Accessibility analysis for automatic inspection in CMMs by using bounding volume hierarchies

Braulio J Alvarez, Pedro Fernandez, J C Rico, Sabino Mateos, Carlos Manuel Suarez

Accessibility analysis for automatic inspection in CMMs by using bounding volume hierarchies

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2006-IJPR-0293.R2
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	19-Jan-2007
Complete List of Authors:	Alvarez, Braulio; University of Oviedo, Department of Manufacturing Engineering Fernandez, Pedro; University of Oviedo, Department of Manufacturing Engineering Rico, J C; University of Oviedo, Department of Manufacturing Engineering MATEOS, SABINO; UNIVERSITY OF OVIEDO, MANUFACTURING SUAREZ, CARLOS MANUEL; University of Oviedo, Department of Manufacturing Engineering
Keywords:	INSPECTION PLANNING, CMM, CLUSTERING
Keywords (user):	ACCESSIBILITY ANALYSIS, BOUNDING VOLUME HIERARCHIES

Accessibility analysis for automatic inspection in CMMs by using bounding volume hierarchies

BRAULIO J. ALVAREZ, PEDRO FERNANDEZ, *J. CARLOS RICO, SABINO MATEOS, CARLOS M. SUAREZ

Main author: J. Carlos Rico (Full Professor)
University: University of Oviedo
Department: Department of Manufacturing Engineering
Address: Department of Manufacturing Engineering
Campus de Gijón
33203 Gijón – Asturias
SPAIN
Telephone No: +34 985182062
Fax No: +34 985182016
E-mail: jcarlosr@uniovi.es

Abstract

Accessibility analysis represents one of the most critical tasks in inspection planning. This analysis determines those probe orientations that can touch an inspection point without collision. This paper presents a methodology based on part discretization and the application of space partitioning techniques (kd-tree) in order to reduce the number of intersection tests between probe and part. A STL model has been used for discretizing the inspection part in a set of triangles, which permits the application of the developed system to any type of part, regardless of its shape and complexity. Likewise, a recursive ray traversal algorithm has been used in order to speed up the traversal of the kd-tree hierarchical structure and to calculate exclusively the intersection of each probe orientation with those part triangles that potentially can interfere with it. In a further step of the analysis, the real geometry of the probe has been considered. Hence, a simplified model has been developed for each probe component (column, head, touch probe, stylus and tip) using different basic geometrical shapes. Finally, collision-free probe orientations are clustered for minimizing the orientation changes during the inspection process. Furthermore, the applied algorithm allows for determining different valid combinations of clusters. The developed system was applied to two example parts in order to prove that this methodology is adequate for the solution of real cases.

Keywords: Accessibility analysis, CMM, inspection planning, clustering, bounding volume hierarchies, collision detection, stereolithography (STL)

Accessibility analysis for automatic inspection in CMMs by using bounding volume hierarchies

1. Introduction

The application of CMMs to dimensional inspection of mechanical parts has proliferated rapidly in recent years. The main characteristics of this type of machines are their ability to acquire 3D points information, their high accuracy as well as the possibility of measuring a wide range of parts. Although they are automatic machines capable of reducing substantially the inspection time, an important effort is required in the inspection task planning and programming, mainly for avoiding human intervention. For this reason, research is being carried out in order to develop automatic inspection planning systems.

An automatic inspection planning system basically involves a series of activities such as part and inspection surface recognition, determination of number and position of inspection points, selection of probe orientation at each inspection point, measurement sequencing and probe path generation.

Among these activities, the selection of probe orientation at each inspection point stands out. For simple surfaces probe tip trajectories will not be affected by the probe orientation. However, for a more complex surface, a high number of collisions between the probe and the part may occur. Moreover, the number of needed probe orientations for the complete part inspection has to be minimal in order to reduce the number of probe orientations changes and consequently the time spent in the qualification of each probe orientation. Therefore, this paper focuses on the automatic determination of the minimal set of probe orientations that allow for measuring the part without collisions.

The determination of the valid probe orientation has been often formulated using the concept of accessibility. Spyridi and Requicha 1990 were the first to introduce this concept by means of accessibility cones. For computing such cones they divide each accessibility cone into a Local Accessibility Cone (LAC) and a Global Accessibility Cone (GAC). Abstracting the probe as an infinite half-line, the local analysis considers only the feature itself as an obstacle whereas the global analysis takes into account the entire part. From the concepts of Gaussian Image and reciprocal of a direction cone, the authors establish a series of properties in order to speed up the computation of local cones. For determining the global accessibility cones, the authors use the Minkowski sums of inspection surface and each of the local accessibility cones, checking whether the resulting sum intersects or not the entire part.

In order to reduce the computation time imposed by the Minkowski sums, Lim and Menq 1994 presented a method based on a ray-tracing algorithm to define point accessibility at points to be measured. The length of the inspection probe is assumed to be infinite and all possible orientations of the probe head are investigated. The 3D accessibility cone is transformed into a 2D map where only the probe orientations expressed by two angles in a spherical coordinate system are considered. A heuristic search is finally performed in order to determine the optimal probe direction for a set of inspected points. Limaiem and ElMaraghy 1999 also use ray-tracing techniques for the accessibility analysis. In this case, the probe is approximated by a set of cylinders. To speed up the calculation of intersections between probe and part, both are decomposed automatically into a set of Cartesian boxes by using the Octree decomposition algorithm. Finally, the system solves the problem for ordering the measurement points and selecting probe orientations by formulating the clustering problem as a classical method of sequencing and resources allocation.

Ajmal and Zhang 1998 and Zhang *et al.* 2000 also use a clustering algorithm to implement grouping of the inspection probe orientations into probe cells. This algorithm is based on the adjacent matrix representation where the columns represent the inspection features and the rows represent the probe orientations. Ajmal and Zhang 1998 use a binary matrix whereas Zhang *et al*. 2000 use weight factors as the matrix elements so that the priority of probe can be considered when the probe is selected. In both cases, a knowledge-based technique has been used in the development of the clustering algorithm module so that a block diagonal matrix can be obtained from the inspection adjacent matrix.

Ziemian and Medeiros 1997, 1998 propose a computational method based on projection techniques capable of defining regions of global accessibility for each inspection feature. The analysis is carried out for parts whose features are represented by planar surfaces. Point accessibility analysis is applied to each of the feature vertices, which represents a gross approximation of the feature boundary and, consequently, a first approximation to obtain valid probe orientations. The orientations obtained in the local feature accessibility analysis are placed at the appropriate point associated with each vertex of the feature. A collision detection algorithm determines the existence of any intersections between the probe vector and any facet of the part model by calculating the associated line-plane intersections. If an intersection is detected, a probe adjustment algorithm based on a geometrical projective technique is used to change the orientation and to determine a collision-free alternative probe orientation.

The method presented by Wu *et al.* 2004 is also based on projection techniques. They proposed an accessibility analysis for inspection of slots and holes based on the projection length of the probe onto the inspection surface. Then, comparisons between

projection lengths of the stylus, the touch probe or the whole probe orientation are established with regard to the slot sides, in order to determine possible collisions.

The problem of accessibility can be also approximated by a simpler requirement called visibility (Kweon and Medeiros 1998). A point on an object is visible from a point at infinity if there exists a straight line segment connecting the two points which does not intersect with the object. For a point on a surface, this condition can be satisfied for different orientations of the straight line segment. At a given point, these visibility orientations can be enclosed in a hemispherical space. The intersection of the hemispheres of surface points results in a visibility map (VMap) which defines the entire accessibility domain. To determine this accessibility domain at a specific point Vafaeesefat and ElMaraghy 2000a, 2000b planned to project the workpiece features surrounding this point as well as potential obstacles on a unit sphere and subtracting them from all possible domains. In order to simplify the procedure, local accessibility and visible projected faces are projected into a two-dimensional local coordinate system defined on the tangent plane at measured point. Once the point accessibility domain in 2D is defined, the feasibility of any probe direction can be checked by means of Boolean operations. Finally, the authors propose a clustering algorithm that classifies the points based on the maximum intersection between their accessibility domains.

Yin *et al.* 2000 have applied a similar method when the part to be inspected and the feasible obstacles are composed by polyhedral faces. In this work, several techniques using visibility culling are adopted in order to improve computational efficiency.

Jackman and Park 1998 introduce a generalized method for constructing an approximate VMap for a given point on the part surface. Starting with an arbitrary small hemisphere centred at the sample point, the radius of the hemisphere is increased iteratively using a

fixed step size. The step size of each iteration depends on the desired level of accuracy. A modified algorithm to generate a VMap for a given point is presented based on a discrete approximation of the workpiece surface. After finding the visible vertices, the algorithm can construct a polyhedral cone which when intersected with a unit sphere gives a VMap. Similarly, Limaiem and ElMaraghy 1997 proposed a method based on the intersection of concentric spherical shells centred at the measurement point. Each spherical shell represents sets of orientations. This shell is equivalent to the projected image in the optical analogy. Geometrical transformation (i.e. spherical scaling and solid intersection) were applied to spherical shells, and their thickness was reduced at each iteration. The resulting surface shells represented the accessibility domain. It is also possible to determine the common accessibility domain for a set of points directly or by comparing the accessibility domain of each point.

Spitz *et al.* 1999, 2000 also use a method based on the visibility concept. In this case, the authors represent the sphere by an enclosing cube providing a faster calculation. The algorithm applied by the authors for determining the GACs is similar to the hemicube algorithm used for computing visibility for graphic applications. The authors can obtain the GAC for a dilated probe with radius r by growing the obstacle by the distance r and computing the GAC for the expanded obstacle and a line probe.

Although a lot of works discuss accessibility in the context of inspection, similar concepts and algorithms are applicable to many other problems such as tool planning assembly, sensor placement for vision or machining by numerical control. From the machining point of view, Ho *et al.* 2001 and Balasubramaniam *et al.* 2003 use the concept of visibility to determine the best orientation of the tool for the cutting operation. They combine a point-cloud representation for the workpiece and a Constructive Solid Geometry (CSG) representation for the tool with an efficient

algorithm based on bounding volumes hierarchy (k-DOPs). Thus, the interference problem is reduced to simple point inclusion queries. Based on the location of the colliding points in the local coordinates of the tool, a rotational or a translational correction is performed to bring the tool out of collision.

Bounding volume interference detection is widely used in computer graphics (Klosowski *et al.* 1998) and recently its application was also found in tool interference detection in NC machining (Ding *et al.* 2004, Illushin *et al.* 2005). The bounding volumes used are usually simple geometrical shapes such as spheres and cubes. These simple geometrical components are used to enclose the objects of interest. Collision detection is checked by first checking overlaps between these simple bounding geometrical volumes. If there is no collision between these bounding volumes, there is no collision between the objects they contained. However, if collisions do occur between these bounding volumes, there is still no certainty that collision would occur between the objects they enclose.

As shown, there are a lot of contributions to solve the accessibility problem by developing different methodologies and algorithms for each case and application. In all cases, they deal with finding probe or tool orientations free of collision with regard to the workpiece. However, the high number of possible orientations and the complexity of the work environment including the part, the probe or the tool cause the computation time to increase enormously. In order to economize the calculation time, researchers apply new methodologies and algorithms and at the same time they consider certain simplifications. In dimensional inspection, these simplifications usually refer to the reductions of the number of inspection points and possible probe orientations, the use of simplified probe representations and to the inspection of simple parts or specific geometrical shapes.

The accessibility analysis presented in this paper is based on part discretization and the application of space partitioning techniques (kd-tree) in order to reduce the number of intersection tests between probe and part. The discretization of the part in a set of triangles (STL model) allows the developed algorithms to be applied to any part or environment obstacle, regardless of their geometry. A recursive ray traversal algorithm has been used in order to speed up the traversal of the kd-tree hierarchical structure and to calculate exclusively the intersection of each probe orientation with those part triangles that potentially can interfere with it. In a further step of the analysis, the real geometry of the probe has been considered. Hence, a simplified model has been developed for each probe component (column, head, touch probe, stylus and tip) using different basic geometrical shapes. Different models for each probe component have been tested to calculate their intersection with the part, and several algorithms have been implemented to accelerate the computation. Finally, collision-free orientations are clustered for minimizing the probe orientation changes during the inspection process. Furthermore, the applied algorithm allows for determining different valid combinations of clusters.

2. Accessibility analysis

The accessibility analysis deals with determining all the feasible probe orientations that allow for performing the part inspection without collision. For a part's isolated surface, probe orientations that are collision-free with that surface are enclosed in a certain space of accessibility called Local Accessibility Cone (LAC). This accessibility space can be calculated as the intersection of the LACs corresponding to all points of the surface.

In practice, this local accessibility cone is restricted by the rest of part surfaces. Therefore, in order to analyse the real accessibility of each part surface, it is necessary to

study the interaction between its LAC and the rest of part surfaces. Except for the part, any of those elements lying in the work environment could be considered as a potential obstacle, mainly the fixture for locating and/or clamping the part to be inspected. The resulting accessibility space is called Global Accessibility Cone (GAC).

2.1. Constraints for the accessibility analysis

In this work the local and global accessibility analysis will be developed for a discrete number of points located on the surfaces to inspect. This supposed simplification is particularly adequate for inspection processes on CMMs where only a discrete number of points is acquired. Also, this involves an important reduction of the computation time.

For determining the accessibility cones, constraints in the inspection system used in practice must be taken into account. The most common probe used in CMMs is an indexable probe capable of reorientation by rotating at a resolution of 7.5º about both the horizontal and vertical axes (A, B). It can rotate from 0º to 105º about axis A and from 0º to ±180º about axis B. Therefore a total of 673 different orientations can be selected for the probe. Such orientations, defined by the pair (A, B), are enclosed in a spherical space.

Also, the dimensions that characterize the different components of the probe (column, head, touch probe, stylus and tip) must be considered. Firstly, an abstraction of the probe by an infinite half-line has been used. Thus, calculations for determining geometrical intersections can be simplified. For taking the tip radius of the probe into account, the origin of the half-line will be located along the surface normal direction at a distance equal to the tip radius from the inspection point. Secondly, the dimensions of the rest of probe components will be considered.

[Insert Fig. 1 about here]

2.2. Proposed Methodology

In order to minimize computation time, this work proposes to complete the accessibility analysis in several stages (Fig. 1):

Stage 1. Local analysis considering the probe as an infinite half-line. This analysis constitutes a first method for reducing the set of valid probe orientations rapidly. The obtained orientations (A, B) will be stored in a binary matrix of accessibility.

Stage 2. Global analysis considering the probe as an infinite half-line. Only undiscarded orientations in the previous stage will be analysed. Computation of this analysis is complex and expensive, because it involves the calculation of multiple intersections between probe and part surfaces. To accelerate these calculations, several simplifications have been adopted, such as the use of a STL model for representing the part. Moreover, different computer graphics techniques like space partitioning, ray traversal algorithm, back-face culling and ray-triangle intersection tests have been also applied.

Stage 3. Global analysis considering the real shape and dimensions of the probe. The valid orientations obtained from the previous stage are checked taking into account the real probe components. For this analysis each component is modelled by means of simple geometrical shapes which are enclosed in bounding volumes to accelerate the intersection tests. Similar techniques to those previously mentioned are applied.

Stage 4. Clustering of inspection points. In order to minimize the number of probe orientation changes in the inspection process, points are grouped in a minimum number

of clusters so that all points within the same cluster can be probed with the same orientation.

Each of these stages is explained in detail in following sections.

3. Local analysis

The concept of local accessibility cone (LAC) is based on the half-space concept used often in CAD solid modelling. A half-space is any of the two spaces into which an infinite plane divides the three-dimensional space. The open half-space is the side defined to be the exterior of the solid. This is the side pointed by the normal of the infinite plane. The other side, called closed half-space, is defined to be the interior of the solid. Therefore, the local analysis determines the open half-space corresponding to the tangent plane at the inspection point. According to this, any orientation within the open half-space must be free of collision with the surface that contains the inspection point. This definition is only valid for planar or convex surfaces (Fig. 2a and 2b), but not for concave surfaces as shown in Fig. 2c.

[Insert Fig. 2 about here]

Even so, in the local accessibility analysis proposed in this work, no difference has been made between concave or convex surfaces. Therefore, the accessibility cones obtained for each point can interfere with the surface. This problem will be solved in the subsequent global accessibility analysis.

Thus the local accessibility cone in a point P will be composed of the set of all directions l which make an angle between 0 and $\pi/2$ with the surface normal n at point P. This can be expressed mathematically as follows:

$$LAC(P) = \left\{ l \mid \angle(l,n) \leq \frac{\pi}{2} \right\} \tag{1}$$

To verify whether a probe orientation l satisfy this condition, it will be sufficient to check that the next relation is satisfied:

$$\vec{l} \cdot \vec{n} \geq 0 \tag{2}$$

For each point P, the result of the analysis will be represented by a binary matrix of orientations, assigning a value of 1 to the (i, j) element when the point is locally accessible for the orientation (a_i, b_j) and a value of 0 otherwise. This accessibility matrix will be used as input information for the global analysis.

4. Global analysis considering the probe as an infinite half-line

The global accessibility cone (GAC) designates the set of probe orientations that do not collide with the part when an inspection point is probed. This definition can be extended to collisions between the probe and other elements in the environment of the inspection process. Therefore, the GAC of a point P can be defined as:

$$GAC(P) = \{ l \mid l_P \cap iW = \varnothing \} \tag{3}$$

where iW represents the interior of the part and l_P denotes a translated version of the half-line l with its origin at point P.

The computation of GAC is complex and expensive from a computational point of view because it involves the calculation of multiple intersections between the probe and the part surfaces. To make this calculation easier, a STL model of the part is used, where each surface is discretized by a set of triangles. Thus, the global accessibility analysis at

a point reduces to determine if there exist interferences between the LAC orientations at that point (see expression 1) and all the part triangles.

Likewise, this part discretization resolves the accessibility analysis of concave surfaces. Thus, the orientations within the local cones that interfere with the real surface (Fig. 3a) also interfere with the triangles that compose the STL format and therefore they will be eliminated in the global analysis (Fig. 3b).

[Insert Fig. 3 about here]

4.1. Space partitioning using kd-trees

The global analysis implies a high number of intersection tests. The use of space partitioning structures like kd-trees allows for reducing the number of tests as it involves checking intersection only with facets which can potentially be traversed by each probe orientation. The part is partitioned in regions bounded by planes and each part facet is assigned to the region within which it is located. Then, regions traversed by each probe orientation are identified and only intersections between this orientation and the facets included in these regions are tested.

This type of algorithms are similar to those of *binary space-partitioning* (BSP) tree type developed by Fuchs *et al.* 1980 for calculating the visibility of a group of objects from an arbitrary point of view.

In the kd-tree case, the division of part in regions (*bounding boxes*) is carried out by means of axis-aligned splitting planes. This way, equations of these planes as well as the distance calculation between them and the starting point of each probe orientation (inspection point) are simplified.

Different methods can be used for positioning the splitting planes:

- The planes divide successively each region into two regions of the same size.
- The planes divide each region in order to obtain two regions containing similar number of facets.
- A cost function is developed to obtain regions with similar probability of being traversed by an arbitrary ray.

The experimentation done using the last two methods did not prove more advantageous than the first one. Computation time is higher and the number of intersection tests is not reduced. Therefore, the first splitting criterion has been chosen. As regards the splitting sequence, the axis-aligned plane normal to the greater dimension of the region is always chosen in first place.

[Insert Fig. 4 about here]

The partition of the part into successive regions can be represented by a binary tree whose root node stands for the region that encloses the part completely (Fig. 4). Internal tree nodes are regions obtained in further partitions and leaf nodes represent regions into which the part is finally divided. Along with the associated region, each node of the tree stores information about the facets included, the splitting plane used for its further partitioning, and references to its children. The number of part subdivisions is equivalent to the number of levels or depth of the tree.

In order to know how many divisions have to be done or decide when a node should be declared as a leaf, some criteria must be applied. Apparently, as the space is more and more subdivided, the resulting regions are smaller; each of them will contain fewer facets and, therefore, the number of intersection tests will be reduced. However, this successive subdivision of the space involves a greater tree, which generates a higher

number of nodes to check. Similarly, a high number of facets can straddle splitting planes, so that these facets will be common to two or more regions. Therefore, the sum of all the facets contained in the different leaves of the tree increases with the depth of the tree and the number of intersection tests does not need to decrease substantially. Likewise, it will be useless to continue partitioning a region when there are no facets inside. Table 1 shows the number of triangles in the kd-tree leaves for a specific part and different number of subdivisions (depth of the tree). As it can be seen, the number of intersection tests decreases to a certain value and, then, increases.

[Insert Table 1 about here]

The process shown in Table 1 has been carried out for several parts, being impossible to find an optimal value for the maximum number of subdivisions (maximum depth of the tree) regardless of the part and the number of facets considered. For this reason, the maximum number of subdivisions will be set by the user.

Likewise, a minimum value for the number of facets has been established in order to declare a node of the tree as a leaf. Although it is desirable that the number of facets inside a region is reduced to a single one, reaching this value in practice is complex because many of the facets are common to several regions. As it is the case with the number of subdivisions (depth of tree), it is impossible to determine an optimal value for the minimum number of facets per leaf regardless of the part and the total number of facets considered.

4.2. Ray traversal algorithm

Once kd-tree has been built it is necessary to apply an algorithm for identifying the sequence of leaf nodes that are intersected by each probe orientation. This algorithm is

called *ray traversal algorithm* and was first developed and applied to a BSP tree by Kaplan 1985. The algorithm designed by Kaplan was based on repetitive computation of a point-location search along the ray path within the kd-tree.

[Insert Fig. 5 about here]

The algorithm chosen in this paper corresponds to a variant of the algorithm developed by Kaplan. In particular, an algorithm of recursive type (*recursive ray traversal algorithm*) similar to those developed by Jansen 1986 and Havran 2000 has been applied. When a ray (probe orientation) enters a interior node (region) of the kd-tree, which has two child nodes, the traversal algorithm decides if both of them are to be traversed by the ray and in which order (Fig. 5). According to the position of the origin of the ray with regard to the splitting plane, the algorithm classifies the child nodes of the current interior node as "near" and "far" child nodes. Therefore three possible cases can be found:

- When the ray traverses only "near" child node (ray (1) in Fig. 5), the algorithm descends to this node and recurses applying itself to this new node.

- When the ray has to visit both child nodes (ray (2) in Fig. 5), the algorithm saves the information about the "far" child node and descends to the "near" child node to repeat the checking process. When no facet is found to be intersected inside the "near" child node, the "far" child node is retrieved and the algorithm recurses, starting at the "far" child node.

- When the ray traverses only the "far" child node (ray (3) in Fig. 5), the algorithm descends to this node and recurses applying itself to this new node.

In order to determine if the ray only traverses the near child node, the far child node or both child nodes, the algorithm compares the signed distances from the ray origin to the splitting plane (*t*) and to the entry (*a*) and exit (*b*) point of the ray with regard to the node (Fig. 5):

- If a < b < t, then the ray only traverses the near child node.
- If a < t < b, then the ray traverses both child nodes.
- If t < a < b, then the ray only traverses the far child node.

When traversal of the tree reaches a leaf node, the probe orientation is checked for intersection with the facets inside that node (see section 4.4). If intersection exists, the probe orientation is considered not valid. Otherwise, it will be necessary to analyse the intersection of this orientation with the rest of the nodes that it traverses. An orientation will be valid if no intersection with any of the facets included in the traversed nodes is detected.

4.3. Back-face culling

Before checking intersection between each probe orientation and all facets included in the previously determined regions, the number of intersection tests can be further reduced by applying a back-face culling algorithm (Foley *et al.* 1997). Thus, a subset of visible triangles is extracted from the initial set of facets included in the traversed regions. This subset does not include triangles whose visibility is completely blocked by other triangles according to an analysed probe orientation.

[Insert Fig. 6 about here]

The example shown in Fig. 6 explains this algorithm. The possible collision between the probe orientation l with respect to a generic facet F_i is analysed for the inspection of the point P. The surfaces marked with a cross represent invisible surfaces when viewed from P in the direction of the probe, so they will not be analysed. However, the rest of surfaces are visible from P and they must be analysed to check whether or not they interfere with the probe. In practice, identification of visible facets will be carried out verifying the angle between the facet normal direction and the probe orientation. If this angle is greater than $\pi/2$, it is necessary to determine if there is interference between facet and probe. On the contrary, the facet will be discarded for the final intersection test.

4.4. Intersection between probe orientations and triangular facets of the part

Implementation of previous algorithms have allowed for reducing the set of facets to check for each probe orientation. Finally, intersection test between each undiscarded facet and probe orientation is accomplished.

[Insert Fig. 7 about here]

Next, the algorithm used to check intersection between a facet, defined by its vertices V_0, V_1 and V_2 and normal unitary vector $\vec{n}$, and any probe orientation $\vec{l}$ located at the inspection point P_0, is described. If next equation (Fig. 7) is satisfied:

$$\vec{n} \cdot \vec{l} = 0 \tag{4}$$

the orientation will be parallel to the supporting plane of the triangle and, therefore, there will be no intersection. If both expression (4) and next condition are satisfied:

$$\vec{n} \cdot \vec{w} = 0 \tag{5}$$

then the probe orientation will be contained in the plane. In expression (5) $\vec{w}$ is a vector with origin at vertex V_0 and end at point P_0 (Fig. 7). When this orientation intersects or coincides with any of the triangle edges, then, intersection between probe and triangle occurs.

If none of the previous relationships is fulfilled, then there is intersection between the probe orientation $\vec{l}$ and the supporting plane of the triangle. The intersection point P_i can be expressed as (Fig. 7):

$$P_i = P_0 - \frac{\vec{n} \cdot \vec{w}}{\vec{n} \cdot \vec{l}} \cdot \vec{l} \tag{6}$$

Finally, it is necessary to check if this point P_i lies inside the triangle defined by the three vertices V_0, V_1 and V_2. This verification is based on the algorithm developed by Möller and Trumbore 1997. The equation of the supporting plane of the triangle V_0, V_1 and V_2 can be expressed as:

$$V(s,t) = V_0 + s \cdot \vec{u} + t \cdot \vec{v} \tag{7}$$

where $\vec{u}$ and $\vec{v}$ are two edge vectors of the triangle with common origin at V_0. A point P_i located on the plane (7) will be inside the triangle if there exist values s_i and t_i that satisfies the next equation:

$$P_i - V_0 = s_i \cdot \vec{u} + t_i \cdot \vec{v} \tag{8}$$

where $s_i \geq 0$, $t_i \geq 0$ and $s_i + t_i \leq 1$.

The values of the parameters s_i and t_i can be determined from the following expressions:

$$s_i = \frac{(\vec{u} \cdot \vec{v}) \cdot (\vec{r}_i \cdot \vec{v}) - (\vec{v} \cdot \vec{v}) \cdot (\vec{r}_i \cdot \vec{u})}{(\vec{u} \cdot \vec{v})^2 - (\vec{u} \cdot \vec{u})(\vec{v} \cdot \vec{v})} \tag{9}$$

$$t_i = \frac{(\vec{u}\cdot\vec{v})\cdot(\vec{r}_i\cdot\vec{u})-(\vec{u}\cdot\vec{u})\cdot(\vec{r}_i\cdot\vec{v})}{(\vec{u}\cdot\vec{v})^2-(\vec{u}\cdot\vec{u})(\vec{v}\cdot\vec{v})} \tag{10}$$

where $\vec{r}_i$ is the vector with origin at V_0 and end at the intersection point P_i.

As it was the case with local cones, the matrix of orientations is updated by assigning a value of 1 to the (i, j) element if the point is globally accessible for the orientation (a_i, b_j) and a value of 0 otherwise.

[Insert Fig. 8 about here]

5. Global analysis considering probe dimensions

The probe orientations obtained in previous sections are based on an ideal representation of the probe as an infinite half-line. Thus the orientation shown in Fig. 8a is considered valid because the infinite half-line does not intersect the part whereas the orientation shown in Fig 8b is considered not valid because the infinite half-line does intersect the part. To solve these problems, it is necessary to take into account the real shape and dimensions of the probe.

In this section, possible collisions with each of the probe components are analysed: tip, stylus, touch probe, head and column (Fig. 9a).

[Insert Fig. 9 about here]

5.1. Probe components modelling

To analyse possible collisions between part and probe, intersections between triangles of the STL part model and each of the probe components must be checked. This calculation is speeded up by using a simplified model of each probe component. The probe head has been modelled by a sphere, the column by a straight prism, and the touch

probe and the stylus-tip set by geometrical shapes like capsules (Fig. 9b). In the particular cases of touch probe and stylus, other geometrical models, such as cylinders or even groups of spheres, have been tested (Fig. 10). Also, in the case of the column a cylindrical model has been initially tested. Among the possible models, those giving rise to a lesser computation time for solving model-triangle intersection tests have been finally chosen (see section 5.3).

[Insert Fig. 10 about here]

5.2. Bounding boxes associated to probe components

According to sections 4.1 and 4.2, a kd-tree algorithm has been implemented in order to calculate exclusively the intersection between the probe orientation and the triangles included in the regions that it traverses. Similarly, for checking the possible interference between the part and each probe component it is also advisable to reduce the intersection tests avoiding analysis of all the part facets.

To carry out this task effectively, each component is enclosed in a bounding volume and only the part regions that interfere with that volume are analysed. Then, intersections between facets included in these regions and the component are checked.

The space partitioning algorithm (section 4.1) splits up the part into a set of regions or bounding boxes whose faces are aligned with coordinate axes. Probe components are also approximated by bounding boxes whose faces are parallel to those of the bounding boxes of the part, and that will be built for each probe orientation. Fig. 10 shows the bounding boxes enclosing different geometrical shapes used for modelling the cylindrical components of the probe.

With this structure it is very easy to determine if a bounding box of the part is overlapped or not by a bounding box of the probe. It will be sufficient to check the relation between the coordinates of their extreme corners. In general, two bounding boxes *B1* and *B2* do not overlap *iff*:

$$x_{\min}^{B1} > x_{\max}^{B2} \;\; or \;\; y_{\min}^{B1} > y_{\max}^{B2} \;\; or \;\; z_{\min}^{B1} > z_{\max}^{B2} \tag{11}$$

or

$$x_{\min}^{B2} > x_{\max}^{B1} \;\; or \;\; y_{\min}^{B2} > y_{\max}^{B1} \;\; or \;\; z_{\min}^{B2} > z_{\max}^{B1} \tag{12}$$

Otherwise, *B1* and *B2* overlap and it will be necessary to check the potential intersection between the triangles included in the bounding box of the part and the probe component included in the other bounding box.

5.3. Intersection between part triangles and probe components

In this section the procedure for testing final intersection between each previously undiscarded facet and each probe component is described. Since several alternatives to model the probe components have been considered, different algorithms for checking intersections are analysed in terms of computational time.

[Insert Fig. 11 about here]

The *cylinder-triangle intersection test* algorithm projects each triangle onto a plane perpendicular to the cylinder axis (Held 1997 and Schneider and Eberly 2003). Initially, the algorithm checks if the triangle (its vertices) lies inside the region bounded by the top and bottom planes of the cylinder (Fig. 11). None of the vertices of triangles T_1 and T_2 are located between planes P_1 and P_2, so these triangles do not intersect with the cylinder. On the other hand, triangles T_3, T_4, T_5, T_6, T_7 and T_8 have some or all of its

vertices inside the region between planes P_1 and P_2. In this case, triangles T_3, T_6, T_7 and T_8 are projected onto the supporting plane of the cylinder base, as well as the polygons obtained by trimming triangles T_4 and T_5 with planes P_1 and P_2, respectively. Then, the cylinder-triangle intersection test simply determines whether intersection between the triangle (or polygon) edges and a circle (base of the cylinder) exists or not. If the edges lie inside the circle (T_7) or intersect it (T_3), there will be intersection between cylinder and triangle. The particular case of a triangle like T_8 that crosses the cylinder axis is not feasible at this stage. If a probe orientation crosses a triangle, this orientation is discarded during the analysis described in section 4, where the probe orientation was abstracted as an infinite half-line. Since the direction of the cylinder axis and the probe orientation are coincident, it is not possible to find a cylinder whose axis crosses a triangle in the current analysis. The application of this algorithm to the particular cases of touch probe and stylus require the coordinates of the triangle to be transformed into the local coordinate system of the cylinder in order to adapt the triangle to each probe orientation. These continuous coordinate system transformations make the computation time increase considerably.

For this reason, in a second approach it was decided to model the cylindrical components (touch probe and stylus) using a set of spheres, because the sphere center coordinates are not affected by probe orientation changes and all points on the surface are equidistant from the center. Therefore, intersection test simply calculates the minimum distance between a point (sphere center) and a triangle (Schneider and Eberly 2003). If this distance is smaller than the radius of the sphere, there will be intersection. However, although the computation time needed for calculating triangle-sphere intersection is faster than triangle-cylinder intersection, the total calculation time

increases because several spheres are needed to model the stylus and the touch probe (Fig. 10).

The chosen option modelled the stylus and the touch probe by a capsule shape, a set of points equidistant from a line segment (Fig. 10). This characteristic allows for using very fast intersection algorithms, based on the triangle-segment minimum distance calculation (Eberly 2000). In this case, intersection analysis is based on finding the minimum distance between each edge of the triangle and the capsule line segment, as well as the minimum distance between each extreme point of the capsule segment and the triangle. If any of these distances is smaller than the radius of the capsule, then there will be intersection. Also, it has allowed the combined analysis of the stylus and the tip of the probe.

[Insert Fig. 12 about here]

Concerning the probe column, it was firstly modelled as a cylinder. The algorithms used for determining cylinder-triangle intersection have been previously explained. In order to reduce the computation time a prismatic model was tested, which implies a tighter representation of the column real shape. To analyse the potential prism-triangle intersection, an algorithm derived from the *separating axis theorem* (Gottschalk 1996) was used. The theorem states that two convex polyhedra, *A* and *B* (Fig. 12), are disjoint if they can be separated along either an axis parallel to a normal of a face of either *A* or *B*, or along an axis formed from the cross product of an edge from *A* with an edge from *B*. The application of this theorem to a prism-triangle test involves checking their relative position with regard to the following potential separation axes (Möller 2001) (Fig. 12):

- The normal direction of the supporting plane of the triangle: $\vec{N} = \vec{B}_1 \times \vec{B}_2$

- The direction of each triangle edge: $\vec{B}_1, \vec{B}_2, \vec{B}_3$

- The directions formed from the cross products: $\vec{A}_i \times \vec{B}_j$ for $1 \le i \le 3$ and $1 \le j \le 3$

If none of these 13 directions is effectively a separation axis, it can be concluded that the triangle and the prism overlap. On the contrary, if a direction is found to be a separating axis, the algorithm finishes and it can be concluded that no intersection exists. To determine if a specific direction is a separating axis, the strategy applied projects over this direction the center of the prism and the vertices of the triangle. If the projection of the greatest distance $\vec{T}$ between the prism center and the triangle vertices onto $\vec{L}$ is greater than the sum of radii r_A and r_B, then intersection will not occur (Fig. 12).

6. Valid orientations common to all the inspection points of the part

From the previous analysis, the probe orientations within the global accessibility cone for each inspection point have been determined. These GAC orientations for each point P are represented by means of a binary matrix, where each element corresponds to a combination of discrete values of A and B angles:

$$A\left(a_i, b_j\right) = \begin{cases} 1 & \text{if } a_i \text{ and } b_j \text{ represents a valid orientation} \\ 0 & \text{if } a_i \text{ and } b_j \text{ represents a not valid orientation} \end{cases} \tag{13}$$

To reduce the operation time related to probe orientation changes, probe orientations (a_i, b_j) common to the greatest feasible number of inspection points must be found. The algorithm used is similar to that developed by Vafaeesefat and ElMaraghy 2000.

In order to simplify the explanation of the algorithm, only four inspection points will be considered, P_1, P_2, P_3 and P_4. Likewise, only four possible orientations (a_i, b_j) will be

considered for the probe. Therefore, the global accessibility matrices could be as follows:

$$A_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},\ A_2 = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix},\ A_3 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \text{ and } A_4 = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \tag{14}$$

From these accessibility matrices, clusters C_i^k are determined. Initially (k = 1), these clusters will be the same as each of the inspection points:

$$C_1^1 = \{P_1\},\ C_2^1 = \{P_2\},\ C_3^1 = \{P_3\} \text{ and } C_4^1 = \{P_4\} \tag{15}$$

Each of these clusters is associated with an accessibility matrix A_i^k, that initially coincide with the accessibility matrices of the inspection points:

$$A_1^1 = A_1,\ A_2^1 = A_2,\ A_3^1 = A_3,\ A_4^1 = A_4 \tag{16}$$

Starting from the clusters C_i^k and from the accessibility matrices A_i^k, a new matrix $CI^k(i, j) = A_i^k \cap A_j^k$ is built showing the common orientations to the clusters two against two:

$$CI^{k=1} = \left(\begin{array}{c:c:c:c} A_1^1 & A_1^1 \cap A_2^1 & A_1^1 \cap A_3^1 & A_1^1 \cap A_4^1 \\ \hdashline & A_2^1 & A_2^1 \cap A_3^1 & A_2^1 \cap A_4^1 \\ \hdashline & & A_3^1 & A_3^1 \cap A_4^1 \\ \hdashline & & & A_4^1 \end{array}\right) = \left(\begin{array}{cc:cc:cc:cc} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ \hdashline & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 1 & 1 & 1 & 1 & 0 & 0 \\ \hdashline & & & & 1 & 1 & 1 & 1 \\ & & & & 1 & 1 & 0 & 0 \\ \hdashline & & & & & & 1 & 1 \\ & & & & & & 0 & 0 \end{array}\right) \tag{17}$$

The sum of 1 values of each cluster or element (i, j) of this matrix represents the quantity H of possible valid orientations for each cluster. This quantity can be expressed by means of a matrix H^k, that will be:

$$H^{k=1} = \begin{pmatrix} 3 & 2 & 3 & 1 \\ & 2 & 2 & 0 \\ & & 4 & 2 \\ & & & 2 \end{pmatrix} \tag{18}$$

With the purpose of creating clusters whose points are associated with the greatest number of valid orientations, the algorithm searches the cluster above the main diagonal that corresponds to the maximum value of matrix H^k inside matrix CI^k. In the example, the cluster (i = 1, j = 3) contains $H_{\max}$ = 3 valid orientations. As a result, new clusters are generated (k = 2) so that cluster $C^1_{j=3}$ becomes empty and their points are joined to cluster $C^1_{i=1}$:

$$C_1^2 = C_1^1 \cup C_3^1 = \{P_1, P_3\},\ C_2^2 = \{P_2\},\ C_3^2 = \{\varnothing\} \text{ and } C_4^2 = \{P_4\} \tag{19}$$

The new accessibility matrices for each cluster will be:

$$A_1^2 = CI^1(1,3) = A_1^1 \cap A_3^1,\ A_2^2 = A_2^1,\ A_3^2 = \varnothing,\ A_4^2 = A_4^1 \tag{20}$$

With these new clusters, matrix $CI^{k=1}$ is updated to $CI^{k=2}(i,j) = A_i^2 \cap A_j^2$

$$CI^{k=2} = \begin{pmatrix} A_1^2 & A_1^2 \cap A_2^2 & \varnothing & A_1^2 \cap A_4^2 \\ & A_2^2 & \varnothing & A_2^2 \cap A_4^2 \\ & & \varnothing & \varnothing \\ & & & A_4^2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 1 & 1 & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & 0 \\ & & & & & & 1 & 1 \\ & & & & & & 0 & 0 \end{pmatrix} \tag{21}$$

Similarly, the matrix $H^{k=1}$ is updated to $H^{k=2}$:

$$H^{k=2} = \begin{pmatrix} 3 & 2 & 0 & 1 \\ & 2 & 0 & 0 \\ & & 0 & 0 \\ & & & 2 \end{pmatrix} \tag{22}$$

As it was previously done, the cluster above the main diagonal with the maximum number of valid orientations is selected. In this case, the selected cluster corresponds to position (i = 1, j = 2) that contains H_{max} = 2 valid orientations. Therefore, new clusters are created (k = 3) so that cluster $C^2_{j=2}$ becomes empty and their points are joined to cluster $C^2_{i=1}$:

$$C^3_1 = C^2_1 \cup C^2_2 = \{P_1, P_3, P_2\},\ C^3_2 = \{\varnothing\},\ C^3_3 = \{\varnothing\} \text{ and } C^3_4 = \{P_4\} \tag{23}$$

The new accessibility matrices for each cluster will be:

$$A^3_1 = CI^2(1,2) = A^2_1 \cap A^2_2,\ A^3_2 = \varnothing,\ A^3_3 = \varnothing,\ A^3_4 = A^2_4 \tag{24}$$

A new update of the matrix CI^k allows for obtaining the matrix $CI^{k=3}(i, j) = A^3_i \cap A^3_j$:

$$CI^{k=3} = \begin{pmatrix} A^3_1 & \varnothing & \varnothing & \varnothing \\ & \varnothing & \varnothing & \varnothing \\ & & \varnothing & \varnothing \\ & & & A^3_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & 0 \\ & & & & & & 1 & 1 \\ & & & & & & 0 & 0 \end{pmatrix} \tag{25}$$

The matrix $H^{k=2}$ becomes $H^{k=3}$:

$$H^{k=3} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 2 \end{pmatrix} \tag{26}$$

Once each one of the elements above the main diagonal of matrices CI^k and H^k have become zero, the process of clustering inspection points finishes. Finally, for this example, the clusters obtained will be:

$$C_1^3 = \{P_1, P_3, P_2\} \text{ and } C_4^3 = \{P_4\} \tag{27}$$

In the example, the maximum number of orientations H_{max} has always been associated to a single cluster. However, in real cases, it may occur that there is more than one cluster with the same number H_{max} of orientations. In this case, the implemented algorithm studies all the possible cases and allows for obtaining different valid combinations of clusters.

[Insert Fig. 13 about here]

7. Application results

In order to analyze the application results of accessibility and clustering algorithms two examples have been included. The STL model for the first part (Fig. 13) contains 2,438 triangles and 38 inspection points have been considered for inspection. These points are shown in Fig. 13.

The accessibility analysis has been divided in several steps:

- Local Accessibility Analysis, where the probe is abstracted as an infinite half-line.

- Global Accessibility Analysis, considering:

  (1) Probe abstracted as a infinite half-line.

  (2) The dimensions of the probe head.

  (3) The dimensions of the probe head and the touch probe.

  (4) The dimensions of the probe head, the touch probe, the stylus and the tip.

  (5) The dimensions of the whole probe including the column.

[Insert Fig. 14 about here]

Fig. 14 shows the accessibility maps for the inspection point P4s39 (point P_4 on surface *s*39) of the first example part considering the different geometrical abstractions of the probe mentioned above. As it can be seen, when actual dimensions and different probe components are taken into account, the accessibility map is substantially reduced. For the rest of inspection points, similar accessibility maps are obtained.

[Insert Table 2 about here]

[Insert Table 3 about here]

Since the accessibility analysis has been performed using algorithms based on kd-trees, the computation time for the analysis is influenced by the number of levels of the tree (depth) and by the number of part triangles. Table 2 shows the time spent computing the global accessibility cones for all the inspection points, taking into account different levels of the tree and different abstractions of the probe components.

The application of the clustering algorithm allows for obtaining the clusters shown in Table 3. In this case a single solution has been found.

[Insert Fig. 15 about here]

A similar analysis may be applied for the second example part shown in Fig. 15. In this case, the number of triangles associated to the STL model is 2,756. Accessibility analysis has been made for 48, 149, 241 and 369 inspection points. Fig. 14 shows the accessibility maps for the inspection point P1s54 (point P_1 on surface s54) considering the aforementioned probe abstractions. Fig. 16 illustrates the total time spent computing valid probe orientations that correspond to the four inspection point sets when kd-tree depth is increased. As it can be seen, the number of points does not have important influence on the selection of the kd-tree depth that minimizes the computation time. Thus the lower computation time is obtained for kd-tree depth between 10 and 12 levels. As regards clustering, three different solutions have been found when considering the 48 inspection point set. Table 4 shows one of them.

[Insert Fig. 16 about here]

[Insert Table 4 about here]

8. Conclusions

The accessibility analysis represents one of the most critical tasks in inspection planning. This analysis determines those probe orientations that can touch an inspection point without collision. Different factors, such as the number of available probe orientations, the complexity of both part and probe components or the number of inspection points, have a direct influence on the accessibility analysis.

Most of accessibility analyses developed only deal with a limited number of probe orientations, simple parts with only planar surfaces or specific geometrical shapes, simplified probe representations or with a short number of inspection points. However,

a new methodology for accessibility analysis is presented in this paper which allows for overcoming the previous limitations:

- All possible orientations (673) of inspection system are taken into consideration.
- The use of STL models permit the developed system to be applied to any type of part, regardless of shape and complexity.
- Real shape and dimensions of the probe are considered for the analysis.
- Implemented algorithms based on Computer Graphics permit to apply the methodology to a higher number of inspection points.
- A clustering algorithm that efficiently groups the inspection points to reduce the number of probe orientation changes is applied.

The accessibility analysis has been carried out in two phases. Local analysis considers only possible collisions with the inspection surface itself whereas global analysis considers the possible collisions with the entire part.

Local analysis at an inspection point has a short computation time because it only involves checking the normal direction of the surface at that point, being the probe abstracted as an infinite half-line. However, the global analysis needs a more complex computation since it includes the calculation of multiple intersections between the probe and all part surfaces. To make this calculation easier, the part has been replaced by its STL model. This way, although the number of part surfaces increases, all of them are planar (triangle facets) and easily recognizable. Also, the STL model permits the application of the developed system to any type of part, regardless of its shape and complexity. The intersection calculation is accelerated by the application of space partitioning techniques as kd-trees. Once the space occupied by the part is partitioned in

regions, recursive ray traversal algorithms are used in order to check intersection only with the part triangles that can potentially be traversed by each probe orientation. To reduce the calculation time further more, techniques for culling back-face triangles have been applied.

For taking into account the real shape of the probe, each of its components has been modelled. In order to speed up the calculation of the intersection between these components and the part triangles, several algorithms and geometrical models have been analysed and described for each component.

With the purpose of reducing the operation time related to probe orientation changes, an algorithm has been implemented for clustering the inspection points with common orientations. This algorithm deals with reducing the number of clusters and maximizing the number of probe orientations within each cluster. When there is more than one cluster combination, the system allows for obtaining each one of them.

The developed system has been applied to different parts with satisfactory results demonstrating its practical application and its possible integration in an inspection planning system. This paper shows the results obtained for two examples taking into account different probe models, number of inspection points and kd-tree depths. For each case the evolution of computation time is shown. Finally, the inspection points have been grouped into clusters.

Future research will concentrate on developing new algorithms that further reduce computation time, on selecting the best probe orientation for each cluster and on generating the final inspection paths.

Acknowledgements

This paper is part of the result of a Spanish State Commission for Science and Technology Research Project (DPI2000-0605) and was supported by this Commission and FEDER.

9. References

Ajmal, A. and Zhang, S.G., The application of a knowledge-based clustering algorithm as an aid probe selection and inspection process planning. *Proceedings of the Institution of Mechanical Engineers, Part B*, 1998, 212, 299-305.

Balasubramaniam, M., Sarma, S.E. and Marciniak, K., Collision-free finishing toolpaths from visibility data. *Computer-Aided Design*, 2003, 35, 359-374.

Ding, S., Mannan, M.A. and Poo, A.N., Oriented bounding box and octree based global interference detection in 5-axis machining of free-form surfaces. *Computer-Aided Design*, 2004, 36, 1281-1294.

Eberly, D.H., 3D Game engine design. A practical approach to real-time computer graphics, 2000 (Morgan Kaufmann Publishers: San Diego).

Foley, J.D., van Dam, A., Feiner, S.K. and Hughes, J.F., Computer Graphics, Principles and Practice, 1997 (Addison-Wesley: Cornell, USA), pp. 663-664.

Fuchs, H., Kedem, Z.M. and Naylor, B.F., On visible surface generation by a priori tree structures. *Proceedings of the SIGGRAPH-80*, 1980, 14, pp. 124-133.

Gottschalk, S., Lin, M.C. and Manocha, D., OBBTree: a hierarchical structure for rapid interference detection. *Proceedings of the SIGGRAPH-96*, 1996, pp. 171-180

Havran, V., Heuristic ray shooting algorithms. PhD thesis, Czech Technical University, 2000

Held, M., ERIT-A collection of efficient and reliable intersection tests. *Journal of Graphics Tools*, 1997, 2, 25-44

Ho, S., Sarma, S. and Adachi, Y., Real-time interference analysis between a tool and an environment. *Computer-Aided Design*, 2001, 33, 935-947.

Hwang, C.-Y., Tsai, C.-Y. and Chang, C.A., Efficient inspection planning for coordinate measuring machines. *International Journal of Advanced Manufacturing Technology*, 2004, 23, 732-742.

Ilushin, O., Elber, G., Halperin, D., Wein, R. and Kim, M.-S., Precise global collision detection in multi-axis NC-machining. *Computer-Aided Design*, 2005, 37, 909-920.

Jackman, J. and Park, D.-K., Probe orientation for coordinate measuring machine systems using design models. *Robotics and Computer-Integrated Manufacturing*, 1998, 14, 229-236.

Jansen, F.W., Data structures for raster graphics, pp. 57-73, 1986 (Springer-Verlag: New York)

Kaplan, M., Space-tracing: A constant time ray-tracer. *Proceedings of the SIGGRAPH-85*, 1985, 19, 149-158.

Klosowski, J.T., Held, M., Mitchell, J.S.B., Sowizral, H. and Zikan, K., Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 1998, 4, 21-36.

Kweon, S. and Medeiros, D.J., Part orientation for CMM inspection using dimensioned visibility maps. *Computer-Aided Design*, 1998, 30, 741-749.

Lim, C.P. and Menq, C.H., CMM feature accessibility and path generation. *International Journal of Production Research*, 1994, 32, 597-618.

Limaiem, A. and ElMaraghy, H.A., A general method for analysing the accessibility of features using concentric spherical shells. *International Journal of Advanced Manufacturing Technology*, 1997, 13, 101-108.

Limaiem, A. and ElMaraghy, H.A., CATIP: A Computer-Aided tactile inspection planning system. *International Journal of Production Research*, 1999, 37, 447-465.

Lin, Z.-C. and Chen, C.-C., Collision-free path planning for coordinate measurement machine probe. *International Journal of Production Research*, 2001, 39, 1969-1992.

Möller, T.A., Fast 3D Triangle-Box Overlap Testing. *Journal of Graphics Tools*, 2001, 6, 29-33.

Möller, T.A. and Trumbore, B., Fast, minimum storage ray/triangle intersection. *Journal of graphics tools*, 1997, 2, 21-28.

Schneider, P.J. and Eberly, D.H., Geometrical Tools for computer graphics, 2003 (Morgan Kaufmann Publishers)

Spitz, S.N., Spyridi, A.J. and Requicha. A.A.G., Accessibility analysis for planning of dimensional inspection with coordinate measuring machines. *IEEE Transactions on Robotics and Automation*, 1999, 15, 714-727.

Spitz, S.N. and Requicha, A.A.G., Accessibility analysis using computer graphics hardware. *IEEE Transactions on Visualization and Computer Graphics*, 2000, 6, 208-219.

Spyridi, A.J. and Requicha, A.A.G., Accessibility analysis for the automatic inspection of mechanical parts by coordinate measuring machines. *Proceedings of the IEEE International Conference on Robotics and Automation*, 1990, pp. 1284-1289.

Vafaeesefat, A. and ElMaraghy, H.A., Automated accessibility analysis and measurement clustering for CMMs. *International Journal of Production Research*, 2000, 38, 2215-2231.

Vafaeesefat, A. and ElMaraghy, H.A., Optimal workpiece orientations for machining of sculptured surfaces. *Proceedings of the Institution of Mechanical Engineers, Part B*, 2000, 214, 671-681.

Wu, Y., Liu, S. and Zhang, G., Improvement of coordinate measuring machine probing accessibility. *Precision Engineering*, 2004, 28, 89-94.

Yin, Z.-P., Ding, H. and Xiong, Y.-L., Visibility theory and algorithms with application to manufacturing processes. *International Journal of Production Research*, 2000, 38, 2891-2909.

Zhang, S.G., Ajmal, A., Wootton J. and Chisholm, A., A feature-based inspection process planning system for coordinate measuring machine (CMM). *Journal of Materials Processing Technology*, 2000, 107, 111-118.

Ziemian, C.W. and Medeiros, D.J., Automated feature accessibility algorithm for inspection on a coordinate measuring machine. *International Journal of Production Research*, 1997, 35, 2839-2856.

Ziemian, C.W. and Medeiros, D.J., Automatic probe selection and part setup planning for inspection on a coordinate measuring machine. *International Journal of Computer Integrated Manufacturing*, 1998, 11, 448-460.

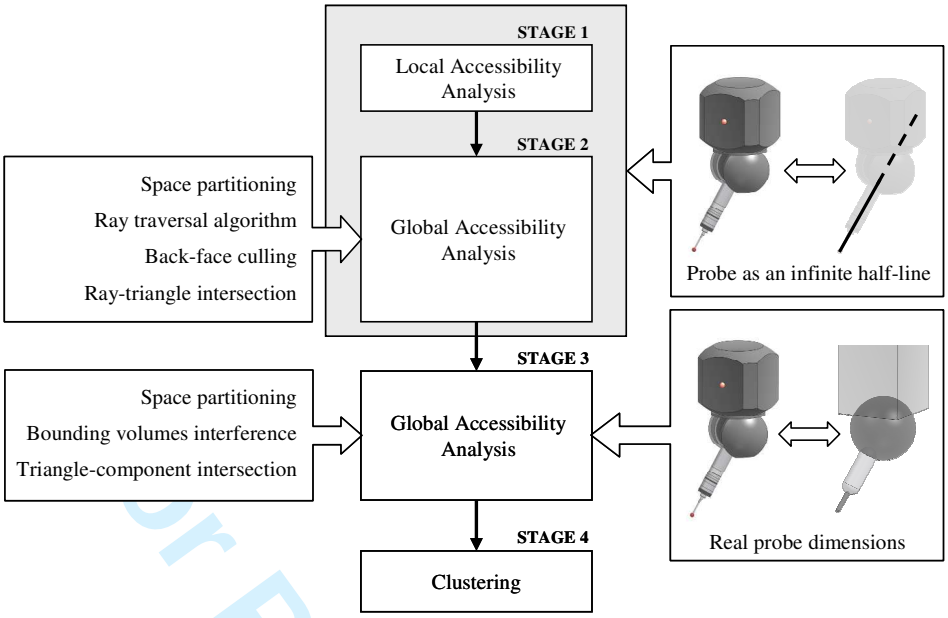


Figure 1

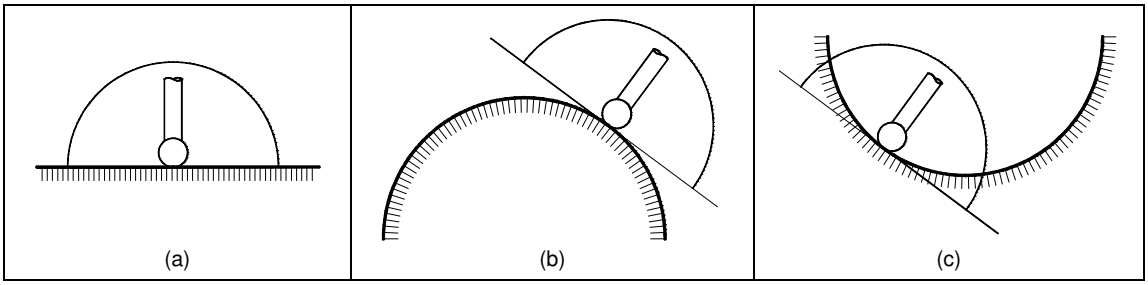
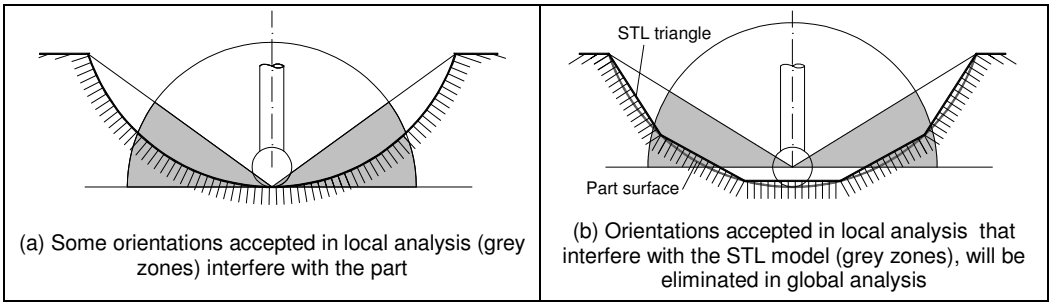


Figure 2



(a) Some orientations accepted in local analysis (grey zones) interfere with the part

(b) Orientations accepted in local analysis that interfere with the STL model (grey zones), will be eliminated in global analysis

Figure 3

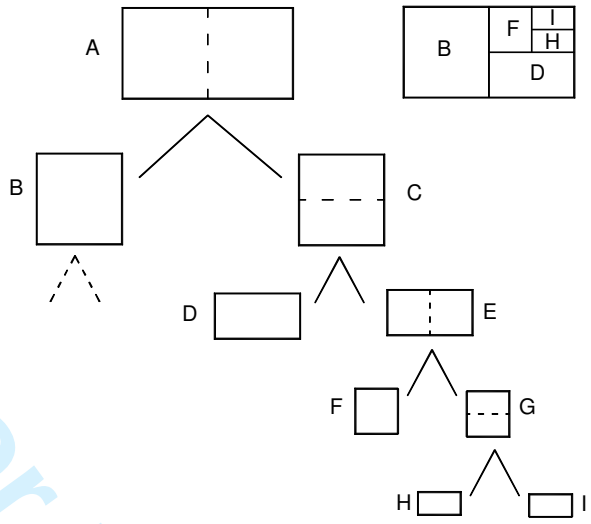


Figure 4

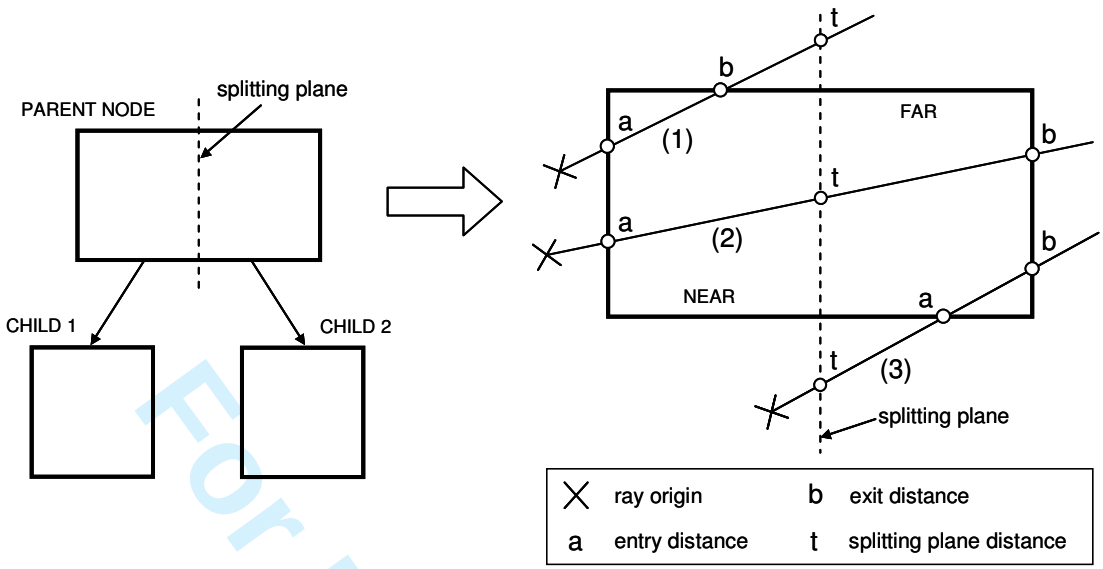


Figure 5

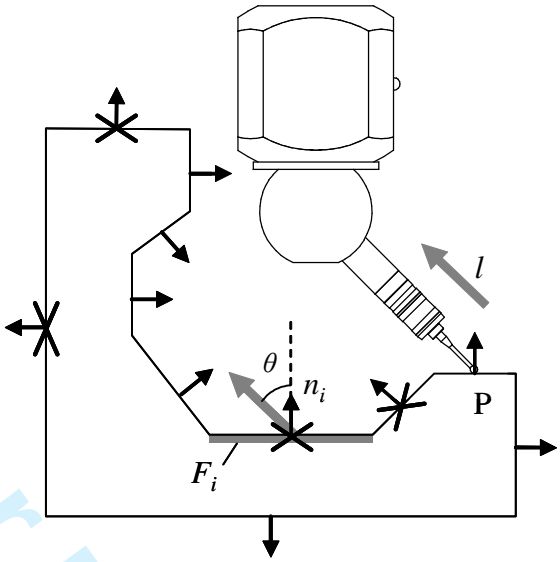


Figure 6

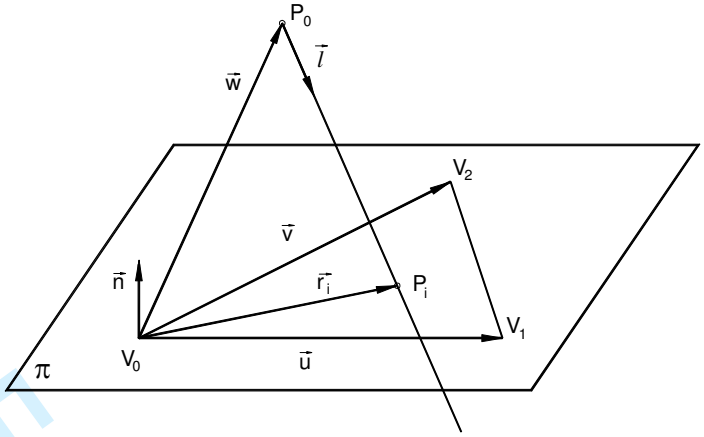


Figure 7

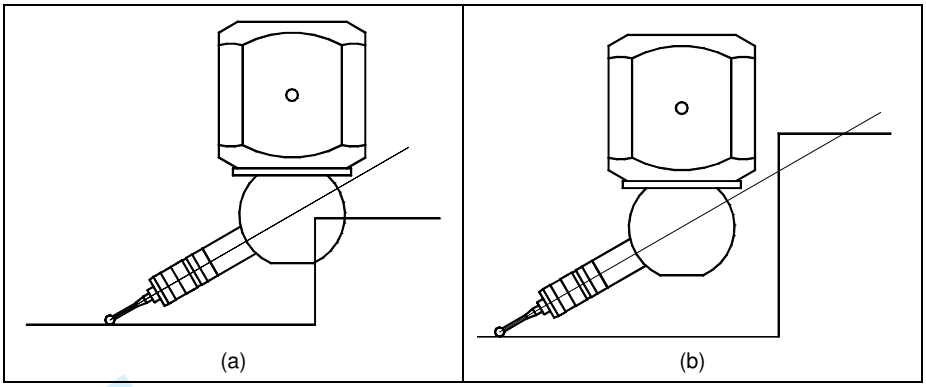


Figure 8

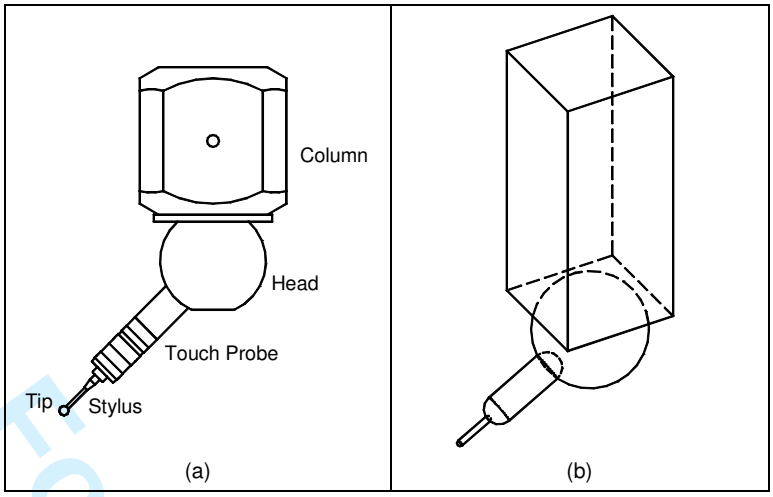


Figure 9

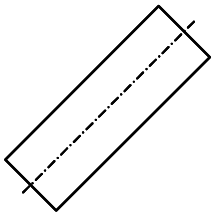
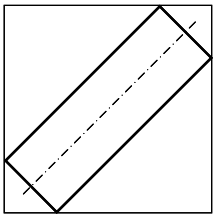
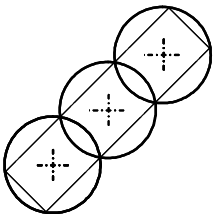
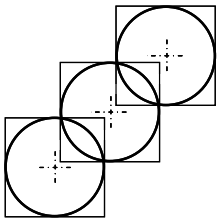
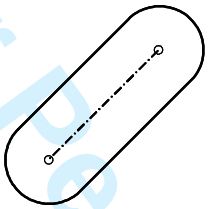
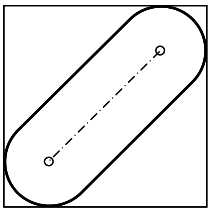
MODEL		BOUNDING BOX	
Cylinder		Axis-aligned bounding box for a cylinder	
Spheres		Axis-aligned bounding boxes for spheres	
Capsule		Axis-aligned bounding box for a capsule	

Figure 10

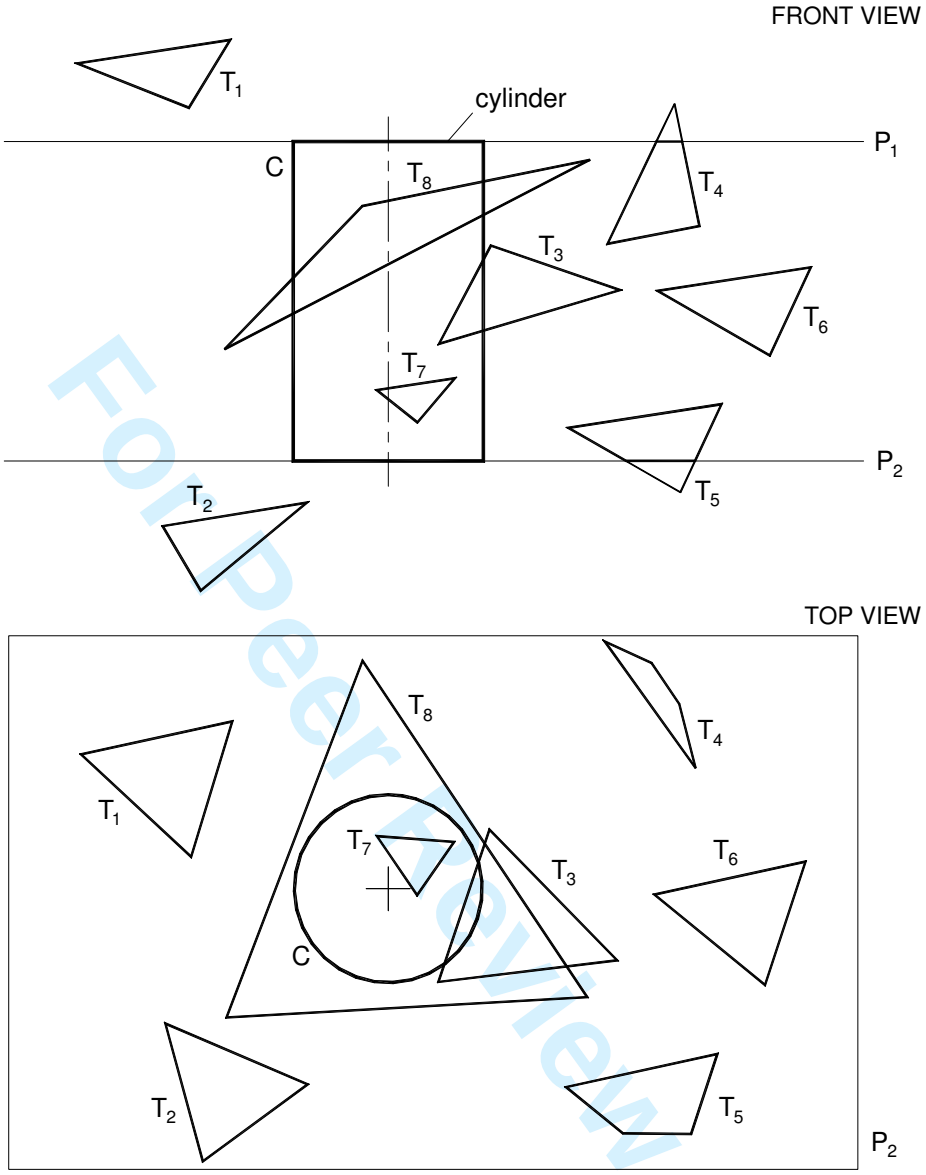


Figure 11

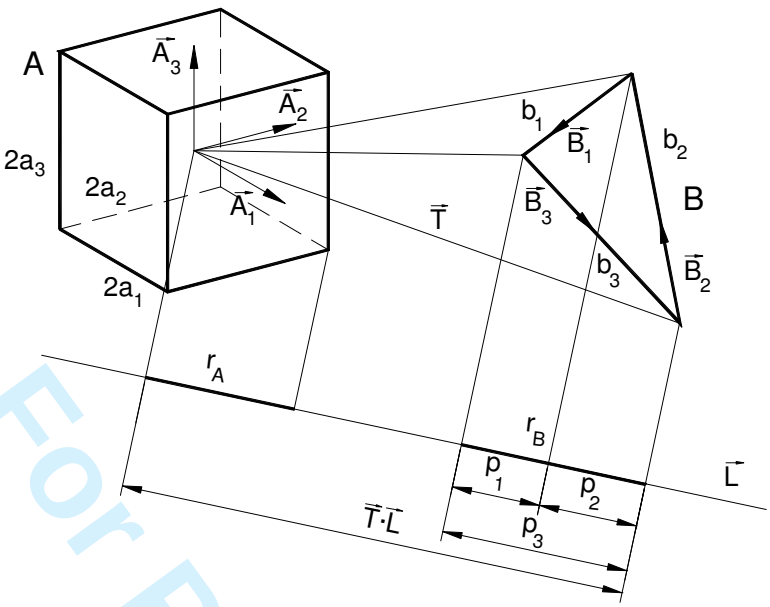


Figure 12

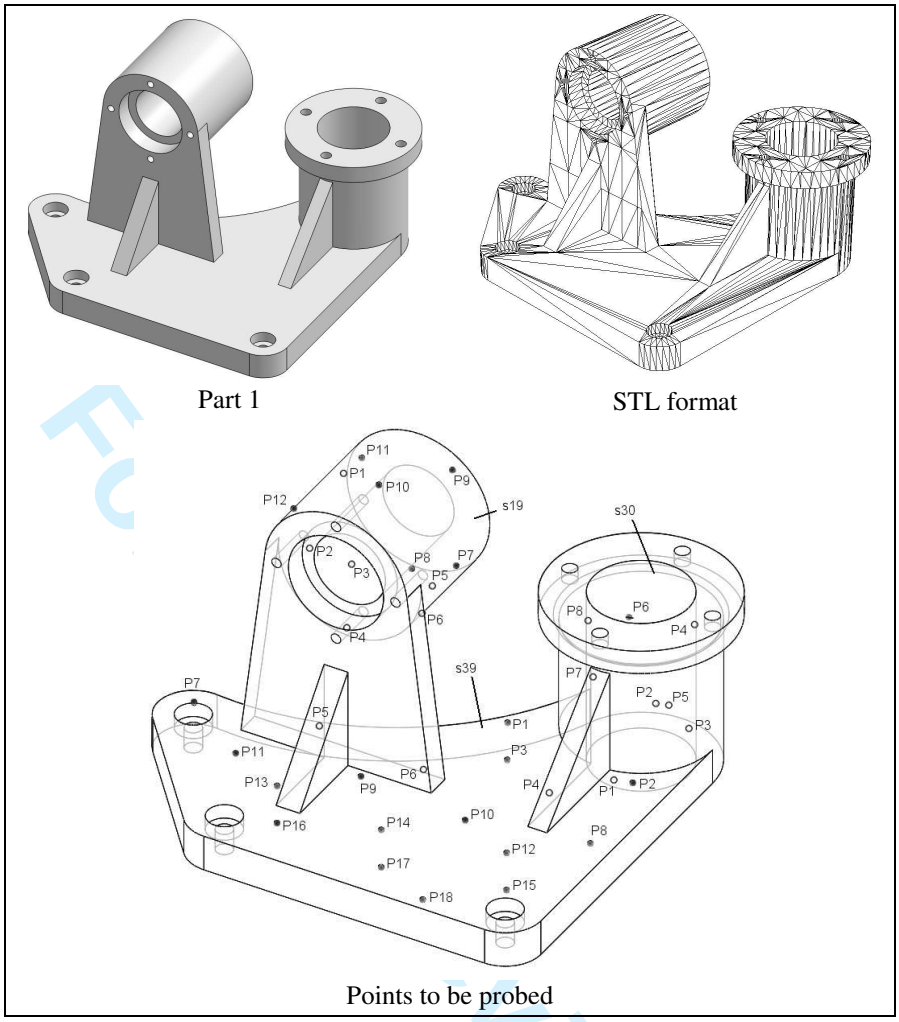

Part 1
STL format
P11
P1
P10
P9
P12
s19
s30
P2
P3
P8
P7
P5
P6
P4
P8
P6
P4
s39
P7
P7
P2
P5
P5
P1
P3
P11
P3
P13
P6
P9
P4
P1
P2
P16
P14
P10
P8
P12
P17
P15
P18
Points to be probed

Figure 13

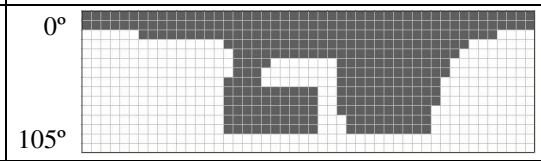
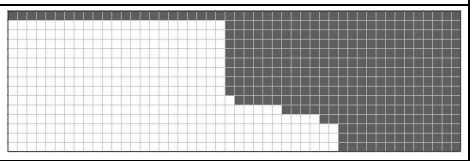
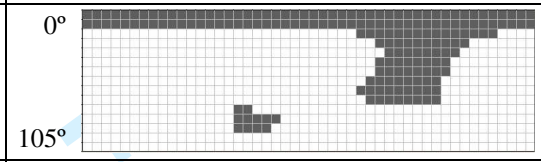
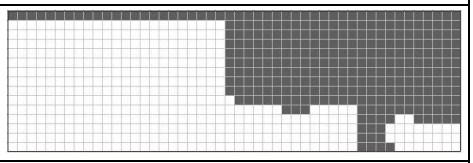
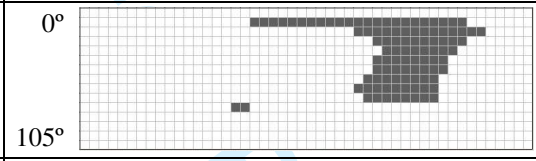
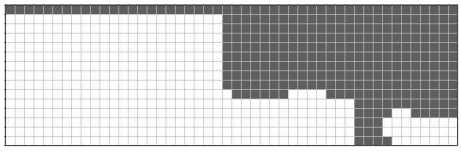
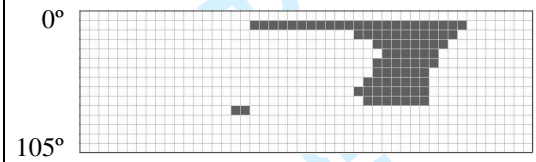
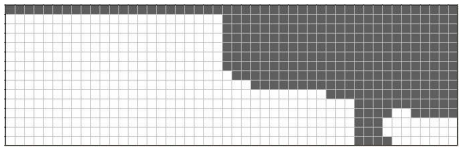
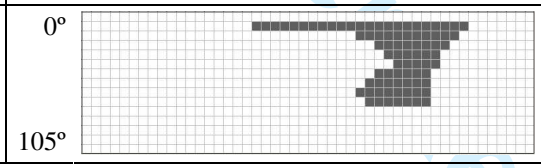
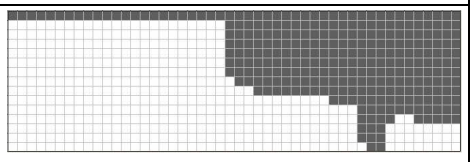
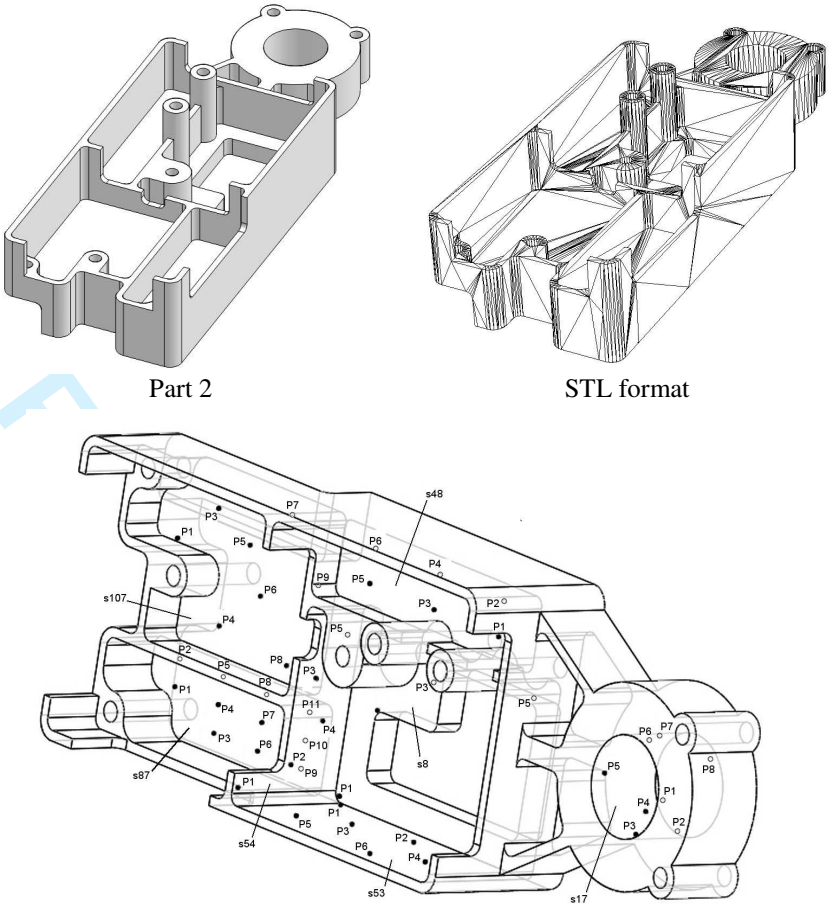
	Valid orientations	
	Part 1	Part 2
	-172.5º 0º 180º	-172.5º 0º 180º
Probe abstracted by a line segment		
Head		
Touch probe		
Stylus and Tip		
Column		

Figure 14



Points to be probed

Figure 15

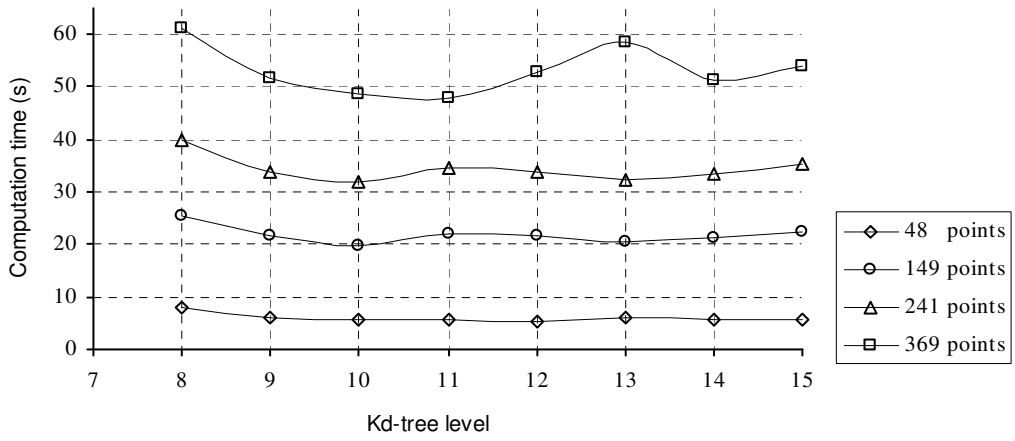


Figure 16

Table 1. Number of intersection tests for different levels of the kd-tree

Levels of the kd-tree	Facets in kd-tree leaves	Intersection tests
4	712	506023
5	860	402500
6	1111	301650
7	1498	175261
8	1965	159620
9	2691	157437
10	3587	120880
11	4787	123696
12	6897	138972
13	10083	149950
14	13893	179482
15	20783	238113
16	31236	301500
17	44025	396320
18	68020	581084

Table 2. Time spent computing the global accessibility cones for all the inspection points of part 1

Levels of the kd-tree	Probe abstracted by a line segment (s)	Real probe				Total (s)
		Head (s)	Touch probe (s)	Stylus and Tip (s)	Column (s)	
8	1.971	2.756	2.112	2.139	0.926	9.904
9	1.319	2.223	1.089	1.145	0.647	6.423
10	1.083	2.165	0.825	0.841	0.476	5.390
11	0.967	1.979	0.677	0.721	0.437	4.781
12	0.959	1.941	0.621	0.608	0.409	4.538
13	0.940	2.067	0.584	0.565	0.366	4.522
14	0.949	2.065	0.593	0.593	0.341	4.541
15	0.981	2.352	0.616	0.592	0.345	4.886

Table 3. Result of the clustering process for part 1

Cluster	Point	Orientation (A, B)	
		A (º)	B (º)
1	P1s19, P2s19, P12s19, P4s7, P4s39, P7s39, P11s39, P13s39	37.5, 45	67.5
2	P3s19, P1s39, P5s39, P6s39	67.5	- 60
3	P4s19, P5s19	97.5	- 90, - 82.5, - 75, - 67.5, - 60
4	P6s19, P8s7	30	- 165, 180
		37.5	- 172.5, -165, - 157.5, - 150, - 142.5, - 135, 157.5, 165, 172.5, 180
		45	- 172.5, - 165, - 157.5, - 150, - 142.5, - 135, - 127.5, - 120, 180
		52.5	- 172.5, - 157.5, - 150, - 142.5, - 135, - 127.5, - 120, - 112.5
5	P7s19, P8s19, P9s19, P10s19, P11s19, P6s7, P2s39, P3s39, P8s39, P9s39, P10s39, P12s39, P14s39, P15s39, P16s39, P17s39, P18s39	15	120, 127.5
		22.5	112.5, 120, 127.5, 135, 142.5, 150, 165
		30	112.5, 120, 127.5, 135, 142.5, 150, 165
6	P2s7, P3s7	7.5	-15
7	P7s7	7.5	-165, -157.5, -150, 165, 172.5, 180
		15	-172.5, -165, -157.5, -150, -142.5, -135, 150, 157.5, 165, 172.5, 180
		22.5	-172.5, -165, -157.5, 172.5, 180
		30	-172.5

Table 4. Result of the clustering process for part 2 and 48 inspection points

		Orientation (A, B)				
Cluster	Points	A(º)	B(º)			
1	P1s8, P4s17, P5s17, P1s53, P1s54, P2s54, P3s54, P4s54, P1s87, P1s107	15	135	142,5	150	157,5
		22,5	135	142,5		
2	P2s8, P1s17, P2s17, P3s17, P1s48, P3s48, P5s48, P2s53, P3s53, P4s53, P5s53, P6s53, P3s87, P4s87, P6s87, P7s87, P9s87, P10s87, P3s107, P4s107, P5s107, P6s107, P8s107, P9s107	7,5	-165	-157,5	-150	-142,5
		15	-120			
3	P3s8, P6s17, P5s54, P2s87, P5s87	7,5	37,5	45	52,5	
4	P5s8, P7s17, P8s17, P2s48, P4s48, P6s48, P8s87, P11s87, P7s107	7,5	-52,5	-45	-37,5	-30
			-22,5	-15		
		15	-52,5	-45	-37,5	-30
			-22,5	-15		
		22,5	-45	-37,5	-30	-22,5
			-15			
		30	-30	-22,5	-15	