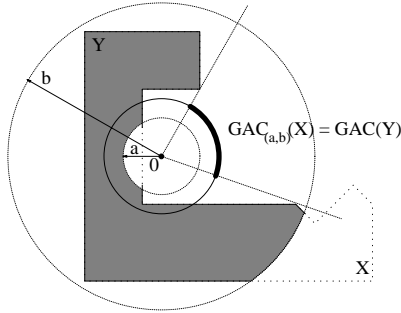


Figure 7: GAC for a truncated half-line.

In this case, $GAC(X)$ and $GAC(X, F)$ are used to denote the accessibility sets $\alpha(X)$ and $\alpha(X, F)$, respectively. $WGAC(X, F)$ is used to denote $\omega(X, F)$, or the *weak global accessibility cone* of F with respect to X .

Figure 6 illustrates the global accessibility cone of a half-line with respect to an obstacle X . $GAC(X)$ is the highlighted portion of the unit sphere. The complement of the GAC on S^2 is the projection of the set difference $X \setminus \{0\}$ onto the sphere (the origin is removed from X , because no half-line contains it and its projection is not defined). This is an important fact that will form the basis of our algorithms to compute GACs in Section 8.

6.2 Truncated Half-lines

A *truncated half-line* is an open segment (a, b) of a half-line, as in Figure 5b. Here a and b are the distances from the endpoints of the segment to the endpoint of the half-line. If we model the probe by a half-line, then $\mathcal{R}(v) = \{tv \mid a < t < b\}$ for any direction $v \in S^2$.

The accessibility set of a truncated half-line is denoted by $GAC_{(a,b)}(X)$. This is called the *global accessibility cone of a truncated half-line*. Clearly, this is a generalization of GACs because $GAC_{(0,\infty)}(X) = GAC(X)$.

Figure 7 shows that the GAC of a truncated half-line is the GAC of a half-

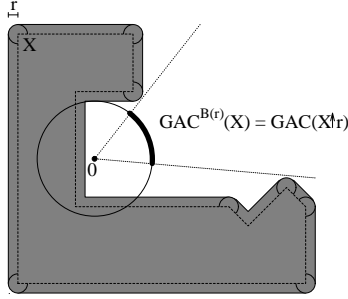


Figure 8: GAC for a dilated half-line.

line but with a different obstacle. The idea is to remove the regions of X that are not relevant. A truncated half-line (a, b) cannot collide with any portion of the obstacle that is at a distance closer than a or further than b from the origin (see Figure 7). We express this formally as:

$$\begin{cases} GAC_{(a,b)}(X) = GAC(Y) \\ Y = X \cap iB(b) \setminus B(a) \end{cases} \quad (28)$$

6.3 Dilated Half-lines

A *dilated half-line* is a half-line swept over a *structuring element* $A \subseteq R^3$, and is used to model the volume of a straight probe. The space occupied by the robot can be expressed as $\mathcal{R}(v) \oplus A$, where $\mathcal{R}(v)$ is the half-line through v . Note that the half-line model corresponds to the special case in which $A = \{0\}$.

We denote the accessibility set of a dilated half-line as $GAC^A(X)$, and call it the *global accessibility cone for a dilated half-line*. By the definition of an accessibility set and Equation 16 we conclude that $GAC^A(X) = GAC(X, A)$. In other words, the GAC for a dilated half-line is identical to the regular GAC of the structuring element A taken as a feature.

In addition, we apply Equation 17 to conclude that $GAC^A(X) = GAC(A' \oplus X)$. In particular, if A is a ball of radius r centered at the origin, then $A = A'$ and $GAC^{B(r)}(X) = GAC(X \uparrow r)$ (see Figure 8).

7 Bent Probe Accessibility

A *bent probe* is a linked chain of two components that are connected at a 2 degree-of-freedom rotary joint (see Figure 9). The CMM ram is aligned with the axis of the second component and we consider it as part of the probe.

We model the probe by a 2-component abstraction. The first component is a truncated straight probe $\mathcal{R}_1(v_1)$. The second component is a straight probe $\mathcal{R}_2(v_2)$ with its endpoint placed at the joint, at a distance d along the axis of the first component. We denote the region of workspace occupied by the bent probe as $\mathcal{R}(v_1, v_2)_p$, where v_1 and v_2 are the directions of the components and

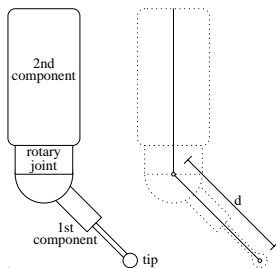


Figure 9: A bent probe (left) and its abstraction (right).

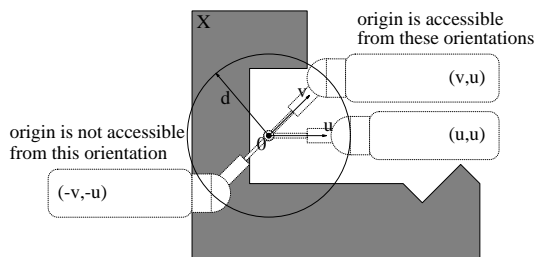


Figure 10: Accessible and non-accessible configurations.

p is the translation factor that corresponds to the position of the center of the tip. Formally, we define $\mathcal{R}(v_1, v_2)$ as:

$$\mathcal{R}(v_1, v_2) = \mathcal{R}_1(v_1) \cup \mathcal{R}_2(v_2)_{d \cdot v_1} \quad (29)$$

This is the union of the regions occupied by each component, where the second component is translated a distance d along the axis of the first component.

Figure 10 shows a bent probe in three configurations. It collides with X in configuration $(-v, -u)$, but not in configurations (u, u) and (v, u) . The last two configurations are members of the bent probe's accessibility set, $\alpha(X)$.

The accessibility set is a 4-D direction cone, which is hard to compute. Fortunately, applications normally need the directions of the second component rather than the entire 4-D cone. For example, in inspection planning for CMMs, the directions of the second component are used to find the orientations for setting up the workpiece on the machine table. Once a second component is selected, then the accessible orientations of the first component can be computed to generate possible PIPs.

We propose an upper bound for the accessibility set. This bound is the Cartesian product of a pair of direction cones, $D_1(X)$ and $D_2(X)$:

$$\begin{cases} D_1(X) = \alpha_1(X) \\ D_2(X) = \omega_2(X, d \cdot D_1(X)) \end{cases} \quad (30)$$

where $\alpha_1(X)$ is the accessibility set for the first component, \mathcal{R}_1 , and $\omega_2(X, d \cdot D_1(X))$ is the weak accessibility set of the feature $d \cdot D_1(X)$ for the second

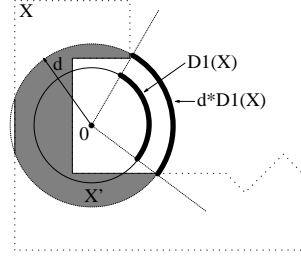


Figure 11: $D_1(X)$ contains the directions from which the first component of a bent probe abstraction can access the origin. $d \cdot D_1(X)$ is $D_1(X)$ projected on the sphere of radius d .

component, \mathcal{R}_2 . The feature $d \cdot D_1(X)$ is the projection of $D_1(X)$ on a sphere of radius d centered at the origin, or, equivalently, the result of scaling $D_1(X)$ by a factor of d (see Figures 11 and 12).

As the following proposition shows, this upper bound is tight with respect to the second component directions.

Proposition 7.1 $\alpha(X) \subseteq D_1(X) \times D_2(X)$ and $D_2(X)$ is minimal in the sense that the inclusion will not hold if $D_2(X)$ is replaced with any $D'_2(X)$ that is strictly smaller than $D_2(X)$.

Proof: (Upper bound) Take $(v_1, v_2) \in \alpha(X)$, then $\mathcal{R}(v_1, v_2) \cap X = \emptyset$, which implies by Equation 29 that: (1) $\mathcal{R}_1(v_1) \cap X = \emptyset$, so $v_1 \in \alpha_1(X) = D_1(X)$; and (2) $\mathcal{R}_2(v_2)_{d \cdot v_1} \cap X = \emptyset$, so $v_2 \in \alpha_2(X, \{d \cdot v_1\}) \subseteq \omega_2(X, d \cdot D_1(X)) = D_2(X)$, because $d \cdot v_1 \in d \cdot D_1(X)$ and by Equation 23.

(Minimality) We take an arbitrary direction $v_2 \in D_2(X)$ and find a direction $v_1 \in D_1(X)$ such that $(v_1, v_2) \in \alpha(X)$. If $v_2 \in D_2(X)$, then by definition of weak accessibility there must be a point $p \in d \cdot D_1(X)$ such that $\mathcal{R}_2(v_2)_p \cap X = \emptyset$. Define $v_1 = d^{-1} \cdot p$ (i.e., v_1 is the direction of the vector p). Then $v_1 \in D_1(X)$ and, by definition of $D_1(X)$, $\mathcal{R}_1(v_1) \cap X = \emptyset$. Furthermore, $p = d \cdot v_1$, so $\mathcal{R}_2(v_2)_{d \cdot v_1} \cap X = \emptyset$. We conclude that $\mathcal{R}(v_1, v_2) \cap X = \emptyset$, and $(v_1, v_2) \in \alpha(X)$. Now define $D'_2(X) = D_2(X) \setminus \{v_2\}$. Then $(v_1, v_2) \in \alpha(X)$ and $(v_1, v_2) \notin D_1(X) \times D'_2(X)$, which shows that $D_2(X)$ is minimal, since no direction v_2 can be removed from it. ■

We have shown that for the entire probe to access the origin, the second component direction must be in $D_2(X)$. Conversely, for every second component direction in $D_2(X)$, it is possible to orient the first component so that the probe accesses the origin.

If the bent probe is abstracted by half-lines, then the first component is a truncated half-line $(0, d)$ and $D_1(X) = GAC_{(0, d)}(X)$ (see Figure 11). The second component is a half-line with endpoint at a distance d along the axis of the first component. In this case $D_2(X) = WGAC(X, d \cdot D_1(X))$. Figure 12 shows the feature $d \cdot D_1(X)$ and the corresponding direction cone $D_2(X)$.

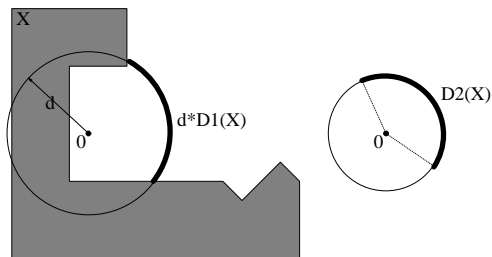


Figure 12: $D_2(X)$ is a direction cone corresponding to all possible directions of a bent probe's second component iff the probe can access the origin.

Equation 30 defined two direction cones whose Cartesian product $D_1(X) \times D_2(X)$ is an upper bound to the accessibility set $\alpha(X)$. We can extend the upper bound definition to the accessibility set of a feature F . Equation 17 showed that $\alpha(X, F) = \alpha(F' \oplus X)$. Therefore, substituting X with $F' \oplus X$ in Proposition 7.1 yields $\alpha(X, F) \subseteq D_1(F' \oplus X) \times D_2(F' \oplus X)$ and $D_2(F' \oplus X)$ is minimal. Thus, we naturally define $D_1(X, F) = D_1(F' \oplus X)$ and $D_2(X, F) = D_2(F' \oplus X)$, or, by Equation 30:

$$\begin{cases} D_1(X, F) = \alpha_1(F' \oplus X) \\ D_2(X, F) = \omega_2(F' \oplus X, d \cdot D_1(X, F)) \end{cases} \quad (31)$$

We now show how to generate bent-probe PIPs that satisfy the accessibility condition by using D_1 and D_2 . This is done in two stages. The D_2 cones are used to select orientations for the workpiece setup. Once a setup orientation v_2 is selected from D_2 , then a subset of D_1 is computed that corresponds to the orientations of the first component from which the feature is accessible when the second component is oriented along v_2 .

Notice that configurations from $D_1(X) \times D_2(X)$ are not guaranteed to be accessible. However, the minimality of $D_2(X)$ ensures that given $v_2 \in D_2(X)$ there is a $v_1 \in D_1(X)$, such that (v_1, v_2) is an accessible configuration. Let $D'_1 \subseteq D_1(X)$ be the set of all such directions for a given v_2 . Then D'_1 is not empty, and:

$$D'_1 = \{v_1 \in D_1(X) \mid \mathcal{R}_2(v_2)_{d \cdot v_1} \cap X = \emptyset\}. \quad (32)$$

If the bent probe is abstracted by half-lines, then these directions correspond to the points on $d \cdot D_1(X)$ that are not obstructed by X in the orthographic projection of $d \cdot D_1(X)$ onto a plane perpendicular to v and lying at a safe distance from X (see Figure 13). This idea is the basis for our algorithm to compute D'_1 in Section 8.

8 Implemented Algorithms

In this section we propose two methods for computing global accessibility cones: a discrete method and an analytical method. The first method uses a discrete

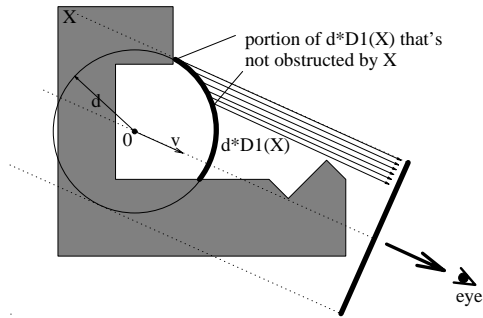


Figure 13: The parallel projection lines correspond to portions of the second component that do not collide with X and have orientation v .

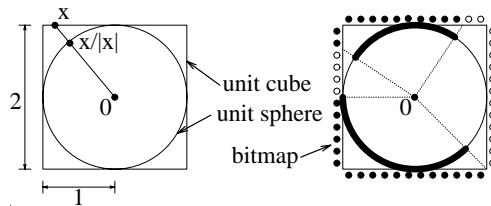


Figure 14: A direction cone is a subset of the unit sphere, which is represented by the unit cube (left). Each face of the cube is a bitmap that specifies the bits that are members of the set (right).

representation of S^2 , computes GACs of points using computer graphics hardware and samples points on surface features to approximate sweeps. In the second, direction cones are represented as boundary representations in S^2 and sweeps are implicitly computed to produce exact GACs (of polygonal features with respect to polyhedral obstacles).

We focus on the computation of the accessibility sets for straight and bent probe abstractions as defined in Sections 6 and 7. Specifically, for straight probes we want to compute $GAC(X, F)$, and for bent probes we want to compute $D_1(X, F)$, $D_2(X, F)$, and D_1^i (for a specific direction in D_2).

8.1 Discrete Method

Recall that a direction cone is a subset of the unit sphere. We represent the unit sphere by a cube of size 2 units, centered at the origin, and aligned with the main coordinate system (see Figure 14). Each face of the cube is represented by a bitmap, which will serve as a “screen” in the following computations. Boolean operations on these cones are trivial. They correspond to logical operation on bitmaps, which can be performed quickly in hardware as well as software.

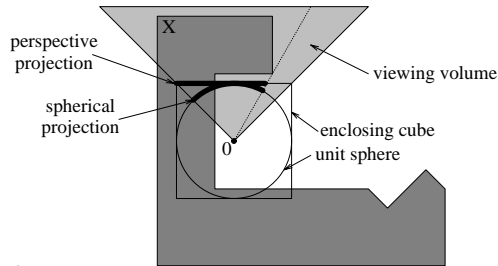


Figure 15: This example illustrates that the perspective projection of X onto the top face of the unit cube (assuming that the “eye” is at the origin) is identical to the spherical projection of X onto the unit sphere for the portion covered by the viewing volume.

8.1.1 Straight Probes

Figure 6 shows that $GAC(X)$ can be computed as the complement of the spherical projection of X onto S^2 . Our algorithm projects X onto the enclosing cube instead of the spherical projection, as shown on the top face of the cube in Figure 15.

The projection can be computed in hardware, by setting a correct perspective viewing matrix for each face, and rendering the obstacles onto these faces. If the six face bitmaps that represent the direction cone are initialized to 1s (corresponding to the entire unit sphere), then we can render the obstacles as 0s and erase them from the cone. This eliminates the need of taking the complement cone in the end of the algorithm. The pseudo-code for this algorithm is given in Figure 16.

```

DirCone GAC( Obstacle X )
1   for each face of the enclosing cube do
2       create a perspective viewing matrix as in Figure 17
3       fill the frame buffer with 1s
4       render X using 0 color bits
5       read the frame buffer into the direction cone's face
6   return the direction cone

```

Figure 16: Pseudo-code for computing discrete $GAC(X)$.

The perspective projection used for each face of the cube assumes that the eye is positioned at the origin looking through the frustum outlined by each face. In addition, we must specify near and far clipping planes. Figure 17 illustrates the viewing parameters for the face normal to the z axis at $z = 1$. The near and far clipping planes are located at $z = \nu$ and $z = \phi$, respectively. To compute $GAC(X)$ we want ν to be very small and ϕ larger than the dimensions of X 's bounding box.

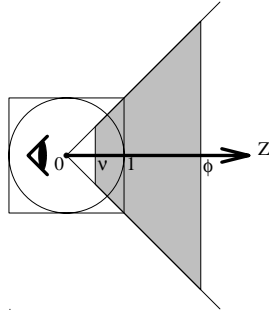


Figure 17: The viewing frustum for the perspective projection through the face at $z = 1$. The view port is the face itself. The near and far clipping planes are located at $z = \nu$ and $z = \phi$, respectively.

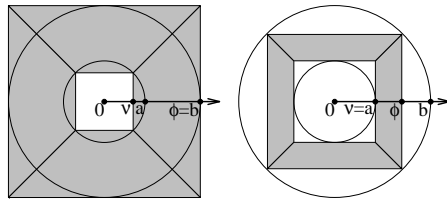


Figure 18: Near and far clipping planes to approximate the intersection of the obstacle X with the volume between spheres of radii a and b . On the left is a pessimistic approximation of $GAC_{(a,b)}(X)$ and on the right an optimistic one.

We use the z clipping planes to approximate the computation of global accessibility cones for truncated half-lines, $GAC_{(a,b)}(X)$. Equation 28 shows that the problem can be reduced to the computation of a regular GAC when we subtract from the obstacle X the ball of radius a and intersect it with a ball of radius b . The clipping planes do something similar, except that the volume removed from X is formed by cubes and not spheres. Figure 18 shows that these cubes can be used as a pessimistic approximation of $GAC_{(a,b)}(X)$ by enclosing the spheres, or an optimistic approximation if they are enclosed in the spheres.

One must be careful, because the clipping operation is not an operation on solids, but rather on the boundary of solids (the mesh that is being rendered). Hence, clipping may produce erroneous results. For example, if the origin is in the interior of X and the far clipping planes are also in X , then the result of clipping X is the empty set!

A better approximation of the intersection with the larger ball can be obtained by the addition of clipping planes, that is, by approximating the ball with an intersection of a large number of planar half-spaces. This operation is supported in standard hardware and is specified in the OpenGL standard [22]. Unfortunately, subtraction of the smaller ball produces a *concave* result that cannot be approximated by the intersection of planar half-spaces.

Alternatively, one may use a depth-buffer [22] to approximate the intersection or subtraction of a ball. The idea is to initialize the depth-buffer with the portion of the unit sphere that is visible through an arbitrary face of the cube (it is symmetric for all faces). When the obstacle is rendered with the depth-buffer enabled, then the surface in the depth-buffer functions as a clipping surface. The comparison between the depth of a pixel and the depth-buffer, i.e., “less-than” or “greater-than”, determines if the clipping operation approximates intersection or subtraction of the unit ball, respectively. Note that the depth-buffer need only be initialized once, after which it can be read-only. Also, to facilitate clipping with a ball of an arbitrary radius $r \neq 1$, the viewing matrix is scaled by r^{-1} .

The complexity of the algorithm described above depends solely on the time to render the obstacles. We eliminate the need of performing expensive Boolean operations on solids and use clipping operations to approximate them. Since the obstacle X may be rendered many times to compute GACs at different points, it is wise to optimize the mesh used to display X . For example, one can use triangle strips and a display list — both standard features in OpenGL [22].

Thus far we have shown how to compute accessibility cones for the origin with half-lines and truncated probes. Consider now global accessibility cones for a feature F . If F is a finite set of points, then Equation 15 and Equation 23 can be directly translated into algorithms. Otherwise, we sample points uniformly from F and use these equations as approximations. One can easily verify that this scheme yields an optimistic approximation for $GAC(X, F)$ and a pessimistic one for $WGAC(X, F)$. Intersections and unions of direction cones are computed by bitmap operations.

For inspection planning with CMMs, the features typically are faces of a part, and X is the entire part plus fixtures. The CMM normally probes discrete

points on X , hence selecting probing points is a natural approximation. This may not be suitable for other domains. The $GAC(X, F)$ may be computed (less efficiently) without discretizing the features by using Minkowski operations as described in the following section, but algorithms to compute $WGAC(X, F)$ without discretization are not known.

8.1.2 Bent Probes

Section 7 expressed bent probe accessibility in terms of the probe's components. The result was a pair of direction cones (D_1, D_2) that were formulated in terms of accessibility sets in Equation 31. If the probe is abstracted by half-lines, then this equation can be expressed with global accessibility cones as follows:

$$\begin{cases} D_1(X, F) = GAC_{(0,d)}(X, F) \\ D_2(X, F) = WGAC(F' \oplus X, d \cdot D_1(X, F)) \end{cases} \quad (33)$$

Using Equation 23 we can express $D_2(X, F)$ as a union of GACs:

$$D_2(X, F) = \bigcup_{p \in d \cdot D_1(X, F)} GAC(F' \oplus X, \{p\}) \quad (34)$$

Using Equation 17, $GAC(F' \oplus X, \{p\}) = GAC(\{-p\} \oplus (F' \oplus X))$. The associative and distributive properties of sweeps imply that $\{-p\} \oplus (F' \oplus X) = F' \oplus (\{-p\} \oplus X)$, which is equivalent to $F' \oplus X_{-p}$. Therefore, $GAC(F' \oplus X, \{p\}) = GAC(F' \oplus X_{-p})$. Using Equation 17 again, we get $GAC(F' \oplus X, \{p\}) = GAC(X_{-p}, F)$ and conclude that:

$$D_2(X, F) = \bigcup_{p \in d \cdot D_1(X, F)} GAC(X_{-p}, F) \quad (35)$$

We have already described how to compute GACs and X_{-p} is simply the obstacle X translated by $-p$. Therefore, we can use the previous algorithms to compute the direction cone pair (D_1, D_2) . We need to sample points from the feature $d \cdot D_1(X, F)$ in order to compute $D_2(X, F)$.

Given a direction of the second component, $v \in D_2(X)$, we showed in Equation 32 and Figure 13 that D'_1 is the set of directions corresponding to points on $d \cdot D_1(X)$ that are not obstructed by X in the orthographic projection of $d \cdot D_1(X)$ onto a plane perpendicular to v . We use this observation in our algorithm to compute the D'_1 that corresponds to v . The viewing parameters for the orthographic projection are depicted in Figure 13. We use a parallel projection with direction v and a view port large enough to enclose the projection of the ball of radius d , which is a superset of $d \cdot D_1(X)$. To check if a point on $d \cdot D_1(X)$ is obstructed by the obstacle X we use the depth-buffer [22]. First, we render X into the depth-buffer. Next, we check if a point is obstructed by X by transforming it to the viewing coordinates and comparing its depth value with the value in the depth-buffer. It is not obstructed by X iff its depth value in the appropriate depth-buffer location is closer to the viewer. The pseudo-code

for the algorithm is outlined in Figure 19. Note that $D_1(X)$ is represented by bitmaps on the faces of a cube and therefore $d \cdot D_1(X)$ is also discretized. This is another approximation used by the algorithm.

```

DirCone Project( Obstacle  $X$ , Dir  $v$ , DirCone  $D_1$ , Scalar  $d$ )
1  DirCone  $c \leftarrow \emptyset$ 
2  create orthographic viewing matrix as in Figure 13
3  clear depth-buffer  $B$ 
4  render  $X$  into  $B$ 
5  for all directions  $u \in D_1(X)$  do
6      Vector  $u' \leftarrow d \cdot u$  /*  $u'$  is of length  $d$  */
7      transform  $u'$  into viewing coordinates
8      if  $u'_z \leq B[u'_x][u'_y]$  then
9           $c \leftarrow c \cup \{u\}$ 
10 return  $c$ 

```

Figure 19: Pseudo-code of algorithm to extract the directions $u \in D_1(X)$ such that a bent probe in configuration (u, v) can access the origin when $v \in D_2(X)$.

8.1.3 Experimental Results

The algorithms described above have been implemented in C/C++ using OpenGL. We tested the code on real-world mechanical parts that were modeled with the ACIS geometric modeler [24]. The ACIS faceter produced the meshes that were used to render the parts. These meshes are a collection of convex polygons that were not optimized for rendering other than being placed in an OpenGL display list. The code executed on a Sun ULTRA 1 with Creator 3D graphics hardware, Solaris 2.1 and 124 MB of memory. Direction cones were represented by six 32×32 bitmaps, for a total of 6144 directions at a cost of 768 bytes.

The obstacle X used in Figure 20 is a non-trivial model containing 103 faces. The mesh used to render this part has 1980 convex polygons (mostly triangles). Figure 20 shows the computation of different accessibility cones. It took 0.05 seconds to compute $GAC(X)$ (the origin is marked in red). It took 0.25 seconds to compute $GAC(X, F)$ and $WGAC(X, F)$ for the face F marked in green. The face was sampled by 5 points, which explains the increase in computation time. It took 0.05 seconds to compute $D_1(X)$, which is not surprising because it uses the same algorithm as $GAC(X)$ but with different clipping planes (the length of the first component d is approximately an eighth of the diameter of X). Finally, $D_2(X)$ was computed in 0.52 seconds by sampling 10 directions from $D_1(X)$. The time it takes to unite (or intersect) direction cones is negligible.

8.2 Analytical Method

In this section we present an algorithm for computing GACs for the faces of a polyhedral workpiece W . The GACs in this case are *polyhedral direction*

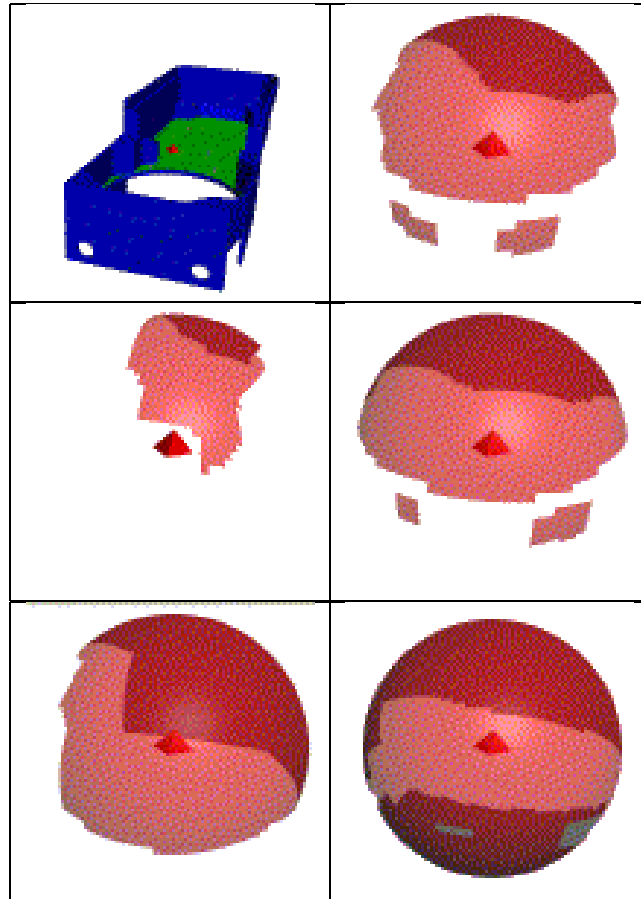


Figure 20: Experimental results of the discrete method: (top down and left to right) workpiece obstacle X with highlighted feature F (green) and origin (red); $GAC(X)$; $GAC(X, F)$; $WGAC(X, F)$; $D_1(X)$; and $D_2(X)$.

cones, i.e., direction cones that are bounded by a finite number of planes that go through the origin. If W is not polyhedral, then a polyhedral approximation is used. Note that in this case the original faces are approximated by planar segments. The GACs of these segments must be combined using Proposition 5.1.

The workpiece is represented by its boundary (BRep), and the cones are represented as 2-D sets on the surface of the unit sphere. The GACs produced are regular 2-D sets (in the relative spherical topology). Boolean operations between cones are performed as standard BRep operations. These are available with standard geometric modelers, such as ACIS [24].

8.2.1 Straight Probes

For inspection planning with CMMs we are interested in $GAC(iW, F)$. The interior of W is used so that accessible directions that are parallel to F are not lost.

Proposition 5.2 can be directly translated into an algorithm for computing $GAC(iW, F) = GAC(F' \oplus iW)$. However, to implement this algorithm, the following two problems must be addressed: (1) computations on open sets, and (2) computation of sweeps.

Our solution to the first problem is to sweep a “shrunk” version of F over W , instead of sweeping F over iW . The justification for this substitution is given by the following proposition. The proof is available in [25].

Proposition 8.1 *Let F be a planar surface feature of a solid W , and N the set of directions parallel to F . Then*

$$GAC(iW, F) \setminus N = GAC(W, i_2F), \quad (36)$$

where i_2 denotes interior in the 2-D relative topology of the planar surface on which F lies.

This proposition states that we can work with the relative interior of F , instead of the interior of W , and by doing so we only lose GAC directions that are parallel to F . These directions can be recovered by 2-D accessibility analysis, and are ignored in the following analysis. Direct usage of Proposition 8.1 does not solve the open set problem, since i_2F is an open set. Therefore, instead of the interior of F we use a new feature F_ϵ , which is a polygonal approximation of i_2F .

Now we must solve the second problem, that is, how to compute the sweep $F'_\epsilon \oplus W$. We start with the observation that what we really need is the “silhouette” of $F'_\epsilon \oplus W$ as seen from the origin (see Figure 6). But the silhouette of $F'_\epsilon \oplus W$ is the same as the silhouette of a set B , such that

$$\partial(F'_\epsilon \oplus W) \subseteq B \subseteq F'_\epsilon \oplus W, \quad (37)$$

where ∂ denotes boundary. This observation is important because we can construct a set B more efficiently than the exact boundary of $F'_\epsilon \oplus W$.

Figure 21: Generalized Gaussian Images.

The computation of the set B depends on the following property of sweeps [1]: any point on the boundary of a sweep $X \oplus Y$, can be expressed as the sum of two points x and y on ∂X and ∂Y , respectively, such that the outward pointing normal of X at x and of Y at y are the same. This is a necessary but not sufficient condition [1]. The set

$$B = \{x + y \mid x \in \partial X, y \in \partial Y, \text{normal}(x) = \text{normal}(y)\} \quad (38)$$

is larger than $\partial(X \oplus Y)$ and satisfies a form of Equation 37. Eliminating the extraneous points takes a considerable amount of work, but is not needed for silhouette computations. To use this property, we partition $\partial(F'_\epsilon)$ and ∂W into entities, and then compute the pairwise sweep of entities with the same normals. An entity can be a face, an edge or a vertex. F'_ϵ is viewed as a degenerate solid with two coincident faces with opposing outward pointing normals.

While the normal of a face is well defined, this is not the case for the normals of edges and vertices. To define the normals of these entities some preliminary notations are needed. A *neighborhood* of a point P with respect to a set X is the intersection of X with an open ball of radius r and center P [20]. A *local supporting half-space* of a point $P \in \partial X$ is a planar half-space H whose boundary contains P , and such that H contains a neighborhood of P with respect to X for arbitrarily small r . The normal of an entity, also called the *generalized Gaussian image* (GGI) of the entity, is defined by the outward-pointing normals of the local supporting half-spaces of all the points in the relative interior of the entity. For example, the GGI of a convex edge E , is a “flat cone” bounded by the normals of the faces adjacent to the edge (Figure 21a), and the GGI of a vertex V , is a “pyramid” whose edges are the normals to the faces of the convex hull of the vertex neighborhood (Figure 21b). The GGI of a face equals the Gaussian image (GI) of the face.

The analytical algorithm for computing GACs of polyhedral workpieces is given in Figure 22. The interesting thing to notice about the algorithm is that it only requires operations in 2-D. In step 5, the sweeps $M = \epsilon \oplus E$ are 2-D sets. For example, if both ϵ and E are faces, then these two faces must be parallel (because they have the same normal), therefore, $\epsilon \oplus E$ is a 2-D set parallel to ϵ and E . It is easy to see that the remaining pairs of entities also produce 2-D

```

DirCone GAC( Polyhedra  $W$ , PolygonalFeature  $F$ )
1  Shrink  $F$  to get  $F_c$ .
2  Compute  $F'_c$ .
3  Compute the GGI for each entity
   (i.e., face, edge and vertex) of  $W$  and  $F'_c$ .
4   $U \leftarrow$  all the pairs  $(e, E)$  with overlapping GGIs,
   where  $e$  denotes an entity of  $F'_c$ 
   and  $E$  an entity of  $W$ .
5  For each  $(e, E) \in U$ 
   compute  $M = e \oplus E$ .
6  For each  $M$ 
   DirCone  $C$  is the projection of  $M$  onto  $S^2$ .
7  Compute the union of all the  $C$  cones.
8  return the complement cone.

```

Figure 22: Pseudo-code for computing $GAC(iW, F)$.

sweeps (edge/vertex, vertex/vertex pairs can be ignored). Direction cones are also treated as 2-D subsets of the surface of the unit sphere.

The above algorithm can be improved, by using the results in [9], where it is shown that the sweeps of pairs face/face and edge/face are not needed. The most expensive step of the algorithm is the cone union operation. This union can be performed by using the algorithm presented in [23], according to which the union of a set of polygons can be performed in $O(m+k)$ time, where m is the total number of edges of the polygons and k the number of edge intersections. In our implementation, the union is performed by pairwise Boolean operations between cones, so this upper bound does not hold.

8.2.2 Bent Probes

To compute $D_1(iW, F) = GAC_{(0,d)}(F' \oplus iW)$ we use Equation 28, which states that $GAC_{(0,d)}(F' \oplus iW)$ equals the GAC of the origin, when the obstacle is the intersection of $F' \oplus iW$ and a ball of radius d centered at the origin (see Figure 11).

The important thing to notice is that now we must compute the exact boundary of the Minkowski sum $F' \oplus iW$, in order to generate the correct cones. This means that we can no longer use just the superset B of $\partial(F' \oplus iW)$, as it was the case for half-lines. However, B can be used as the starting point for computing the exact boundary of $F' \oplus iW$. Most of the algorithms for computing Minkowski sums ([1], [9]) work by first computing B , and then eliminating the portions of B that do not belong to the boundary of the Minkowski sum. Therefore, the set B used for computing GACs for infinite probes can be reused (with additional manipulations) for computing GACs for finite probes.

The cone $D_1(iW, F)$ need not be polyhedral, even if W is a polyhedral object. Therefore, operations on curved cones must be available for computing the exact $D_1(iW, F)$.

Figure 23: Experimental results of the analytical method.

In the current implementation we do not have the capability of generating the exact boundaries of sweeps. We compute GACs for truncated half-lines by intersecting the set B by a polyhedral approximation of the sphere. This is a *clipping* operation, which may produce incorrect results.

We do not have an algorithm for computing the exact $D_2(iW, F)$ either. We can easily compute a subset of $D_2(iW, F)$ as the union of GACs (in the presence of obstacle $F' \oplus iW$) of some sample points of the feature $d \cdot D_1(iW, F)$. Note that the same superset B of $\partial(F' \oplus iW)$, used for computing the $GAC(F, iW)$, can be used for computing the GACs of the sample points, and therefore for computing the subset of $D_2(iW, F)$. As a closing remark for this method we mention that the cone D_2 is related to the concept of weak visibility [18]. The weak visibility problem, however, has only been solved for polygons in 2-D [18, 2], and a special case in 3-D [29].

8.2.3 Experimental Results

Figure 23 shows experimental results (additional experimental results can be found in [25]). The feature of interest is shown in bold. Cone A is the GAC of the feature before adding the obstacle, and cone B is the GAC of the same feature after adding the obstacle. The bold lines indicate the boundary of the cone, when the cone is considered a subset of the unit sphere. Note that GACs need neither be connected nor convex. The time for computing each of these cones, when running on a Sun SPARCstation 10/31 with 96 MB of memory, is approximately 1 second.

8.3 Comparison of Methods

The two methods that were presented differ substantively in how they represent direction cones and compute sweeps. The discrete method is very efficient, because it uses computer graphics hardware. However, this efficiency comes at the cost of approximate solutions. The quality of the approximation increases with the resolution used to represent the direction cone, which in turn increases

the complexity of the algorithm in space and time. The quality also depends on the sampling of features in order to compute sweeps.

The analytical method is less efficient, but computes the exact GAC of a polygonal feature with respect to a polyhedral obstacle. The algorithm uses non-trivial heuristics to minimize the computation of sweeps, which are only performed in 2-D.

Both algorithms must use a polyhedral approximation of the obstacle, if it has curved surfaces. The discrete method may use a mesh of the obstacle that is optimized for rendering. Both methods can benefit from spatial acceleration schemes, but this issue is not addressed in this paper.

Both algorithms currently approximate the computation of the GAC for truncated probes using clipping operations. The analytical method performs clipping against a sphere in software, whereas the discrete method clips against a cube in hardware. The latter is a much coarser approximation, but with no added complexity to the algorithm.

Both methods use a sampling of the feature to approximate weak GACs. Computing exact WGACs is an open problem.

9 Discussion and Conclusions

Dimensional inspection planners and other spatial reasoning systems must solve complex geometric problems. Exact algorithms for solving these problems often are unavailable, or they are too expensive for real-life applications. Therefore, approximations are unavoidable. Fortunately, the solution spaces of practical planning problems are often large, and optimality (which is usually unattainable with the limited computational resources available) is not required, although the resulting plans must not be grossly suboptimal.

Pessimistic approximations may discard good solutions but they ensure that those found are correct. Optimistic approximations, on the other hand, may include incorrect solutions. And certain approximations may both miss correct solutions and include incorrect ones. The trade-offs between computational complexity and the correctness and quality of the solutions are difficult to assess. It is important to understand that geometric algorithms used in spatial reasoning are not a goal unto themselves. They are meant to be used within a planning framework. Thus, although geometric algorithms that use optimistic approximations may produce invalid results, these can be excluded if a planner includes a verification step. If a verifier fails, the plan must be discarded, and backtracking is needed. Backtracking itself is an expensive proposition, and failures must not occur often.

The algorithms described in this paper use a variety of abstractions and approximations to control the complexity of the computations. A non-exhaustive list of these approximations follows.

- Approachability is approximated by accessibility, which is more tractable. Accessibility implies approachability for straight probes, but not for general bent probes, and therefore the approximation is optimistic.

- The CMM (Coordinate Measuring Machine) is abstracted as a ram/probe assembly. This ignores possible collisions caused by remainder of the CMM, and therefore it is optimistic.
- The ram and the probe are further abstracted as bounded or semi-infinite lines. Again, this is optimistic because it ignores the finite volume of the ram and the probe, which may cause collisions. Instead, we could use the dilated-line abstraction introduced in Section 6.3, which is pessimistic, since the actual probe is enclosed by the abstraction. But this would force us to grow the workpiece by computing its solid offset [21], which is hard to do and leads to curved models, even for polyhedral workpieces.
- The discrete algorithms use a few points on a surface feature rather than the entire feature. This leads, again, to an optimistic approximation of the GAC (Global Accessibility Cone) of the feature. However, if the inspection planner first selects the sampling points, and then these are used in the GAC computation, no approximation is involved.
- The quality of the results of the discrete algorithm depends on the number of pixels of the screens (faces of a cube) used to represent the direction cones. The resolution is a parameter of the algorithm, but higher resolution implies increased computation.

How the accessibility analysis tools described here are used in our planner is beyond the scope of this paper. In brief, the general idea is to use coarse approximations and cheap computations first, and attempt to construct a plan with a small amount of computation. This plan is then verified by simulation. If the plan fails (because of collisions) or is judged too far from optimal (e.g., because of too many setups), we remove from the GACs the directions that cause the collisions and resort to better (and more expensive) approximations and search algorithms.

In summary, the notions of approachability and accessibility defined mathematically in this paper play an important role in automatic planning for dimensional inspection of mechanical parts. We show that GACs can be used to generate Primitive Inspection Plans (PIPs), which are the atomic steps in a high-level inspection plan. We describe algorithms for computing GACs both for straight and for orientable (bent) probes. Experimental results show that the implemented algorithms are sufficiently efficient to be used with real-world mechanical parts. A dimensional inspection planner that uses the accessibility tools described here is currently being implemented.

Acknowledgments

The research reported in this paper was supported in part by the National Science Foundation under grants DMI-96-34727 and DDM-87-15404. Earlier versions of portions of this paper were presented at: EURISCON '91 (European

Conference on Robotics and Intelligent Systems), Corfu, Greece, June 23-28, 1991; and the IEEE International Conference on Robotics and Automation, Cincinnati, OH, May 13-18, 1990.

References

- [1] C. Bajaj and M. Kim. Generation of configuration space obstacles: Moving algebraic surfaces. *The International Journal of Robotics Research*, 9(1):92–112, February 1990.
- [2] A. J. Briggs and B. R. Donald. Automatic sensor configuration for task-directed planning. In *IEEE International Conference on Robotics and Automation*, pages 1345–1350, May 1994.
- [3] L.-L. Chen, S.-Y. Chou, and T. C. Woo. Separating and intersecting spherical polygons: Computing machinability on three-, four-, and five-axis numerically controlled machines. *ACM Transactions on Graphics*, 12(4):305–326, October 1993.
- [4] M. F. Cohen and D. P. Greenberg. The Hemi-Cube: A radiosity solution for complex environments. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 31–40, August 1985.
- [5] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):407–416, May 1988.
- [6] J. G.-K. Gan. *Spherical Algorithms for Setup Orientation of Workpieces with Sculptured Surfaces*. PhD thesis, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, 1990.
- [7] P. Gupta, R. Janardan, J. Majhi, and T. Woo. Efficient geometric algorithms for workpiece orientation in 4- and 5-axis NC machining. *Computer-Aided Design*, 28(8):577–587, 1996.
- [8] H. J. A. M. Heijmans. *Morphological Image Operators*. Boston: Academic Press, 1994.
- [9] A. Kaul. *Computing Minkowski Sums*. PhD thesis, Columbia University, 1993.
- [10] B. Khoshnevis and Z. Yeh. An automatic measurement planning system for coordinate measuring machines. *Manufacturing Review*, 6(3):221–227, September 1993.
- [11] D.-S. Kim. *Cones on Bezier Curves and Surfaces*. PhD thesis, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI 48109-2117, 1990.

- [12] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.
- [13] C. P. Lim and C. H. Menq. CMM feature accessibility and path generation. *Robotics and Computer Integrated Manufacturing*, 32(3):597–618, 1994.
- [14] A. Limaïem and H. A. ElMaraghy. A general method for accessibility analysis. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 2346–2351, April 1997.
- [15] A. Limaïem and H. A. ElMaraghy. A general method for analysing the accessibility of features using concentric spherical shells. *The International Journal of Advanced Manufacturing Technology*, 13:101–108, 1997.
- [16] A. D. Marshall and R. R. Martin. Automatic inspection of three-dimensional geometric features. *ASME Concurrent Engineering*, 59:53–67, 1992.
- [17] B. Mendelson. *Introduction to Topology*. Allyn and Bacon, Inc., Boston, MA, 3rd edition, 1975.
- [18] J. O’Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York, 1987.
- [19] Renishaw product catalog, issue 3, 1996.
- [20] A. A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling: boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1):30–44, January 1985.
- [21] J. R. Rossignac and A. A. G. Requicha. Offsetting operations in solid modelling. *Computer Aided Geometric Design*, 3(2):129–148, August 1986.
- [22] M. Segal and K. Akeley. The OpenGL graphics system: A specification (version 1.1). Technical report, Silicon Graphics, Inc., 1997.
- [23] D. L. Souvaine and I. Bjorling-Sachs. The contour problem for restricted-orientation polygons. *Proceedings of the IEEE*, 80(9):1449–1470, September 1992.
- [24] Spatial Technology, Inc. *ACIS 3D Toolkit: Technical Overview*, August 1997.
- [25] A. J. Spyridi. *Automatic Generation of High Level Inspection Plans for Coordinate Measuring Machines*. PhD thesis, University of Southern California, Department of Computer Science, August 1994.
- [26] A. J. Spyridi and A. A. G. Requicha. Accessibility analysis for the automatic inspection of mechanical parts by coordinate measuring machines. In *IEEE International Conference on Robotics and Automation*, pages 1284–1289, Cincinnati, Ohio, May 13–18 1990.

- [27] A. J. Spyridi and A. A. G. Requicha. Accessibility analysis for polyhedral objects. In S. G. Tzafestas, editor, *Engineering Systems with Intelligence: Concepts, Tools and Applications*, pages 317–324. Dordrecht, Holland: Kluwer Academic Publishers, Inc., 1991.
- [28] K. Tarabanis and R. Y. Tsai. Viewpoint planning: the visibility constraint. In *Image Understanding Workshop*, pages 893–903, May 1989.
- [29] S. J. Teller. Computing the antipenumbra of an area light source. *Computer Graphics*, 26(2):139–148, July 1992. Proceedings SIGGRAPH 1992.
- [30] E. Trucco, M. Umasuthan, A. Wallace, and V. Roberto. Model-based planning of optimal sensor placements for inspection. *IEEE Transactions on Robotics and Automation*, 13(2):182–193, April 1997.
- [31] R. H. Wilson. *On Geometric Assembly Planning*. PhD thesis, Stanford University, Department of Computer Science, 1992.