Modeling Sensor Detectability with VANTAGE Geometric/Sensor Modeler

Katsushi Ikeuchi and Jean-Christophe Robert

February, 1989

CMU-CS-89-120

This research was sponsored by the Defense Advanced Research Projects Agency, DOD, through ARPA Order No. 4976, and monitored by the Air Force Avionics Laboratory under contract F33615-87-C-1499.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or of the U.S. Government.

ABSTRACT

This paper describes a method of modeling sensor detectabilty and its implementation in the VANTAGE geometric/sensor modeler.

Each sensor consists of one or more light sources and TV cameras, which illuminate and observe, respectively, various portions of faces of an object. Sensor detectability, which specifies where the sensor can "see" (detect), can be described by a result of AND/OR operations between illuminated and camera-observable portions of 3D faces.

In order to represent this sensor detectability, first we will define a G-source, an abstract sensor component, representing either a light source or a TV camera. Then, we will investigate G-source illumination conditions to describe under what condition each G-source illuminates (or observes) surface portions with respect to its illumination direction. We will show that the sensor detectability can be decomposed into its component G-sources' illumination conditions and AND/OR operations between them. Based on this findings, we will propose the sensor composition tree. The tree consists of G-source illumination conditions as its leaf nodes and set operations which indicate a combination of the component G-source illumination conditions, as its branch nodes. We will also propose several 2D structures to represent detected 2D appearances given by VANTAGE using a sensor composition tree. Finally, we will show how to use these capabilities in model-based vision.

Contents

1 Introduction

Geometric modelers allow users to create, store, and manipulate models of three-dimensional (3D) solid objects [17,18,16]. These geometric modelers have found many applications in CAD/CAM and robotics areas. Such examples include mechanical part designs [22,6], off-line robot simulations [15,21], automatic creation of programs for numerically controlled (NC) machineries [4], finite element analysis.

Building a model-based vision system based on a geometric modeler [5,7,8] is one of the most interesting applications. The relevant knowledge of an object required for recognition is extracted from its representation in a geometric modeler and then used for recognition by a vision program. It has become very common to design an object by means of a geometric modeler; this designing process provides a geometric representation of an object. This geometric representation can be used to build a model-based vision system. If we can establish a systematic method of converting those representations into a recognition program of that object, recognition programs can be relatively easily obtained.

A geometric modeler represents a 3D object, while a recognition program observes a 2D appearance of the object, and thus requires its 2D representation. Moreover, model-based vision systems often use various active sensors to obtain visual information: an active sensor projects a special set of lights onto an object and measures depth/surface orientation from the returned light. Under these active sensors, the object appearances are determined by the *product* of an object model with a sensor detectability, which defines where the sensor can "see" (detect) [10]. As shown in Figure 1, the same object under the same attitude, thus represented as the same object model, can create different appearances (and features) when detected by different sensors. The edge-based binocular stereo reliably detects depth at edges perpendicular to the epipolar lines, which are the lines parallel to the line connecting its two TV cameras. The photometric stereo, which consists of three light sources and one TV camera, detects surface orientations of the surfaces which are illuminated by the three light sources and are visible to the TV camera. The light-stripe range finder, which consists of one light source and one TV camera, detects distances of surfaces which are illuminated by the light source and visible to the camera.

Thus, it is necessary for a geometric modeler to represent not only an object model but also a sensor detectability in order for it to be fully used for a model-based vision system. It is also important to represent the 2D appearance of an object explicitly and symbolically, because such a 2D representation is the one to be used by the vision system, and thus should be well-organized and easily accessible. Surprisingly, however, little effort has been expended in this direction, even though some of the early applications of the geometric modeler were vision applications [20,2]. This is probably because the main application of the geometric modeler is still in designing mechanical objects, and the main concern is how to represent 3D rather than 2D information.

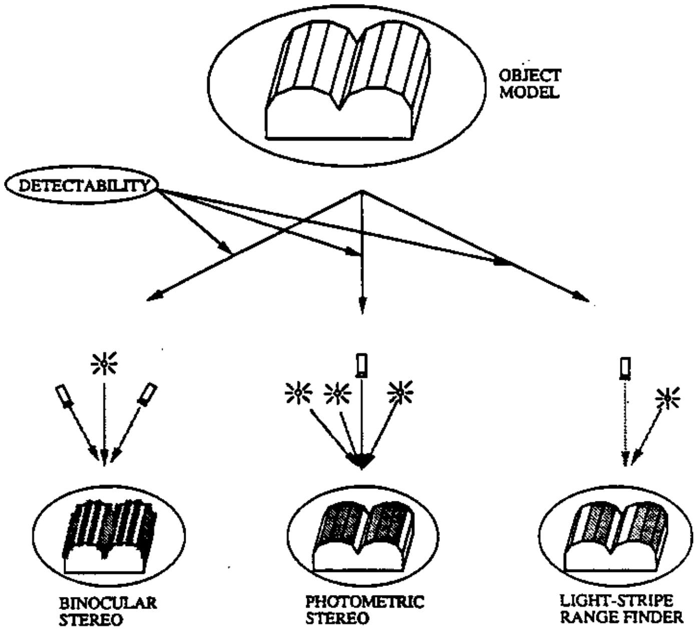


Figure 1: Object Appearances.

This paper proposes to represent sensor detectabilty and to produce symbolic 2D representations of an object appearance using a geometric modeler. First, we examine how to specify the sensor detectabilities. We propose to represent them using a tool called *a sensor composition tree* and consider how tc implement this sensor composition tree in our geometric modeler, VANTAGE. Then we will propose several 2D structures to represent an object appearance symbolically. We will also discuss some of the applications of this system.

2 Definition of Sensor Composition Tree

Each sensor consists of one or more light sources and TV cameras, which illuminate and observe, respectively, various portions of 3D faces. Sensor detectability, which specifies where the sensor can detect, can be described as a result of AND/OR operations among illuminated and camera-observable portions of 3D faces.

In order to represent this sensor detectability, first this section defines a G-source as an abstract sensor component representing either a light source or a TV camera, and then, investigates G-source illumination conditions to describe under what condition each G-source illuminates (or observes) surface portions with respect to its illumination direction. This section also proposes a sensor composition tree, which consists of G-source illumination conditions as its leaf nodes and set operations which indicates the combination of the component G-source illumination conditions, as its branch nodes.

2.1 Feature Configuration Space

Sensor detectability depends upon various factors which include: position of a feature, orientation of a feature, reflectivity of a feature, transparency of air, ambient lighting, and so forth. In most object recognition problems the orientation and position of a feature are the most important factors. This subsection proposes a method of representing the angular freedom between a feature and a sensor, which affects sensor detectability the greatest. Later on, we will consider the effect of distance.

In order to specify the angular freedom explicitly, we attach a coordinate system to each point of an object feature and consider the relationship between the sensor coordinate system and the feature coordinate system. For example, on a 3D face, we define a coordinate system so that the z axis of the feature coordinate system agrees with the surface normal at that point and the x-y axes lie on the face, but are defined arbitrarily otherwise. For other features, we can also define a feature coordinate system appropriately. See Appendix A for more details.

Since angular relationships between the two coordinate systems are relative, for the sake of convenience we fix the sensor coordinate system and discuss how to specify feature coordinates with respect to it. The angular relation betweem the sensor coordinate system and feature coordinate system can be specified by three degrees of freedom: two degrees of freedom in the direction of the z-axis of the feature coordinate system, and one degree of freedom in the rotation about the z-axis. See Figure 2 (a).

Since we want to consider the angular relationship, we can translate the feature coordinate system so that the two coordinate systems share the origin. We will then define a sphere whose origin is the origin of the sensor coordinate system, and whose north pole is the z axis of that system. We will specify a feature coordinate system as a point on the sphere. Referring to Figure 2 (b), the point on the north pole of the sphere represents the feature coordinate system aligned completely with the sensor coordinate system. Any other point on the spherical surface represents a feature coordinate system obtained by rotating the sensor coordinate system around the axis perpendicular to a plane given by the sphere center, the spherical point, and the north pole until the direction from the sphere center to the point coincides with the z axis of the feature coordinate system. Figure 2 (c) represents various sensor coordinate systems corresponding to spherical points. As for a point inside of the sphere, the distance from the spherical surface to the point represents the angle of rotation (modulo 360°) around the z axis from the coordinate system corresponding to the surface point to the coordinate system corresponding to the inside point. Figure 2 (d) shows those coordinate systems corresponding to points on a radial axis. We will refer to this sphere as the feature configuration space, and represent the ability of a given sensor to detect in it.[1],[2]

[1] This representation will not create discontinuities around the north pole as opposed to the case in which Euler angles from the sensor coordinate system to the feature coordinate system are used to specify spherical points; this representation will instead create discontinuities at the center of the sphere and at the south pole. However, this is advantageous because we mainly use the area around the north pole to discuss detectability and reliability.

[2] Most sensors detect faces. In this case, we only need to consider the spherical surface as the configuration space instead of the total sphere.

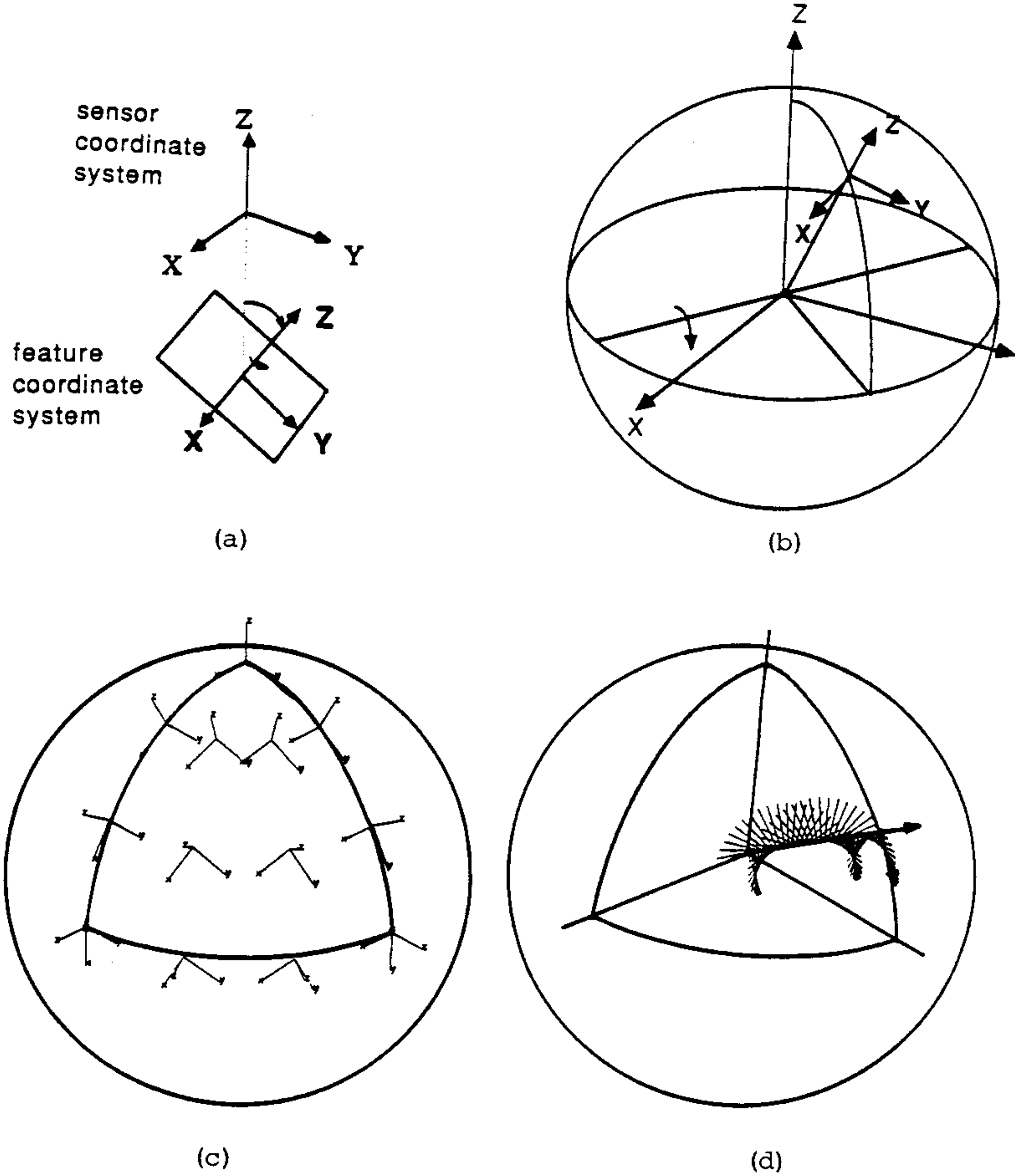


Figure 2: Feature Configuration Space.

2.2 G-source Illumination Condition

Each sensor consists of two kinds of components: light sources and TV cameras. For example, both a time-of-flight range finder and a light-stripe range finder have one light source and one TV camera. The binocular stereo has one light source (without light sources you cannot observe anything) and two TV cameras; the photometric stereo has three light sources and one TV camera.

This paper regards both light sources and TV cameras as generalized sources (G-sources). Each G-source has two properties: the illumination direction and the illuminated configurations. The G-source illumination direction of a light source denotes its light source direction; the G-source illumination direction of a TV camera denotes the direction of its line of sight. G-source illuminated configurations of a light source denote the collection of the feature coordinate systems in which the feature can be illuminated, provided that its illumination direction is not occluded; G-source illuminated configurations of a TV camera denote the collection of those systems in which the feature is visible to the TV camera, provided that its illumination direction is not occluded.

Thus, in order for a feature to be illuminated by a G-source, the feature coordinate system should be in the illuminated configurations, and the G-source illumination direction should not be occluded. We will call these two conditions the *illumination condition* of the G-source.

The G-source illumination direction can be represented in the feature configuration space by a radial line from the sphere center. Let us denote this G-source illumination direction as V, which is a unit vector from the origin of the feature coordinate system to the sensor coordinate system. G-source illuminated configurations can be specified as a volume in the configuration space. For example, let us suppose a light source is located at the origin of the sensor coordinate system and the origin of a feature coordinate system is on the negative z direction of the sensor coordinate system. Then, its G-source illumination direction is represented as the line segment from the sphere center to the north pole. If this G-source illuminates faces whose surface orientation is less than 90 degrees from the illumination direction, then its G-source illuminated configurations are represented by the northern hemisphere of the feature configuration space, as shown in Figure 3. In most cases, however, the area near the equator is noisy, so we exclude that area and have a spherical cone whose axis is the same as V

and whose apex angle is *d*.[3]

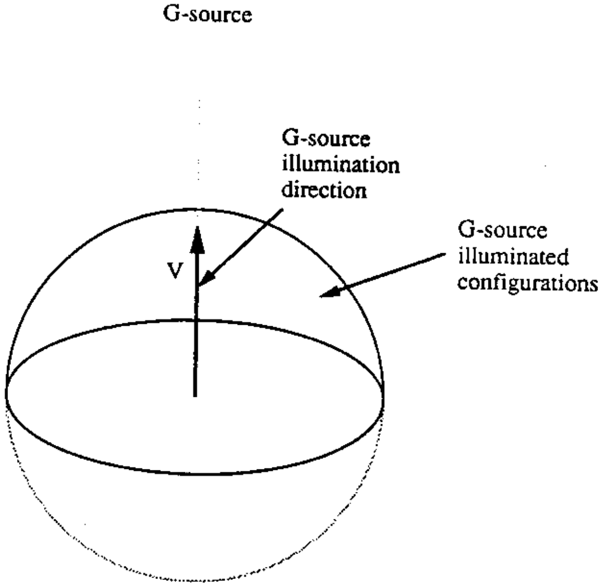

G-source
G-source illumination direction
G-source illuminated configurations
V

Figure 3: G-source illumination condition.

Once a G-source illumination condition is applied to a 3D face, it generates an illuminated portion or portions and a shadowed portion or portions on the 3D face.

2.3 Sensor Composition Tree

The detectability of a sensor, that is where a sensor can detect, can be *decomposed* into illumination conditions of its component G-sources and operations between them. For example, a photometric stereo system can detect only the portions of a 3D face on which its three light sources project light directly and which its TV camera observes. Thus, we can decompose the detectability of a photometric stereo system into four different G-source illumination conditions (three light sources and one TV camera), and *AND* operations between them. In

[3] The current implementation of VANTAGE can handle only the normal G-source, whose illuminated configurations are depicted as a spherical cone, describe above. For other types of G-sources, see [10].

the case of a light-stripe range finder, for a portion to be seen, it is necessary that it be illuminated by the light source and be observed from its TV camera. Thus, we can decompose the detectability into two G-source illumination conditions and *AND* operations between them.

Accordingly, portions of a 3D face detected by a sensor can be obtained by *independently* applying its component G-source illumination conditions to the 3D face and then applying set operations to the illuminated portions on the 3D face. For example, the photometric stereo's detected portions can be obtained by applying its four G-source illumination conditions to a 3D face to generate four kinds of illuminated portions on the 3D face, and then applying *AND* operations among the illuminated portions. For a light-stripe range finder, its detected portions can be obtained by applying two G-source illumination conditions to a 3D face to generate two kinds of illuminated portions on it and then applying an AND operation between the portions on the 3D face.

A TV camera has functions as G-source and as a projector. As a G-source, a TV camera generates illuminated (visible) portions on a 3D face. In other words, by tracking the lines of sight in reverse directions, we can define the illuminated (visible) portions on the 3D face. (Actually, those portions are visible from the TV camera.) Then, the TV camera projects the illuminated (visible) portions of a 3D face onto the image plane to generate its 2D appearance.

Based on this consideration, *AND, OR*, and *PROJECTION* operations are necessary for combining illuminated portions on a 3D face and for projecting, to generate a 2D appearance of the portions detected by a sensor. It is also necessary to specify the order of the set operations. For this, we will define a sensor composition tree. That is a binary tree, each of whose leaf nodes represents a G-source illumination condition and each of whose branch node represents one of the set operations: *AND,OR*, or *PROJECTION*.

We can represent several sensors' detectability using this sensor composition tree. For example, Figure 4(a) shows a light-stripe range finder and its sensor composition tree. The right leaf node, *GS1*, corresponds to the illumination condition of the light source, and the center node, *GS2*, represents the illumination condition of the TV camera. These two nodes are connected by the operation, *AND*. Since the appearance is generated with respect to the coordinate of the TV camera, the operation, *PROJECTION*, is applied with respect to *GS2*.

Two other examples, a two-light light-stripe range finder and a photometric stereo, are also shown in Figure 4.

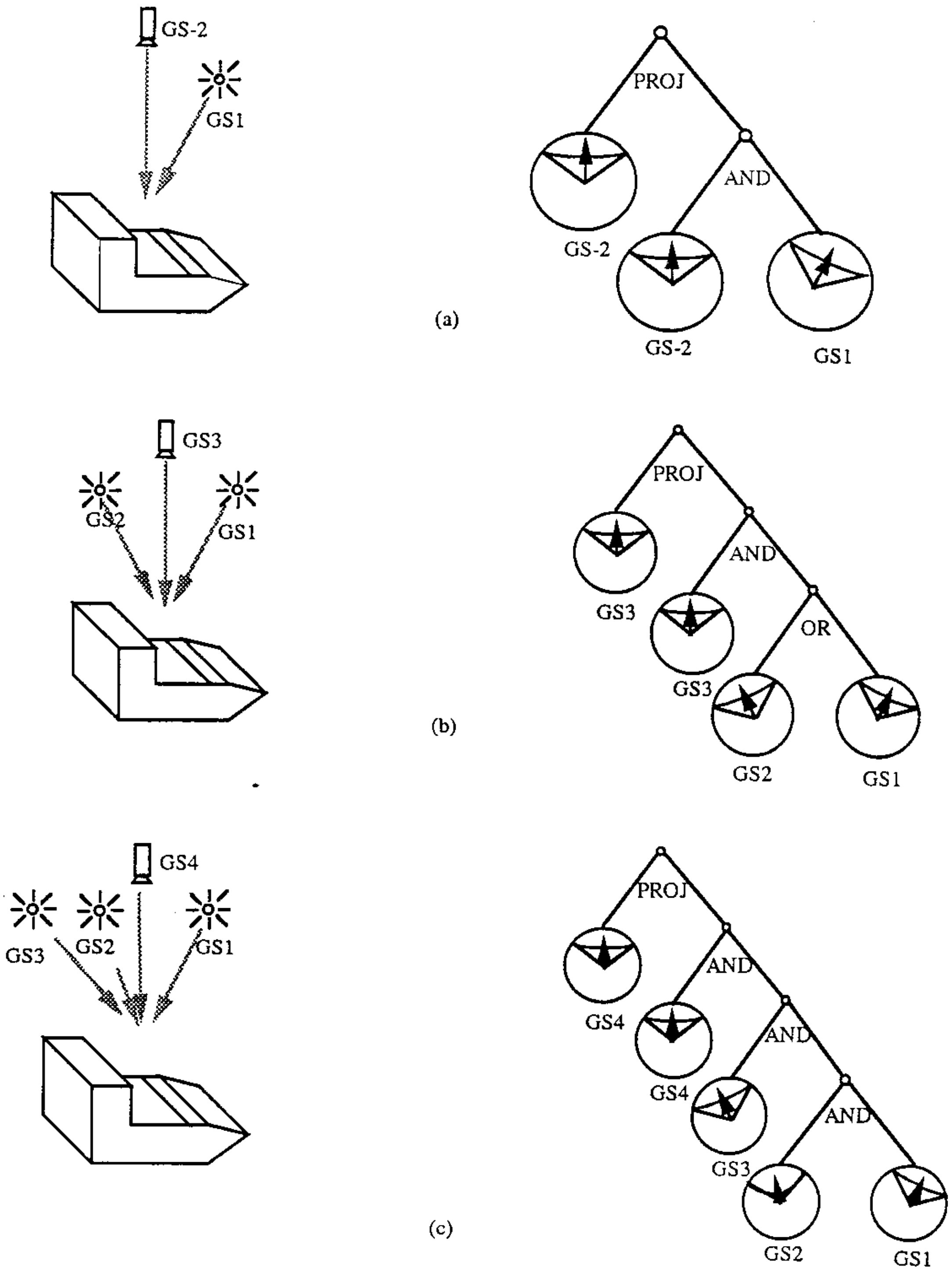


Figure 4: Example of sensor composition tree: (a) The sensor composition tree of a light stripe range finder; (b) The sensor composition tree of a two-light stripe range finder; (c) The sensor composition tree of a photometric stereo sensor.

3 Sensor Composition Tree in VANTAGE

This section considers a method of representing the sensor composition tree and the operations required to determine where a sensor can detect in VANTAGE.

3.1 G-source and Its Operations

This subsection defines a method of representing a G-source and its illumination condition in VANTAGE. VANTAGE represents a G-source illumination condition as a node of the sensor composition tree. A VANTAGE function, SCTNODE, creates a node of a sensor composition tree in VANTAGE.

The G-source illumination direction, and thus also the illuminated configurations depend on the position of the feature coordinate system with respect to the the sensor coordinate system. However, if we can assume that a G-source is far away from the object, then the G-source illumination direction becomes a constant vectors at any point of an object considered. In this case, that direction can be specified as a constant vector, $\mathbf{V} = (X_v, Y_v, Z_v)^t$ with respect to the sensor coordinate system at any point of the object.

The G-source illuminated configurations are also represented as an invariant spherical cone regardless of the position of the feature. The axis of the cone is the same as the illumination direction, V, while the apex angle is denoted as d. Using these parameters, the illumination condition of the distant G-source, called the *Orthographic Generalized Source*, can be represented as

```
(SCTNODE OGS g-source-name V d)
```

in VANTAGE, where `g-source-name` is a node name used in the sensor composition tree.[4]

In general, the G-source illumination direction depends on the position of a feature. As a result, the G-source illuminated configurations change over the object. However, the shape of the spherical cone, whose apex angle is d, is invariant with respect to the illumination direction. Thus, in this case, called the *Perspective Generalized Source*, we will specify the explicit position of the G-source.[5]

[4] The current implementation of VANTAGE has also optional arguments to indicate *type*, *focal length*, and *image size*.

[5] More precise specification is possible such as defining focal length. See [1] for more details.

```
(SCTNODE PGS g-source-name X d),
```

where X denotes a matrix that indicates the coordinate system of the G-source with respect to the sensor coordinate system.

VANTAGE calculates the illumination direction based on the position of the feature with respect to the sensor and determines the illuminated portions of a face. These illuminated portions of the face is attached to the *3D boundary representation* of the 3D face as 3D face illumination properties.

Figure 5(a) shows an example of a 3D scene, where a 3D scene refers to a collection of several 3D objects. In this example, it consists of one cube, one arch, and one table. To this 3D scene, we will apply one sensor which consists of two G-sources: one light source from the right direction and one TV camera from the left direction. We will use this 3D scene as an illustrative example throughout this paper.

One G-source, corresponding to the light source, generates 3D illumination properties on its 3D faces, as shown in Figure 5(b), while another G-source, corresponding to the TV camera, generates other 3D illumination properties on them, as shown in Figure 5(c). Since a property does not necessarily take a uniform value over a single 3D face (for example, only one portion of a face may be shadowed while other portions may be illuminated by G-source 1), VANTAGE divides the 3D face according to the different G-sources, and creates corresponding 3D subfaces.

3.2 Implementation of Sensor Composition Tree

It is necessary to consider set operations on 3D illumination properties on a 3D face. We will define an *AND* operation between an illuminated property I and a shadowed property S by two different G-source, i, j as follows:

$$\begin{aligned} I &\leftarrow (AND\ I_i\ I_j) \\ S &\leftarrow (AND\ I_i\ S_j) \\ S &\leftarrow (AND\ S_i\ S_j) \end{aligned}$$

For example, referring to Figure 6, each G-source, GL-1, GL-2, and GL-3, generates its own illuminated portion and shadow portion. Applying *AND* operations to an illuminated portion gives the result shown in Figure 6 (a). The

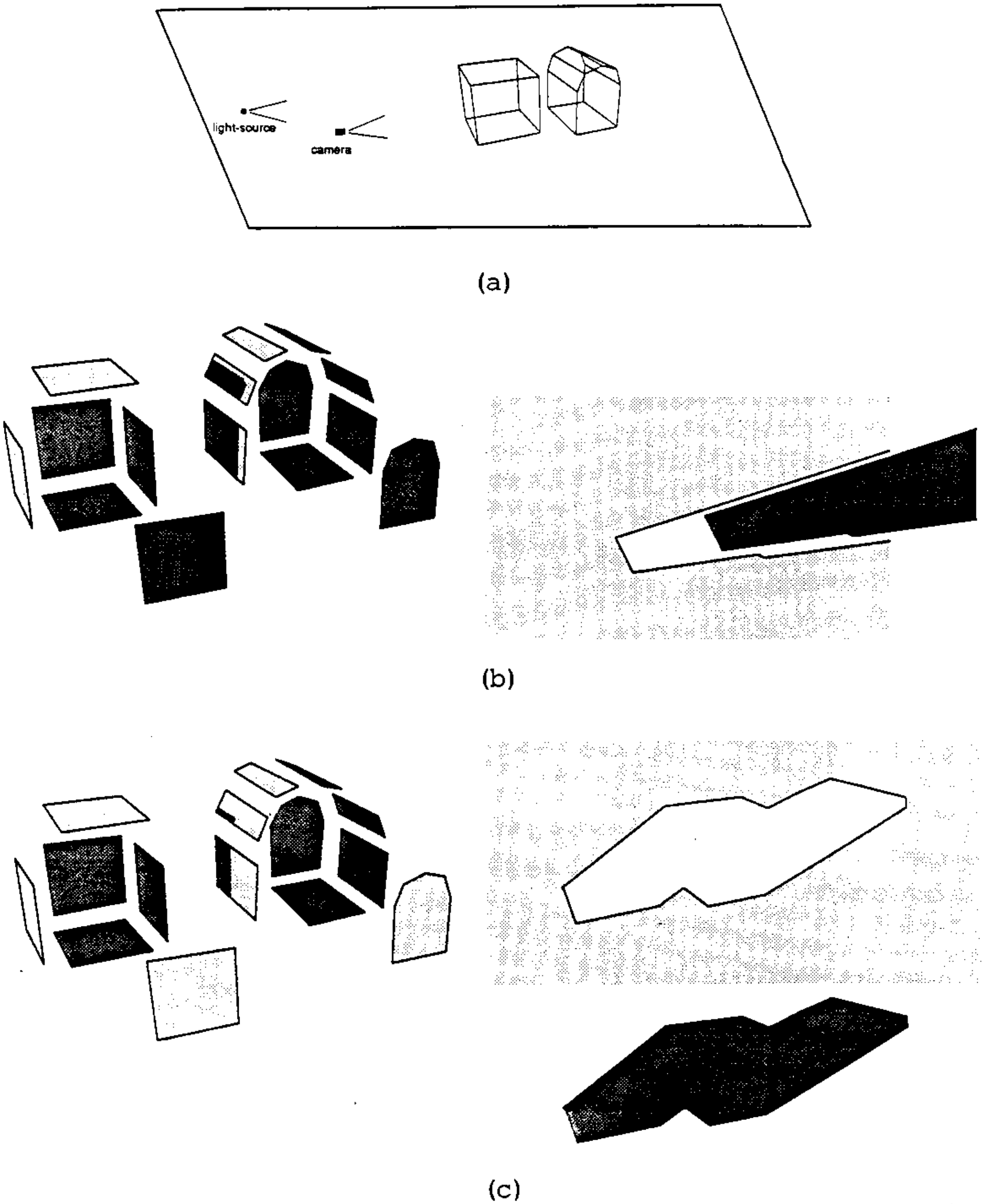


Figure 5: 3D face properties: (a) 3D scene consists of one cube, one arch, and one table. A sensor, which consists of two G-sources, one light source from the right direction and one TV camera from the left direction, is applied to the 3D scene; (b) 3D illumination properties generated by the light source; (c) 3D illumination properties generated by the TV camera.

illuminated portions can be obtained by applying *intersection operations*, as shown in Figure 6 (b). The shadowed portions can be obtained by applying *union operations*, as shown in Figure 6 (c). VANTAGE implements this *AND* operation as intersection operations through a recursive polygon clipping process [23]. See Appendix B for more details.

We define *OR* operations as follows:

$$\begin{aligned} I &\leftarrow (OR\ I_i\ I_j) \\ I &\leftarrow (OR\ I_i\ S_j) \\ S &\leftarrow (OR\ S_i\ S_j) \end{aligned}$$

The operation *OR* has a complementary relationship with the *AND* operation. Let us suppose the same three light sources generate three illuminated portions on the same 3D face and we need a part of *OR* of the three illuminated areas. See Figure 7 (a). The illuminated portion can be obtained by applying *union* operations to the illuminated portions, as shown in Figure 7 (b), while the shadowed portion can be obtained by applying *intersection* operations to the shadow portions.

Let us consider the case where we generate an appearance of a 3D face, from a TV camera, of a portion illuminated by a different light source. We will generate only a portion that is illuminated and visible. The operation *AND* is executed between the illuminated portion and visible portion on the 3D face. Then, that portion is projected using the operation, *PROJECTION*, onto the image plane of the TV camera.

These set operations are defined as nodes of the sensor composition tree. In order to create such nodes, we use

```
(SCTNODE operation node-name left-node right-node)
```

where operation is one of the following:

- `AND` – AND operation is applied between two illuminated portions.
- `OR` – OR operation is applied between two illuminated portions.
- `PROJECTION` – a projection is generated from a 3D face.

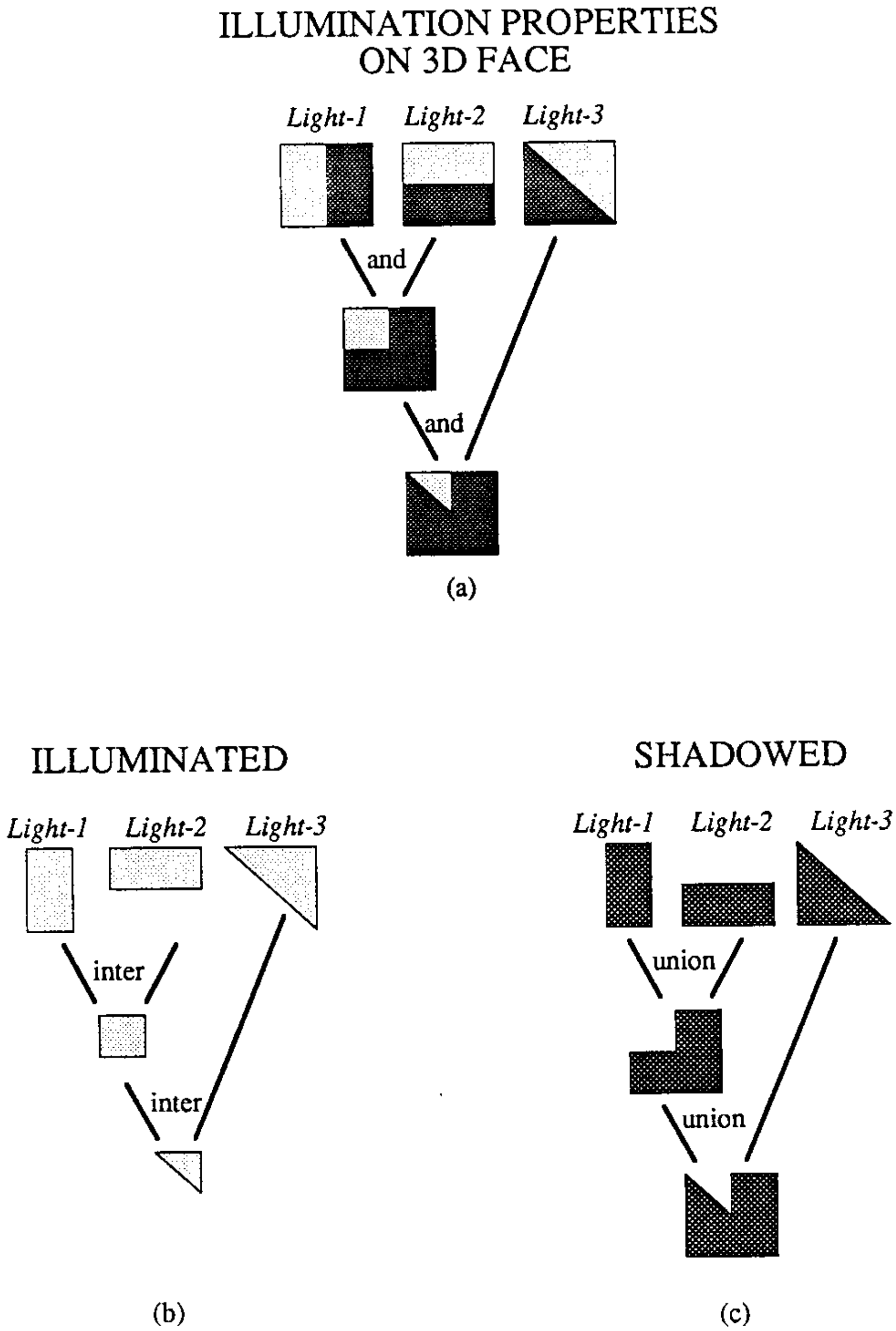
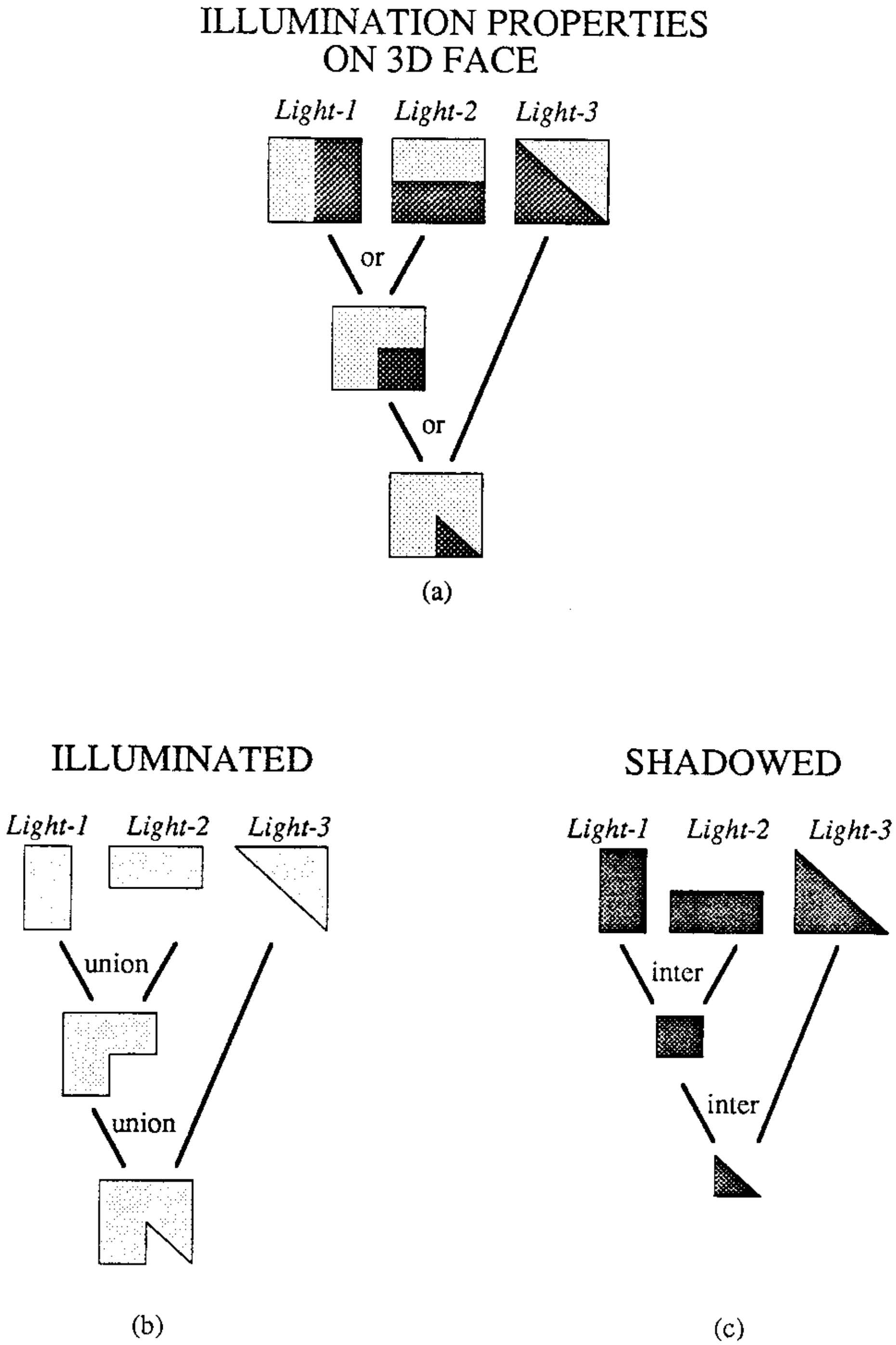


Figure 6: AND operation among illuminated portions.



(a)

(b)

(c)

Figure 7: OR operation among illuminated portions.

Once a sensor composition tree is represented, it is applied to a 3D scene by a function called `apply-sensor-to-scene`. This function is executed in three steps. In the first step, portions illuminated by each G-source on a 3D face are generated and stored as 3D illumination properties on the 3D face. These properties define how to divide each 3D face into several subfaces with the same properties. In the second step, set operations, such as the *AND* and *OR* operation are executed between illumination properties along the sensor composition tree, and the results are also stored on a 3D face. In the third step, visible illuminated and visible shadowed portions of 3D faces are projected onto the image plane as its 2D appearance.

The resulting 2D appearance consists of visible 2D faces, which are divided into illuminated portions and shadowed portions. Strictly speaking, the sensor composition tree should give only illuminated portions, and thus, the resulting 2D appearance should consist only of illuminated portions. However, depending on applications, we may be interested in shadowed portions. Thus, the current implementation of VANTAGE will produce shadowed portions as well as illuminated ones.

4 Two-Dimensional Structure

This section examines the representational structure of 2D appearances. Before discussing the main topic, we will review 3D representation of VANTAGE [11]. Then, we will examine several 2D structures having various priorities regarding illumination information with respect to the 3D structure.

4.1 3D Structure

We will begin by considering the definition of a 3D face. Several definitions of faces are possible depending on continuity conditions across their boundaries. Referring to Figure 8, if we define a face as the surface patch such that across its boundary C^0 continuity is violated, then, the entire boundary of an object becomes one face. VANTAGE refers to this level as the *Solid level.*

We can define another type of face as the surface portion such that across its boundary, at most, C^1 continuity is violated. This level, called as *Merged level,* generates faces as shown in Figure 8(b). Note that the cylindrical part and the planar part are connected. We can also define C^2 faces as shown in Figure8(c). This level is referred to as the *Grouped level* in VANTAGE.

While Class $C^0, C^1, andC^2$ faces are defined purely mathematically, VANTAGE has one more level; at this bottom level, the boundary representation of a object contains only planar faces, which have originated from either polyhedral primitives or from polyhedral approximations of curved primitives.[6] The latter faces are referred to as approximation faces. This level will be referred to as the *APP level.*

VANTAGE builds the higher levels of representation from the APP level representation. First of all, VANTAGE can group a set of adjacent planar faces that approximate the same curved surface into one *curved* face. This grouping operation can generate a grouped level (Class C^2) representation of an object. Second, faces that are tangent across an edge, that is, C^1 continuous, are also merged into a *merged* face and the merged level (Class C^1) representation is completed. Finally, VANTAGE can group all faces that are connected and generate the solid level (Class C^0) representation. We will refer to this whole structure as the *3D structure* of an object.

[6] VANTAGE represents a complicated object as a collection of simple primitive objects, such as a cube or a cylindrical object, by using CSG representation.

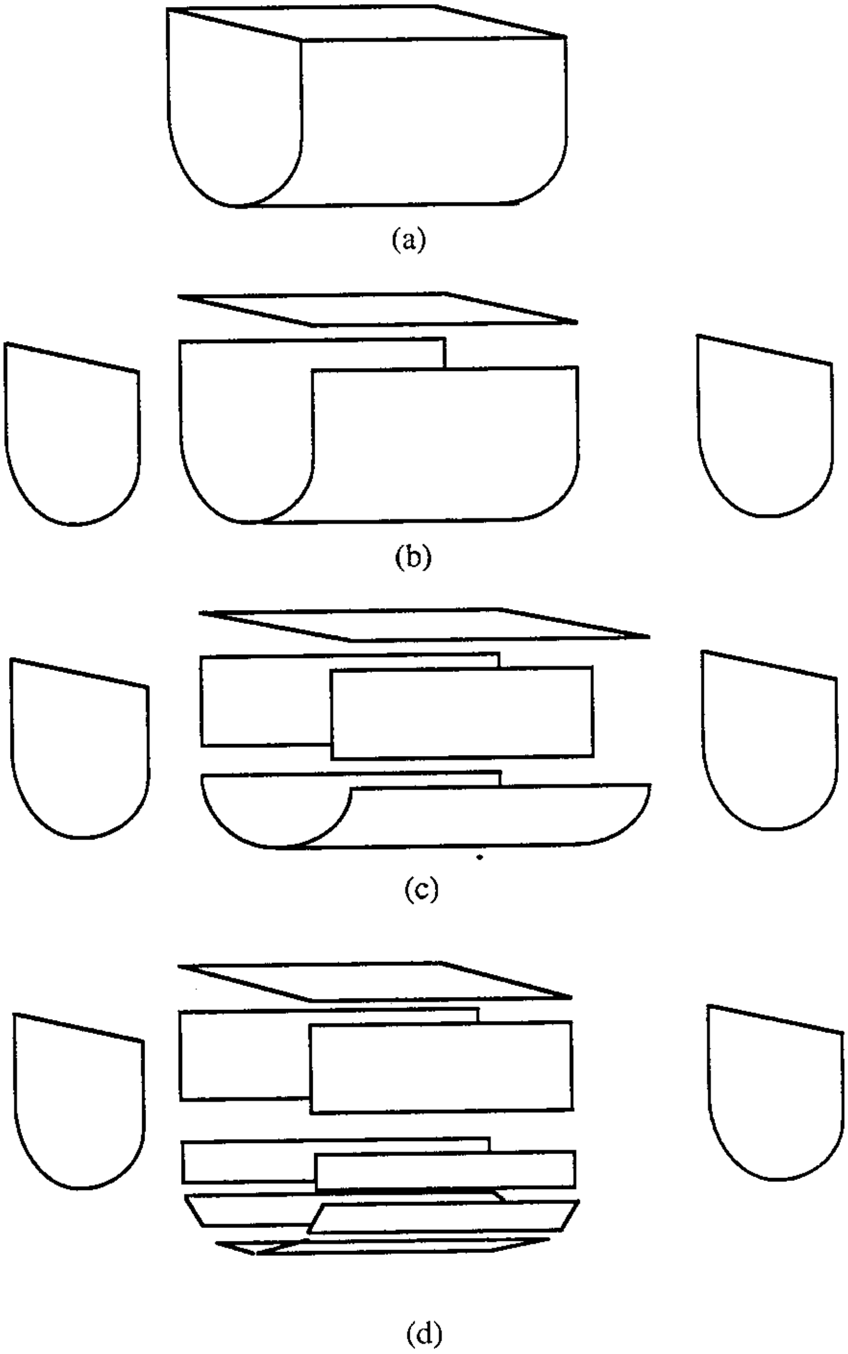


Figure 8: Definition of Faces: (a) Class C^0 face: *Solid level*; (b) Class C^1 faces: *Merged level*; (c) Class C^2 faces: *Grouped level*; (d) polygon approximation faces: *APP level face*.

VANTAGE maintains a 3D boundary representation of an object at each level. This representation contains lists of faces, edges and vertices at each level. Vertices contain their respective coordinate values, and edges join these vertices. The edges are either straight lines or curved lines, depending on the level. The faces are either planar polyhedra or curved polyhedra and are represented by a collection of connected edges at that level. The neighborhood information or *topology* relates the edges, faces, and vertices of the object, and is stored in the form of *winged-edge* representation [2], where each level maintains its own winged-edge representation. For example, the arch in the example scene shown in Figure 5(a) has the 3D structure as shown in Figure 9.

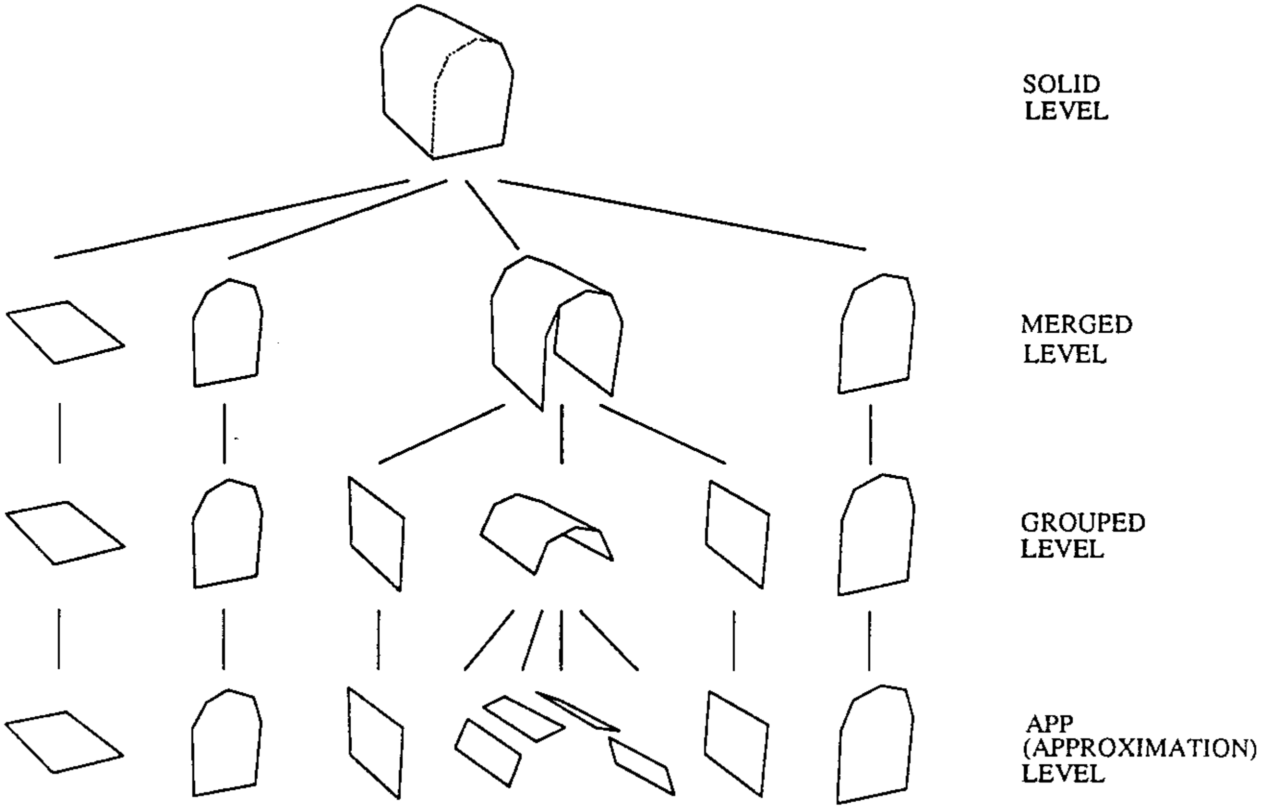


Figure 9: 3D structure.

Several 2D structures are possible, depending on the priority of illumination information with respect to the 3D levels. For example, we can put the illumination information below the 2D APP level; we can also put the illumination information above the 2D solid level. In order to examine these cases, we will consider the 3D scene shown in Figure 5(a). We will assume a sensor which consists of two G-sources: a light source which illuminates the scene from the right direction and a TV camera which observes it from the left direction. Applying this sensor to the 3D scene, VANTAGE will generate a 2D appearance of it. In the following subsections, we will investigate several 2D structures of the 2D appearance.

4.2 Flat Structure

The 2D appearance of a 3D scene is an explicit symbolic representation obtained by projecting the scene onto the image plane using the sensor composition tree. VANTAGE first generates a 2D boundary representation by projecting the 3D APP level boundary representation of the scene. Thus, a 2D boundary representation of a scene is a set of 2D faces, 2D edges, and 2D vertices, linked by winged-edge relations. Each 2D element corresponds to its 3D counterpart.

Then, VANTAGE projects the 3D illumination properties, generated along the sensor composition tree, of the 3D faces onto the 2D boundary representation. Thus, each 2D face is divided into several subfaces to correspond to illumination properties of its 3D face. Subfaces do not maintain winged-edge relations; they represent internal structure of a 2D face.

Thus, the basic 2D boundary representation is a collection of 2D APP faces, which maintain the illumination properties below them as subfaces. This representation, which we will refer to as the *flat structure*, is the default 2D representation of VANTAGE.

The current implementation of VANTAGE supports three kinds of 3D illumination properties: illuminated (*I*), cast-shadowed (*CS*), and self-shadowed (*SS*). These three values can be consolidated into two values, illuminated (*I*) and shadowed (*S*) by the operation:

$$\begin{aligned} I &\leftarrow I \\ S &\leftarrow (OR\ CS\ SS). \end{aligned}$$

This is necessary to be able to apply *AND/OR* operations to illumination properties.[7]

[7] VANTAGE has two methods of generating a 2D appearance: applying a sensor composition

Figure 10(a) shows the 2D appearance of the example scene. This 2D appearance is represented by the flat structure as shown in Figure 10(b). Note that the several approximate faces are divided into two parts to store the illumination information.

4.3 Illumination-Oriented Structure

This structure gives priority to the illumination information. In it, a 2D appearance will be classified into two categories: illuminated, and shadowed. Within each category, its own 2D structure is constructed.[8]

First, VANTAGE classifies the 2D subfaces into the two categories. In each category, the following operations are repeated. From the 2D APP level, as is the case in 3D, VANTAGE builds three more levels of representation based on surface properties. VANTAGE can group a set of adjacent 2D subfaces which come from 3D APP faces of the same curved surface into a *2D curved* face. Second, 2D faces which come from 3D faces that are tangent across an edge, are also merged into a *2D merged* face. And finally, VANTAGE groups 2D subfaces which come from the same object into a complete 2D structure.

Figure 11 shows the illumination-oriented structure of the example scene. Note that the illuminated portion of the cylindrical part of the arch is represented in several levels.

This illumination-oriented structure is useful when applying the sensor composition tree to a 3D scene, because the sensor usually detects only illuminated portions which are in the illuminated category.

This structure is also useful when generating a 2D structure of a 3D scene of a simple camera projection. In this case, we will put the light source at the same place as the viewer. Then, all visible portions become illuminated, and thus, we can get a 2D structure of all visible portions of a 3D scene.

As mentioned before, the current implementation of VANTAGE supports three kinds of illumination properties: illuminated (*I*), cast-shadowed (*CS*), and self-shadowed (*SS*). In the sensor composition tree module, the following oper-

tree, and applying a TV camera and a light source directly. We explain the former method in this paper. If we use the latter method, we can generate a 2D appearance which also maintains three categories of illumination information: illuminated, self-shadowed, and cast-shadowed.

[8] If the 2D flat structure maintains three categories of illumination properties, then three structures are generated within the three categories: illuminated, cast-shadowed, and self-shadowed.

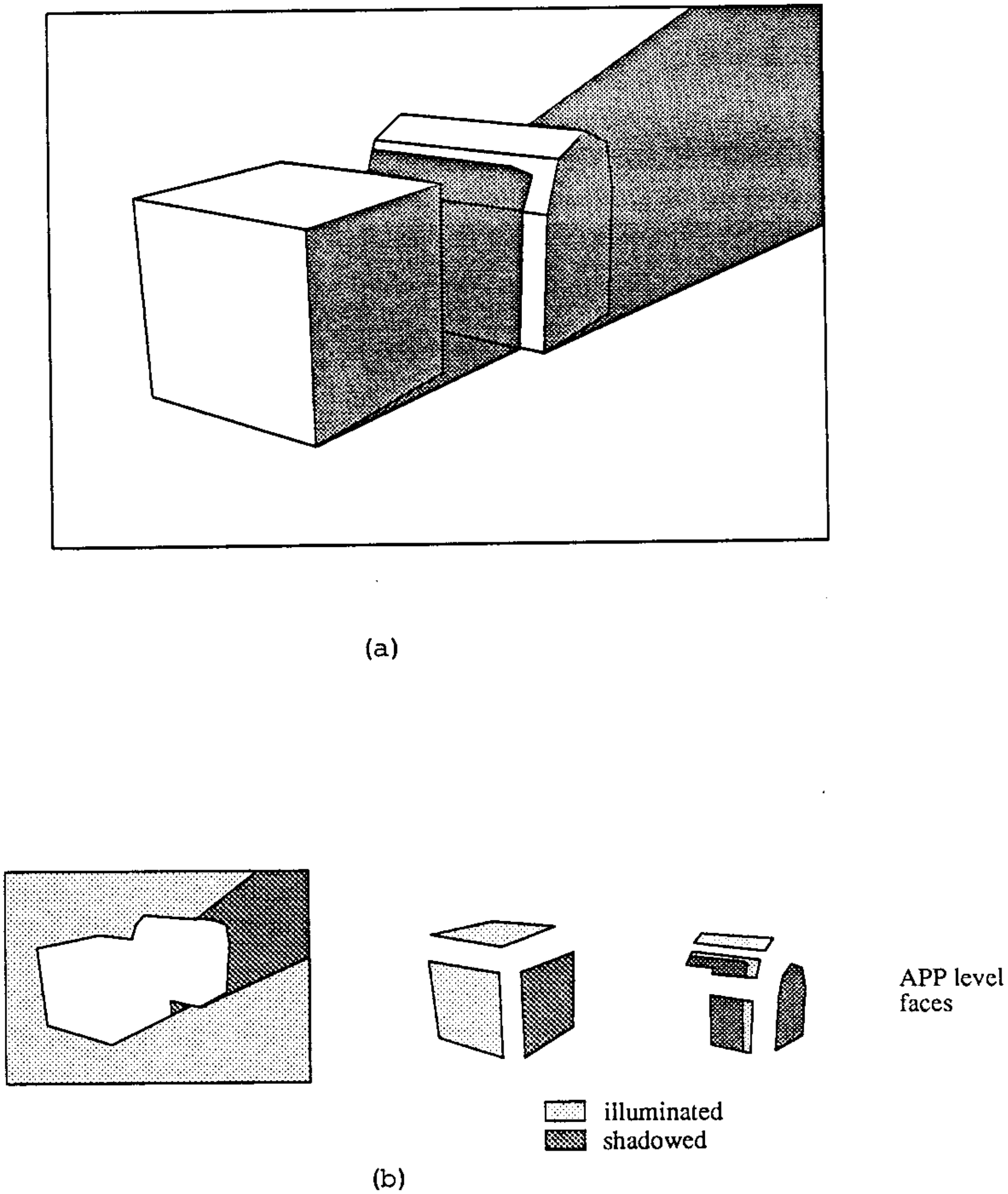


Figure 10: Flat structure: (a) 2D appearance of the example scene; (b) Flat structure of the 2D appearance.

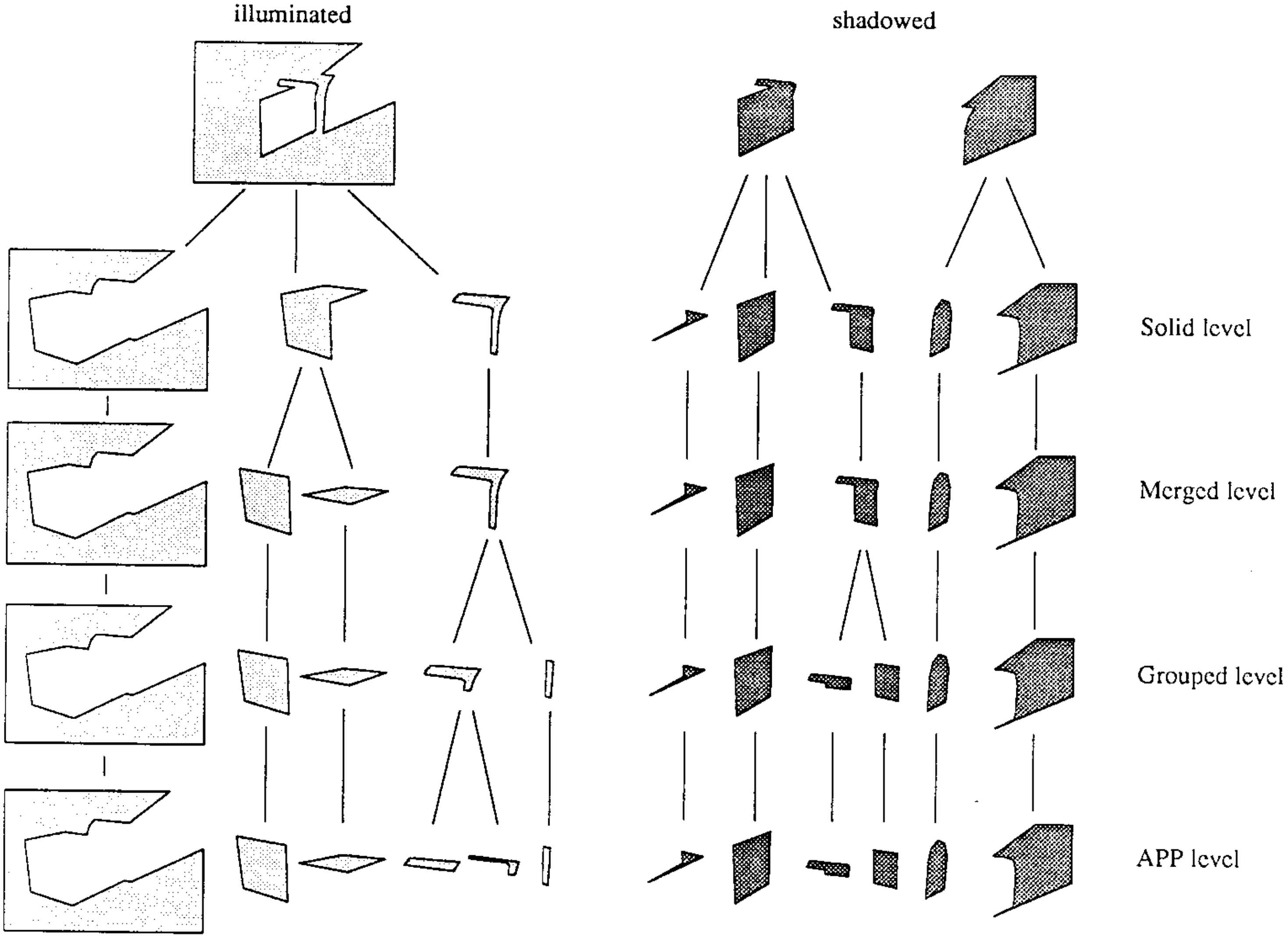


Figure 11: Illumination-Oriented Structure.

ations are performed on the illumination properties:

$$\begin{aligned} I_i &\leftarrow I_i \\ S_i &\leftarrow (OR\ CS_i\ SS_i), \end{aligned}$$

where i denotes that the illumination property is given by the G-source i. However, we can also define

$$\begin{aligned} I_i &\leftarrow (OR\ I_i\ CS_i) \\ S_i &\leftarrow SS_i \end{aligned}$$

Then, by applying *OR* operations to a 3D scene, the shadow category gives a collection of the portions represented by $(AND\ SS_i\ SS_j)$. Using this schema, we can generate 2D structures of various kinds of combinations between G-sources. This is particularly useful when each G-source has a different color and we are interested in analyzing the color of a 2D appearance.

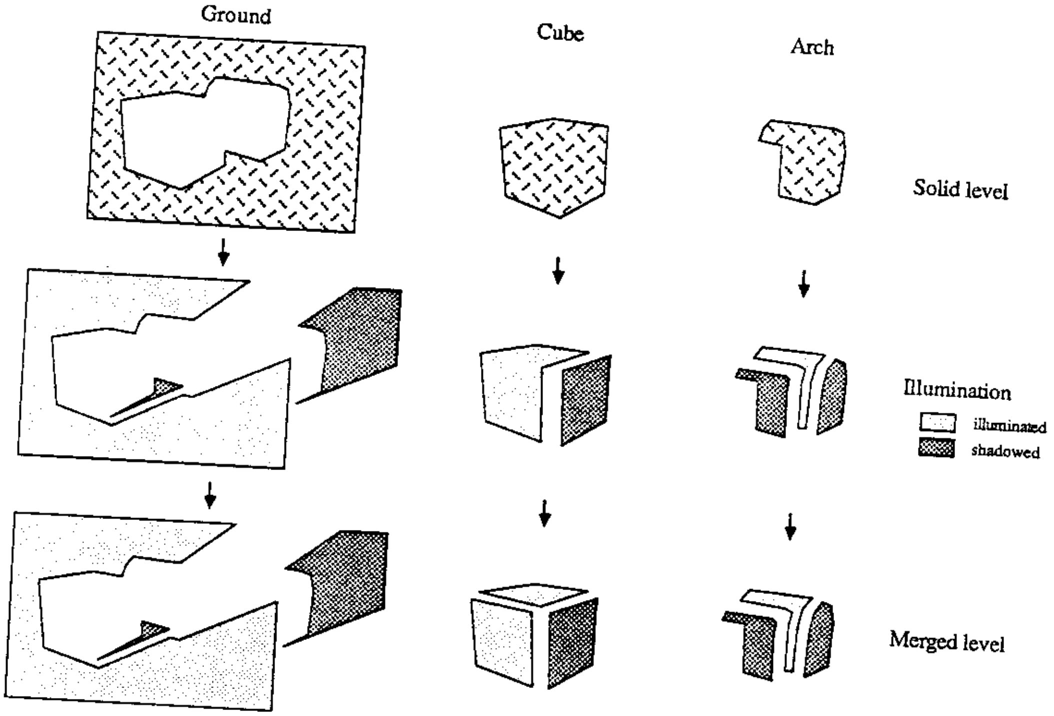
4.4 Geometry-Oriented Structure

We can also set the priority of the illumination information somewhere between the top level (solid level) of the 2D structure and the bottom level (APP level).

Let us suppose that we put the illumination information between the solid level and the merged level. Then, at each solid, a face structure is generated. In the example scene case, since there are three objects, three structures are generated: the front cube, the middle cylinder, and the back cube. Then, each structure will be divided into two parts: illuminated and shadowed. See Figure 12(a). This structure is particularly useful when we analyze illumination conditions of each object.

We can also put the illumination information between the merged level and the grouped level. Then, a merged face will be divided into three categories of illumination conditions, as shown in Figure 12(b). This structure is useful when we produce detectable faces by means of an active sensors, because the direct output of an active sensor are the collection of C^1 faces.

The illumination information can also be stored between the grouped level and the APP level. Then we will obtain the structure as shown in Figure 12(c).



(a)

Figure 12 (cont)

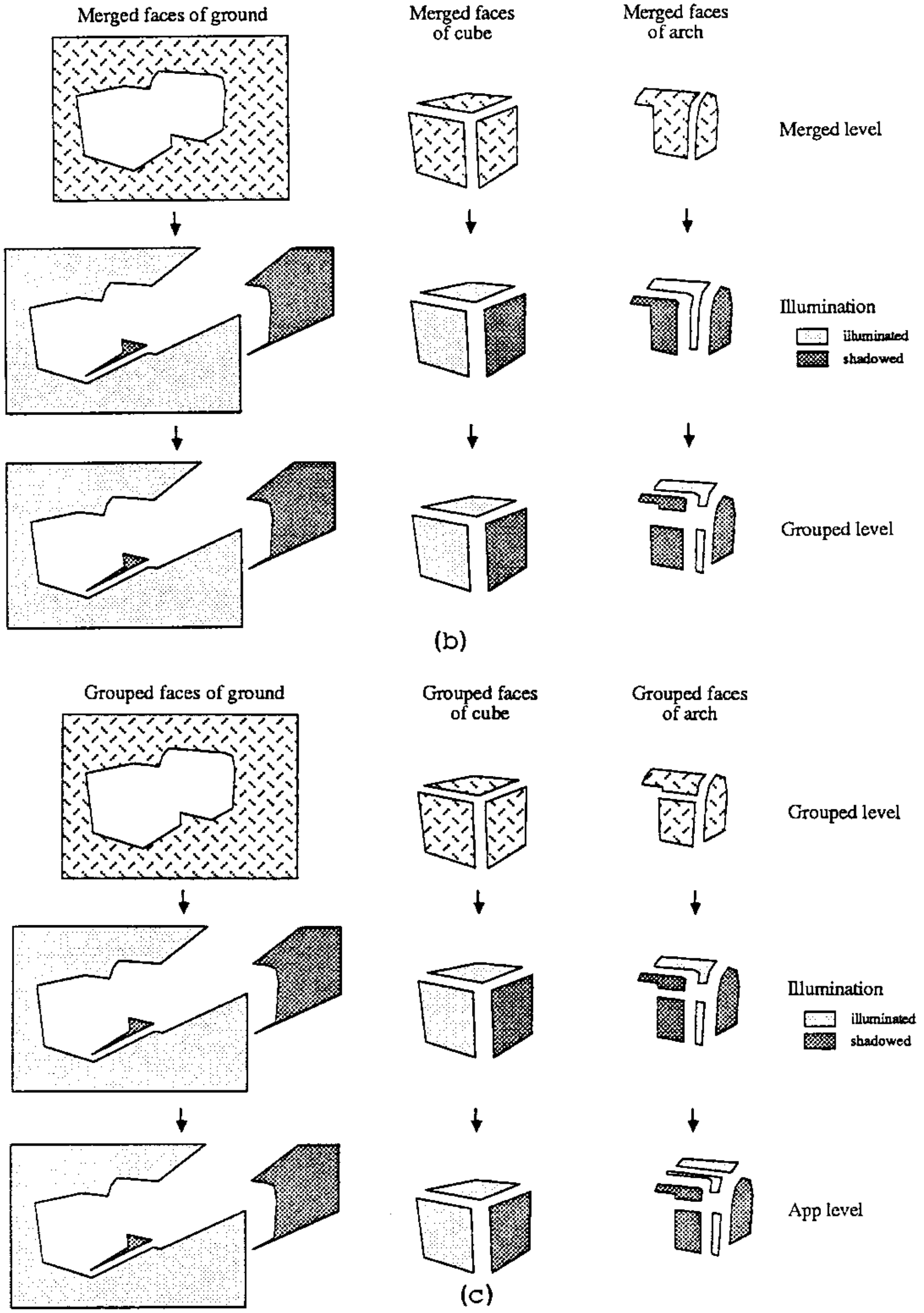
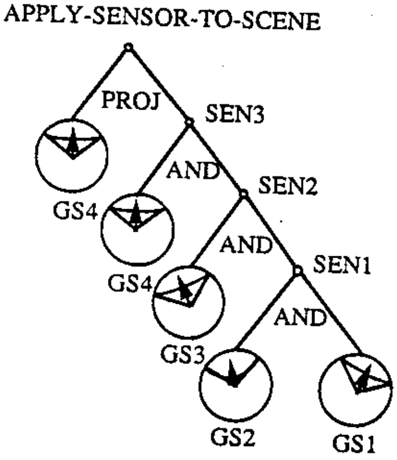


Figure 12: Geometry-oriented structure: (a) Solid-merged; (b) Merged-grouped; (c) Grouped-APP.

5 Applying VANTAGE to Model Based Vision

This section briefly outlines how to apply the VANTAGE geometric/sensor modeler to one of the applications of Model-based vision: automatic generation of object recognition programs.

From the 3D representation of an object in VANTAGE and the sensor detectability of a particular sensor, we can generate a possible appearance of that object under that sensor. For example, Figure 13(a) depicts a sensor composition tree of photometric stereo, while Figure 13(b) represents the internal representation of the tree in VANTAGE.



(a)

```
> (SENSOR-TREE)
--------------------------------------------------------------------------
NODE     TYPE      LEFT-NODE   RIGHT-NODE   PARENT[S]
--------------------------------------------------------------------------
SEN3     AND       CS4         SEN2         NIL
CS4      CAMERA    NIL         NIL          (SEN3)
SEN2     AND       CS3         SEN1         (SEN3)
CS3      LIGHT     NIL         NIL          (SEN2)
SEN1     AND       CS2         CS1          (SEN2)
CS2      LIGHT     NIL         NIL          (SEN1)
CS1      LIGHT     NIL         NIL          (SEN1)

> (APPLY-SENSOR-TO-SCENE  'SEN3 'SCENE1 'CS4)
```

(b)

Figure 13: Sensor composition tree in VANTAGE: (a) The sensor composition tree of the photometric stereo; (b) The internal representation of the sensor composition tree. In the current implementation, the internal representation keeps only *AND/OR* operations. The final projection is combined into a `apply-sensor-to-scene` function.

Figure 14 shows an example of the application of the sensor composition tree of the photometric stereo to a car model. Figure 14 (a) is the car model. This sensor consists of four G-sources: three light sources and one TV camera. Thus, the first step in applying the sensor composition tree to the car model generates 3D illumination properties of four G-sources, from *GS1* through *GS4*. These 3D illumination properties are stored on 3D faces. Then, set operations between 3D illumination properties are executed along the sensor composition tree as shown in Figure 14(b). Finally, the projection of the car is generated with respect to the G-source *GS4*. Figure 14(d) shows the corresponding needle map of the car obtained by the photometric stereo system, while Figure 14 (c) shows the original scene. They correspond with each other quite well. As you can see in this example, we cannot predict object appearances correctly without the sensor modeling capability of a modeler like VANTAGE.

Using the illumination-oriented structure, we can obtain the portions of the car detectable by the photometric stereo. Since the appearance is representable by frames, we can easily convert the output from VANTAGE to the appearance represented in Figure 15(a). We can classify and categorize various appearances into possible aspects, where each aspect shares the same combination of detected 2D faces. Since whether a 2D face is detected depends on sensor detectability, the possible aspects also depend on sensor detectability. In other words, without this capability of VANTAGE, we cannot generate possible aspects of an object under one particular sensor. Thus, this capability is one of the most fundamental components of an automatic generation of object recognition program.

One aspect structure, a symbolic representation of an aspect, is constructed at each aspect group. Since each 2D appearance is represented symbolically, it is relatively easy to construct such an aspect structure (as shown in Figure 15(b)) based on 2D appearance structures generated from the output of VANTAGE. Predicted ranges of uncertainty of geometric features are determined using the sensor reliability and added to the aspect structures.

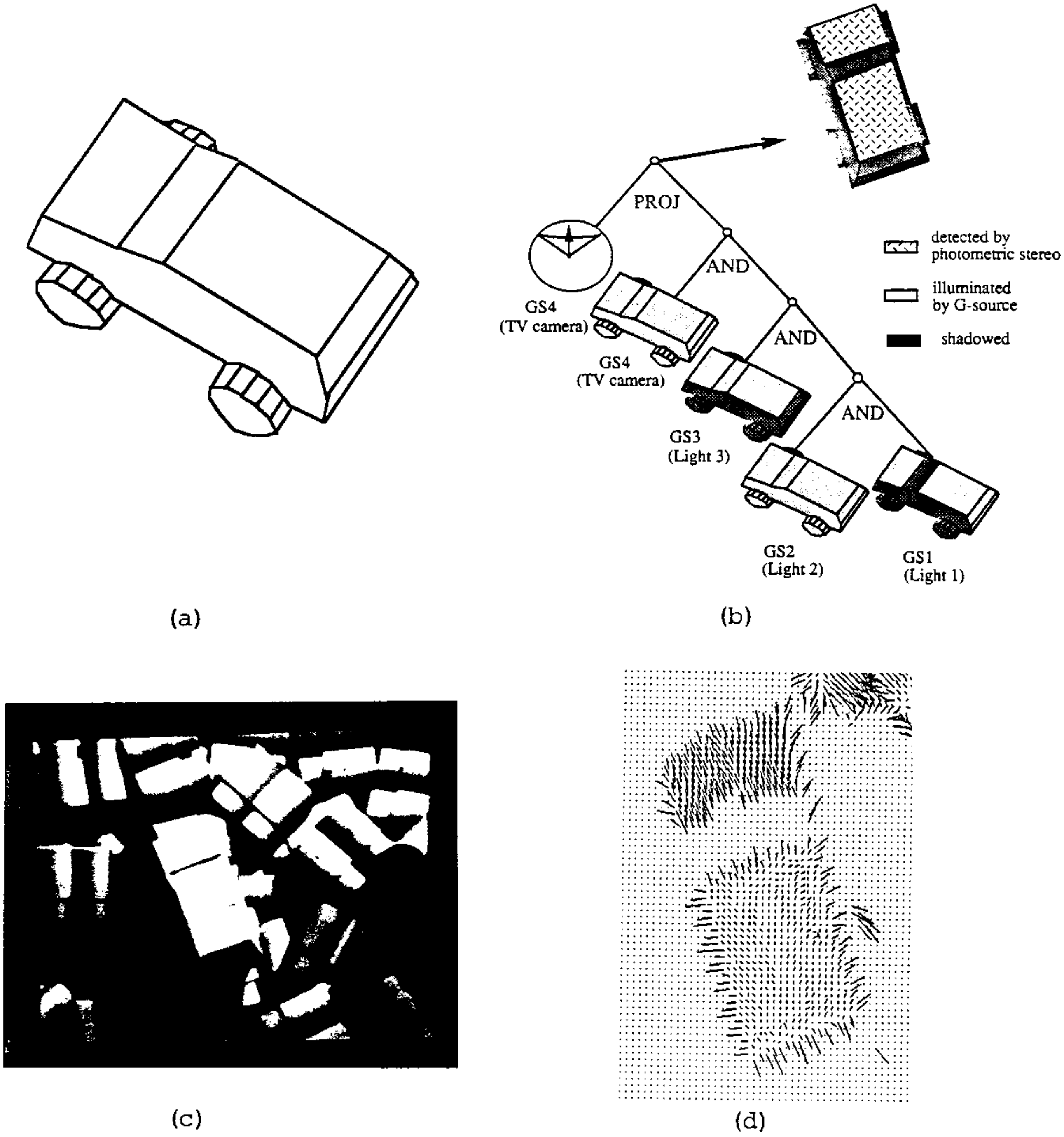


Figure 14: Predicted appearance by VANTAGE and obtained appearance: (a) Car model; (b) Applying the sensor composition tree to the car model; (c) Original scene containing the car; (d) A needle map obtained by the photometric stereo. The predicted appearanace is consistent with the obtained appearance.

The recognition strategy is divided into two parts: aspect classification and linear shape change determination. An unknown appearance will be classified into one of the possible aspects, and then the precise attitude will be determined within that aspect.

Aspect classification is generated by performing recursive sub-divisions of possible aspects examining uncertainty ranges of aspect features. Generation begins by creating a root node which contains all possible aspects. After this operation, the following operation will be applied recursively to each node containing a group of aspects along a set of features. At each node, the generation process examines whether it can divide a group of aspects at the node into several groups of aspects by examining the uncertainty range of one particular feature of the aspects. If it can, it stores the feature name at the node, generates subnodes corresponding to subgroups of aspects, and connects them to the node. It also stores the threshold values to divide the groups at the node. If it cannot, it does nothing. Then it repeats the above process using the next feature along the set of features. The generated recognition strategy is represented as a tree structure, which we refer to as an *interpretation tree*, whose intermediate nodes correspond to classification stages of aspects and store feature kinds and values for classifications. Each leaf node contains one particular aspect.

At each leaf node, aspect classification gives correspondences between image regions and model faces. Linear change determination is generated by using these correspondences [9]. We have made rules concerning how to define a face coordinate system on a face. Using these rules, the generation process defines a face feature coordinate system on a model face, stores the used rules, and calculates the transformation from the face coordinate system to the body coordinate system from a geometric model. Once this process generates the recognition strategy, it converts the strategy into a program using an object library.

The generated program is applied to a scene. The program extracts a feature value, specified at each node, from the region, compares it with the threshold feature values, and classifies the region into one or several possible aspects. The generated program estimates the body coordinate systems at these possible aspects. Comparing the edge distributions generated by VANTAGE from the estimated body coordinate systems and the real edge distribution, the program determines the most likely position and attitude, as shown in Figure 16, where the most likely one, generated by VANTAGE, is superimposed over the scene.

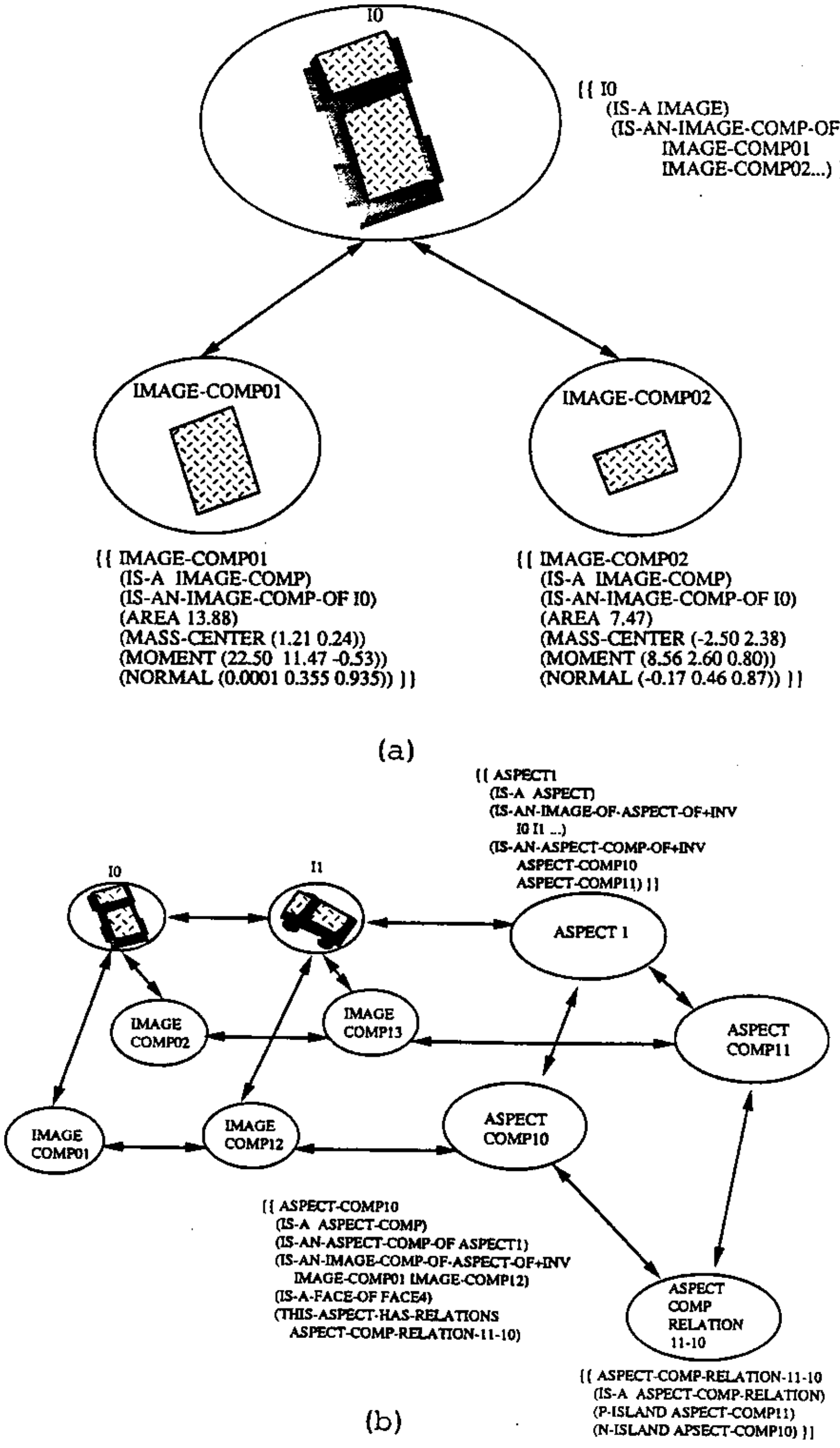

I0
{{ I0
(IS-A IMAGE)
(IS-AN-IMAGE-COMP-OF+INV
IMAGE-COMP01
IMAGE-COMP02...) }}
IMAGE-COMP01
IMAGE-COMP02
{{ IMAGE-COMP01
(IS-A IMAGE-COMP)
(IS-AN-IMAGE-COMP-OF I0)
(AREA 13.88)
(MASS-CENTER (1.21 0.24))
(MOMENT (22.50 11.47 -0.53))
(NORMAL (0.0001 0.355 0.935)) }}
{{ IMAGE-COMP02
(IS-A IMAGE-COMP)
(IS-AN-IMAGE-COMP-OF I0)
(AREA 7.47)
(MASS-CENTER (-2.50 2.38)
(MOMENT (8.56 2.60 0.80))
(NORMAL (-0.17 0.46 0.87)) }}
(a)
{{ ASPECT1
(IS-A ASPECT)
(IS-AN-IMAGE-OF-ASPECT-OF+INV
I0 I1 ...)
(IS-AN-ASPECT-COMP-OF+INV
ASPECT-COMP10
ASPECT-COMP11) }}
I0
I1
ASPECT 1
IMAGE COMP02
IMAGE COMP13
ASPECT COMP11
IMAGE COMP01
IMAGE COMP12
ASPECT COMP10
{{ ASPECT-COMP10
(IS-A ASPECT-COMP)
(IS-AN-ASPECT-COMP-OF ASPECT1)
(IS-AN-IMAGE-COMP-OF-ASPECT-OF+INV
IMAGE-COMP01 IMAGE-COMP12)
(IS-A-FACE-OF FACE4)
(THIS-ASPECT-HAS-RELATIONS
ASPECT-COMP-RELATION-11-10)
ASPECT COMP RELATION 11-10
{{ ASPECT-COMP-RELATION-11-10
(IS-A ASPECT-COMP-RELATION)
(P-ISLAND ASPECT-COMP11)
(N-ISLAND APSECT-COMP10) }}
(b)

Figure 15: Aspect structure: (a) Image structure. This structure is generated based on the detectable portions, represented using the illumination oriented structure; (b) Aspect structure.

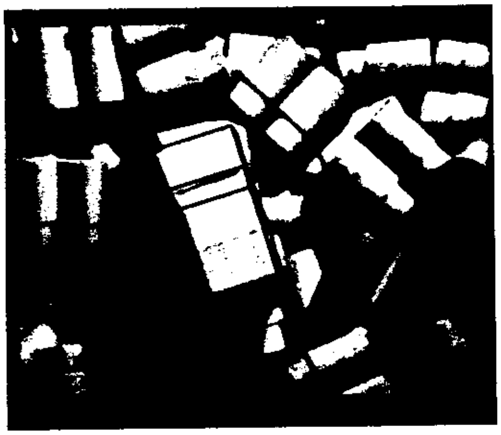


Figure 16: Example: Predicted object position and attitude are generated by VANTAGE and superimposed on the image.

6 Summary

This paper has described a method of modeling sensor detectabilty and its implementation in VANTAGE sensor modeler. Each sensor consists of one or more light sources and TV cameras. Sensor detectability, that is where the sensor can detect, can be specified by a set operation of portions illuminated by its light sources and portions visible to its TV cameras.

In order to represent this sensor detectability, we defined a G-source as an abstract sensor component representing either a light source or a TV camera. Then, we proposed G-source illumination conditions to describe under what condition each G-source illuminates surface patches with respect to its illumination direction.

One of the most important findings is that the detectability of a sensor can be decomposed into the sensor's component G-sources' illumination conditions and set operations between them. We can apply its G-sources' illumination conditions independently to an object model, and then apply set operations between them, to obtain detectable portions.

According to these findings, we proposed a sensor composition tree, which consists of G-source illumination conditions as its leaf nodes and set operations as its branch nodes. These set operations perform the combination of the component G-source illumination conditions. The execution of the sensor composition tree is implemented in three steps: applying the component G-sources' illumination conditions to all 3D faces of the object and storing them as 3D illumination properties on those faces; applying set operations between stored illumination properties; and projecting the resulting illuminated portions onto the image plane to generate a 2D appearance.

We also proposed several 2D structures to represent detected 2D appearances, given by VANTAGE using a sensor composition tree. Since the 3D structure of VANTAGE has four levels of representation: solid, merged, grouped, and approximation and since 2D appearances of 3D scenes can be obtained by projection, illumination information can be stored at, above, between, or below these levels in 2D.

Finally, we showed how to use these capabilities in model-based vision.

7 Acknowledgments

Takeo Kanade provided many useful comments and encouragements. Keith Petersen proofread earlier drafts of this paper and provided many useful comments which have improved the readability of this paper. Purushothaman Balakumar and Regis Hoffman also partly implemented VANTAGE. Ki Sang Hong actively used VANTAGE for his project and provided many useful comments which have improved the flexibility and stability of this system.

Appendix A: Definition of Coordinate systems

Face We define the z-axis of the feature coordinate system to agree with the surface normal, and the $x - y$axes lie on the face, but are defined arbitrarily otherwise.

Edge We define the z-axis to agree with the average direction of the two normals of the two adjacent faces incident to the edge. We define the x-axis of the feature coordinate system to agree with the edge direction. The y-axis is determined according to x and z.

Vertex We define the z-axis to agree with the average direction of the normals of the adjacent faces incident to the vertex. The x and y axes lie on the plane perpendicular to the z-axis, but are defined arbitrarily otherwise.

Appendix B: Polygon Clipping

This appendix presents a detailed description of a general algorithm for polygon clipping This algorithm is based upon an algorithm described by Weiler and Atherton [23], which was modified for implementation of the 2D part of VANTAGE.

The algorithm clips one 2-D polygon (subject) to the borders of another 2-D polygon (reference). Both polygons may be concave and may have one or more convex or concave holes. The final result consists of two sets of 2-D polygons: the regions of the subject polygon that are inside the borders of the reference polygon, and the ones that are outside. The algorithm takes care of all cases that could produce degenerated polygons. The polygonal form of the input and output of the algorithm makes it very suitable for any applications involving manipulations of 2-D polygons.

Polygon Clipping

Description of a General Algorithm

Jean-Christophe Robert

B.1 Input and output

B.1.1 Input

The algorithm performs the clipping of one 2-D polygon (called *subject*), by another 2-D polygon (called *reference*). Both polygons may be concave and have one or several holes.

A 2-D polygon is defined by its *outer-boundary*, and possibly one or several *hole-boundaries*. A boundary can be defined as an ordered list of vertices. Each *vertex* is considered here as a symbolic element, characterized by its x-y coordinates in a specified 2-D frame of coordinates. Each pair of consecutive vertices defines an *edge* of the boundary. The vertices of an outer-boundary are listed in a counter-clockwise order, whereas the vertices of a hole-boundary are listed in a clockwise order (see Figure B-1). It is assumed that a vertex cannot appear more than once in a boundary. It is possible, however, that a vertex belong to more than one boundary of a polygon.

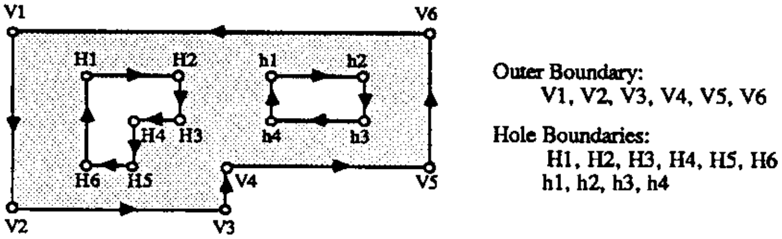


Figure B-1: Polygon definition

The following operations have to be completed before starting the clipping process itself:

1. **Comparison of the bounding boxes of the polygons**

   This test is very useful if the clipping algorithm is to be applied repeatedly on a large set of polygons. It consists of a simple comparison of the maximum and minimum x and y coordinates of the two polygons. If the bounding boxes of the two polygons do not overlap, then the subject is completely outside the reference, and no further operation is necessary (See Figure B-2).

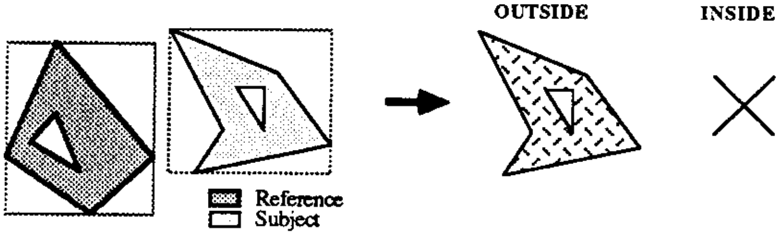


Figure B-2: Bounding boxes do not overlap

2. **Check for duplicate vertices in a boundary of either polygon**

Whenever a polygon has a boundary that contains a vertex more than once, this polygon can be split into one or several polygons whose boundaries have no duplicate vertices (see Figure B-3). This step ensures that the input polygons comply with the properties mentioned above.

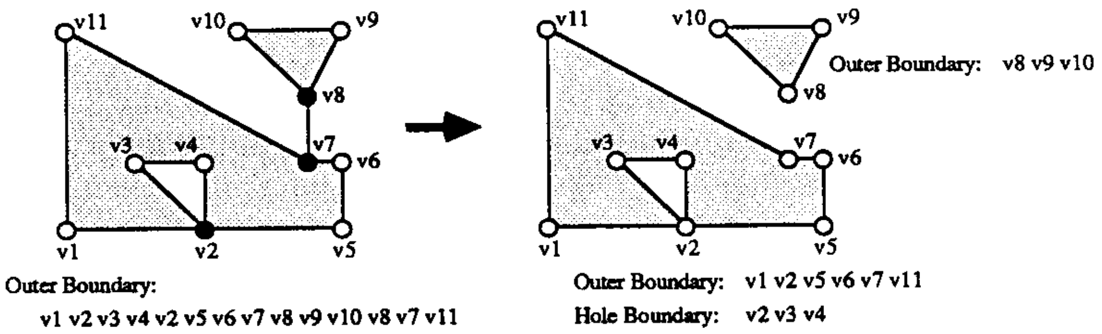

v11
v10
v9
v8
v7
v6
v3
v4
v1
v2
v5
Outer Boundary:
v1 v2 v3 v4 v2 v5 v6 v7 v8 v9 v10 v8 v7 v11
v11
v10
v9
Outer Boundary: v8 v9 v10
v8
v7
v6
v3
v4
v1
v2
v5
Outer Boundary: v1 v2 v5 v6 v7 v11
Hole Boundary: v2 v3 v4

Figure B-3: Elimination of duplicate vertices

3. **Determination of the vertices that are common to both polygons, and modification of the boundaries of the two polygons so that they include all the common-vertices.**

A *common-vertex* is a vertex matching one of the three following criteria:

a. vertex already contained in one boundary (outer or hole) of both polygons.

b. vertex already contained in one boundary (outer or hole) of one polygon, and located on an edge of the other polygon

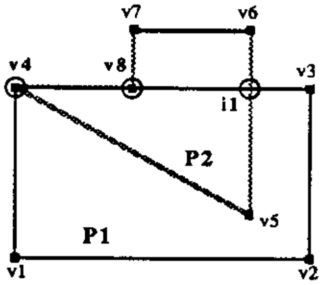
c. vertex not already contained in any boundary (outer or hole) of both polygons, but resulting from the intersection of edges of the two polygons:

Common-vertices of the second or third category (as defined above) must be added respectively to the appropriate boundary of one polygon or of both polygons (see Figure B-4). All common-vertices are tagged.

Note: Finding the common-vertices should be treated with special attention because of the precision problems it raises (vertex on an edge, intersection of edges). The corresponding techniques are not discussed here.

B.1.2 Output

The algorithm returns two sets of 2-D polygons defined as above. One set contains the portions of the subject that are outside the reference, the other set contains the portions of the subject that are inside the reference.



Before intersection:

P1: v1, v2, v3, v4
P2: v4, v5, v6, v7, v8

After intersection:

P1: v1, v2, v3, i1, v8, v4
P2: v4, v5, i1, v6, v7, v8

Figure B-4: Common-vertices

Note: The output of the algorithm consists of 2-D polygons whose format is identical to the format of the input polygons. This feature can be very useful in many applications involving polygons.

B.2 Position of Subject

B.2.1 Classification of the Common-vertices

The following classification describes the relative location of the subject polygon with respect to the reference polygon.

The classification procedure is called during the clipping process to compare a boundary (S) of the subject to a boundary (R) of the reference. It classifies each vertex (V) common to S and R according to the position relatively to R of the two edges of S connected to V .

On S, the vertex that precedes V and the vertex that follows V define the two edges of S connected to V: the *incoming-edge* and the *outgoing-edge*. Both edges are classified according to whether they are inside R (INSIDE), outside R (OUTSIDE), or on an edge of R (PARALLEL). Figure B-5 shows all the possible classifications.

The test PARALLEL/INSIDE/OUTSIDE for an edge can be performed by first checking if the end-vertex of the edge (other than V) belongs to R (PARALLEL), and then for instance by checking the inclusion of the middle-vertex of the edge with respect to R (INSIDE or OUTSIDE).

Note: The test to determine whether a vertex is inside or outside a polygon is not discussed here.

Note:Figure B-6 presents all all the possible geometric configurations at a common-vertex, and the corresponding action to take.

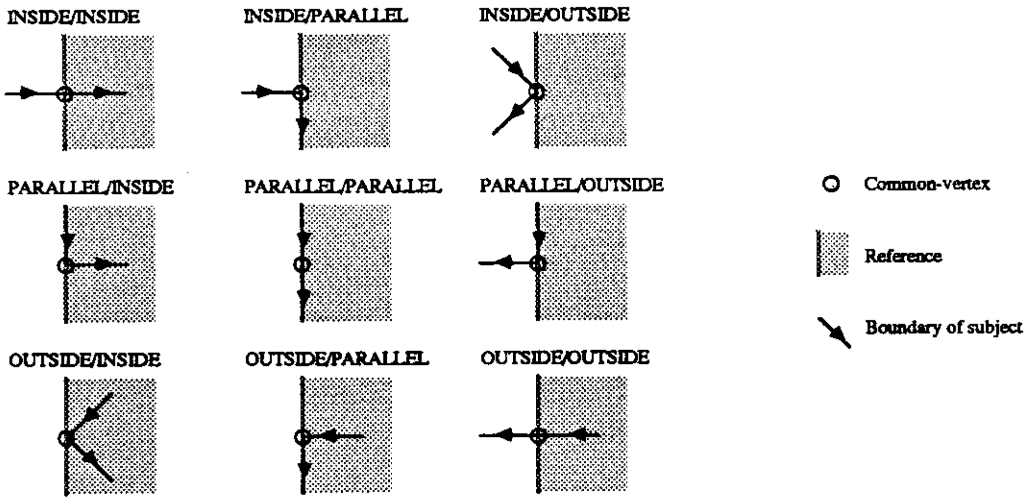


Figure B-5: Common-vertices classification

B.2.2 Relations between boundaries

Definitions

A boundary B is inside another boundary B' if it lays inside B'. B and B' may have vertices and even edges in common.

B is **inside* B' if B is inside B', and B and B' have no edge in common (but may have vertices in common).

Determination of the relation between two boundaries

We assume that S is a boundary (of the subject polygon) and R another boundary (of the reference polygon). The common-vertices of S and R are known and classified with respect to R (as described above).

- **S and R have no common-vertices:**

  Since the two boundaries do not intersect, they are in one of the following cases:

  - S is inside R
  - R is inside S
  - S and R do not overlap

  To test the inclusion of a boundary in the other, just check the inclusion of one vertex.

- **S and R have common-vertices:**

  The common-vertices are classified with respect to R. The relative position of the

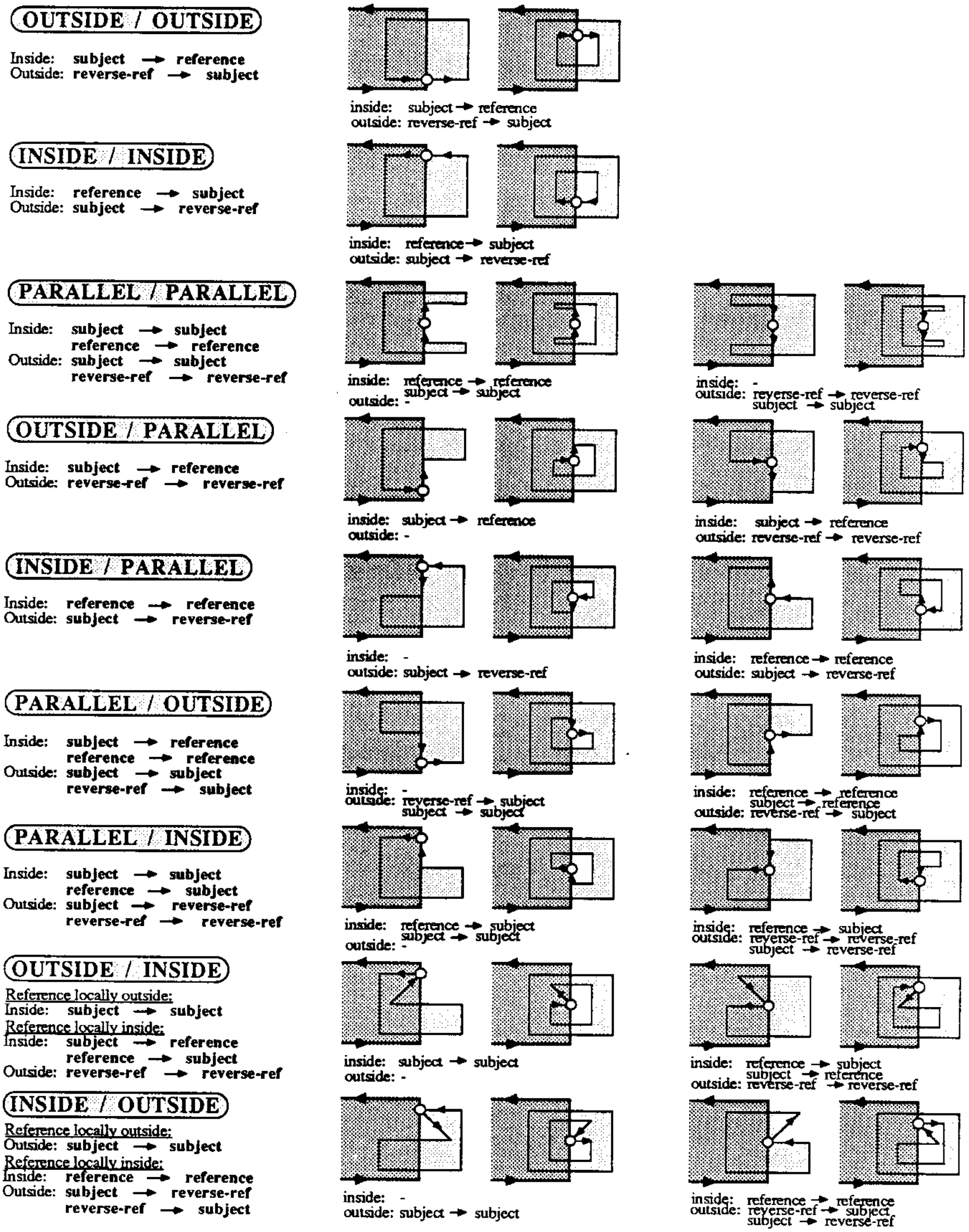

OUTSIDE / OUTSIDE
Inside: subject → reference
Outside: reverse-ref → subject
inside: subject → reference
outside: reverse-ref → subject
INSIDE / INSIDE
Inside: reference → subject
Outside: subject → reverse-ref
inside: reference → subject
outside: subject → reverse-ref
PARALLEL / PARALLEL
Inside: subject → subject
reference → reference
Outside: subject → subject
reverse-ref → reverse-ref
inside: reference → reference
subject → subject
outside: -
inside: -
outside: reverse-ref → reverse-ref
subject → subject
OUTSIDE / PARALLEL
Inside: subject → reference
Outside: reverse-ref → reverse-ref
inside: subject → reference
outside: -
inside: subject → reference
outside: reverse-ref → reverse-ref
INSIDE / PARALLEL
Inside: reference → reference
Outside: subject → reverse-ref
inside: -
outside: subject → reverse-ref
inside: reference → reference
outside: subject → reverse-ref
PARALLEL / OUTSIDE
Inside: subject → reference
reference → reference
Outside: subject → subject
reverse-ref → subject
inside: -
outside: reverse-ref → subject
subject → subject
inside: reference → reference
subject → reference
outside: reverse-ref → subject
PARALLEL / INSIDE
Inside: subject → subject
reference → subject
Outside: subject → reverse-ref
reverse-ref → reverse-ref
inside: reference → subject
subject → subject
outside: -
inside: reference → subject
outside: reverse-ref → reverse-ref
subject → reverse-ref
OUTSIDE / INSIDE
Reference locally outside:
Inside: subject → subject
Reference locally inside:
Inside: subject → reference
reference → subject
Outside: reverse-ref → reverse-ref
inside: subject → subject
outside: -
inside: reference → subject
subject → reference
outside: reverse-ref → reverse-ref
INSIDE / OUTSIDE
Reference locally outside:
Outside: subject → subject
Reference locally inside:
Inside: reference → reference
Outside: subject → reverse-ref
reverse-ref → subject
inside: -
outside: subject → subject
inside: reference → reference
outside: reverse-ref → subject
subject → reverse-ref

Figure B-6: Which boundary to follow, depending on the vertex classification.

two boundaries is one of the following:

- S and R coincide
  This is the case when all outgoing-edges are PARALLEL.

- S inside R
  This is the case when there is no OUTSIDE outgoing-edge.

- S *inside R
  This is the case when there is no OUTSIDE outgoing-edge and no PARALLEL outgoing-edge.

- R inside S
  This is the case when there is no INSIDE outgoing-edge, and one vertex of R is inside S (if all vertices of R are common-vertices, check for instance the inclusion in S of the middle-point of an edge of R that is not common to R and S).

- R *inside S (R inside S, and R and S have no edge in common)
  This is the case when there is no INSIDE outgoing-edge, no PARALLEL outgoing-edge, and one vertex of R is inside S (if all vertices of R are common-vertices, check for instance the inclusion in S of the middle-point of an edge of R).

- S and R do not overlap
  This is the case when there is no INSIDE outgoing-edge, and one vertex of R is outside S (if all vertices of R are common-vertices, check for instance the non-inclusion in S of the middle-point of an edge of R that is not common to R and S).

- S and R do not overlap and have no edge in common
  This is the case when there is no INSIDE outgoing-edge, no PARALLEL outgoing-edge, and one vertex of R is outside S (if all vertices of R are common-vertices, check for instance the non-inclusion in S of the middle-point of an edge of R).

- S and R overlap
  This is the case when there is at least one INSIDE outgoing-edge and one outside outgoing-edge.

B.3 The reference polygon has one or several holes

The clipping of a subject polygon by a reference polygon with holes in it is performed by iteratively applying a clipping operation that uses a reference polygon with no holes in it. This method lies on the simple idea that the inside of a hole of a polygon is outside the polygon. Therefore the inside polygons obtained by clipping the subject by a hole of the reference considered as a regular polygon (*i.e.* the order of its vertices has been reversed to become counter-clockwise) are final outside polygons. Also, the outside polygons resulting from such a clipping operation can be successively clipped by the other holes of the reference, and eventually by the outer-boundary of the reference.

This step can be summarized as follows (see Figure B-7):

If the reference has one or several holes, clip iteratively the subject, successively using as:

- subject: first the original subject polygon, then the outside polygons returned by the previous clipping operation.
- reference: the hole boundaries (ordered in a counter-clockwise way) of the original reference polygon, and finally the outer-boundary of the original reference polygon.

Any polygon inside a hole of the reference or outside the outer-boundary of the reference is returned as a final outside polygon. Any polygon inside the outer-boundary of the reference (last iteration) is returned as a final inside polygon.
If the clipping by one hole does not return any outside polygon, then the whole original subject polygon is inside the hole, and therefore is entirely outside the reference. No further operation is necessary.

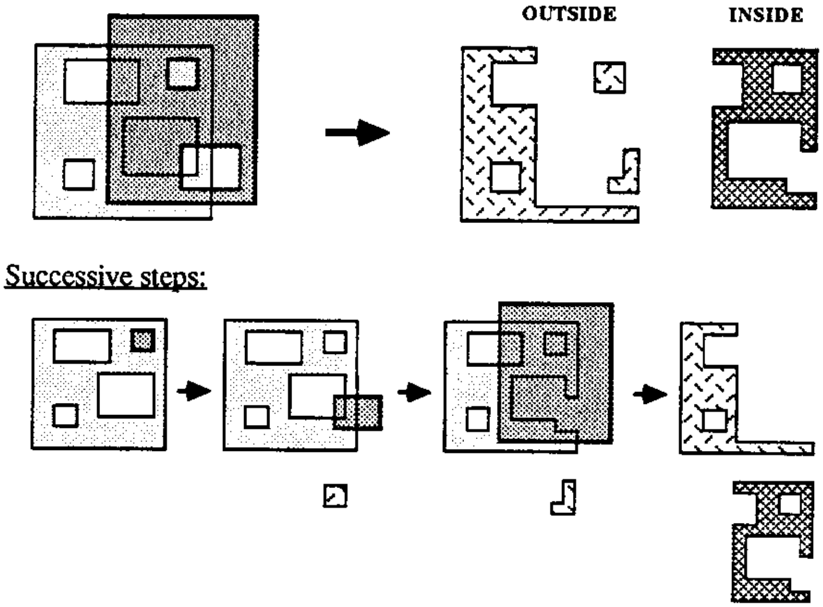


Figure B-7: Holes in the reference polygon

The following steps describe how to clip a subject polygon by a reference polygon that has no holes in it.

The method first handles simple cases that do not require any subdivision of polygons. For those cases, the resulting inside and outside polygons are obtained directly from the subject and the reference polygons by copying and combining the boundaries of the original polygons, as indicated on the corresponding figures.

Later cases require to subdivide polygons, using the actual clipping method explained in section B.5, or to make the union of polygons, as explained in section B.7.

B.4 Possible cases

B.4.1 The reference and the subject have no common-vertices

The subject is inside the reference

There is no outside part, and the inside part is the subject polygon (Figure B-8).

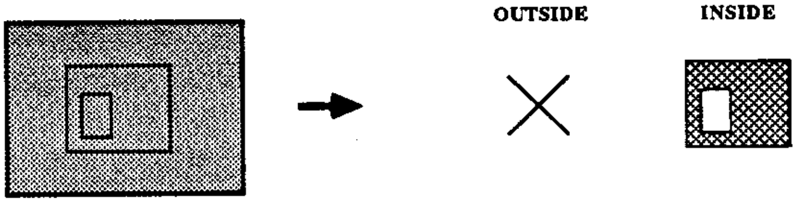


Figure B-8: Subject inside reference

The reference is inside the subject

The reference is inside a hole of the subject

The outside part is the subject polygon, and there is no inside part (Figure B-9).

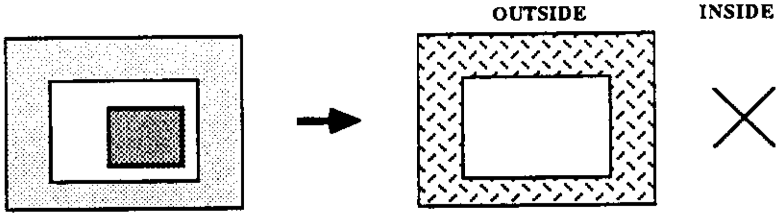


Figure B-9: Reference inside hole of subject

The reference is not inside a hole of the subject

The outside part has for outer-boundary the outer-boundary of the subject, and for holes the reference (after reversing the order of its vertices) and the holes of the subject that are not inside the reference.

The inside part has for outer-boundary the boundary of the reference, and for holes the holes of the subject that are inside the reference (Figure B-10).

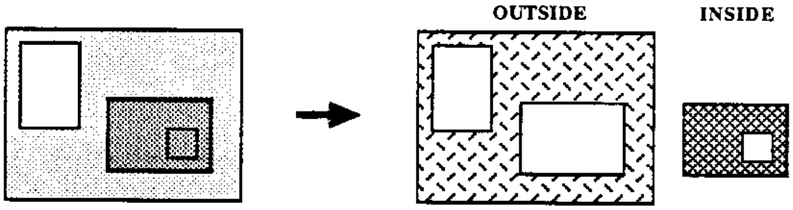


Figure B-10: Reference inside subject

The reference and the subject do not overlap The outside part is the subject polygon, and there is no inside part (Figure B-11).

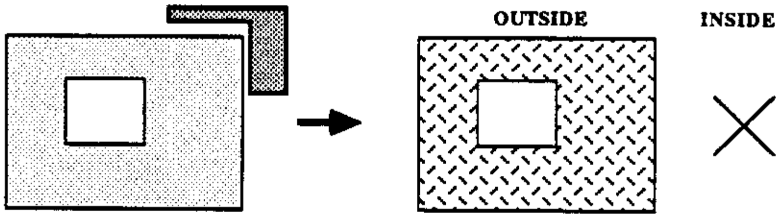


Figure B-11: No overlap

B.4.2 The reference is *inside the subject

This corresponds to one one the two following cases (see page 42):

1. None of the common-vertices belongs to the outer-boundary of the subject. Therefore the reference is inside the outer-boundary of the subject and intersects with at least one hole of the subject.
2. At least one common-vertex belongs to the outer-boundary of the subject, and the reference is inside the outer-boundary of the subject, but has no edge in common with it.

The reference is inside or coincides with a hole of the subject The outside part is the subject polygon, and there is no inside part (Figure B-12).

General case This case requires the generation of new boundaries (Figure B-13).

The outside part is obtained by making the union of the reference with the holes of the subject (see section B.7). A new hole-boundary is generated by this process, and possibly some other outer-boundaries inside the new hole. These new outer-boundaries must then be clipped by the holes of the subject (see section B.6).

The inside part is obtained as the resulting outside part after clipping the reference by the

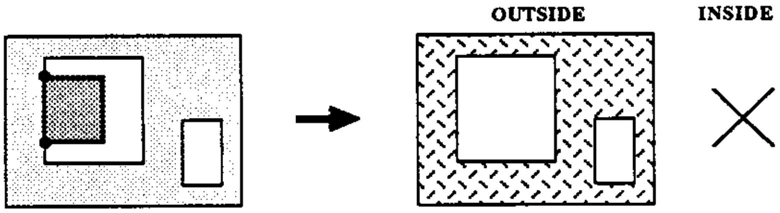


Figure B-12: Reference inside hole of subject

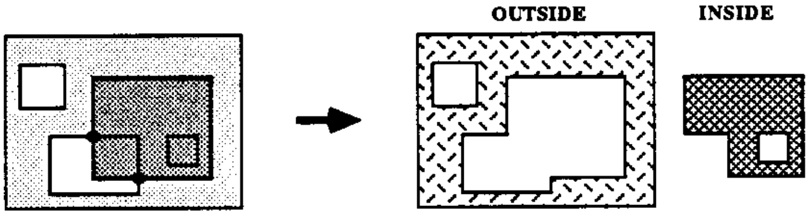


Figure B-13: General case

holes of the subject (see section B.6).

B.4.3 The reference is not *inside the subject

The outer-boundary of the subject is inside or coincides with the reference

There is no outside part, and the inside part is the subject polygon (Figure B-14).

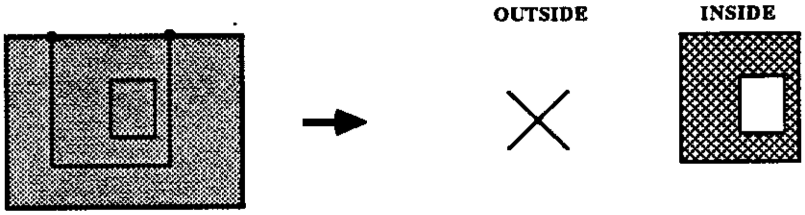


Figure B-14: Subject inside reference

The subject and the reference do not overlap

The outside part is the subject polygon, and there is no inside part (Figure B-15).

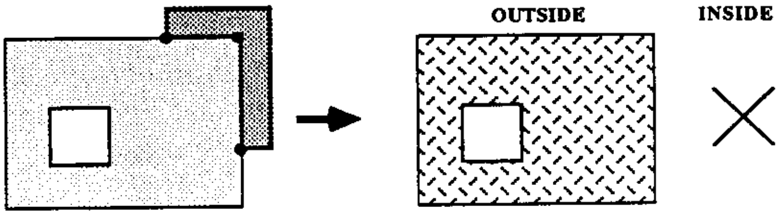


Figure B-15: No overlap

The subject and the reference overlap

This case requires the generation of new boundaries (Figure B-16).

The outside part is obtained after a two-step process:

1. Clipping (outside part) of the outer-boundary of the subject by the reference, using the algorithm described in section B.5.
2. Clipping of the polygons resulting from the previous operation by the holes of the subject (see section B.6).

Two different cases apply for the generation of the inside part:

- *The reference is inside the subject* (and has at least one edge in common with it) In this case, the inside part is obtained by clipping the reference by the holes of the subject (see section B.6).
- *The reference is not inside the subject* In this case, the inside part is generated in a similar way as the outside part:
    1. Clipping (inside part) of the outer-boundary of the subject by the reference, using the algorithm described in section B.5.
    2. Clipping of the polygons resulting from the previous operation by the holes of the subject (see section B.6).

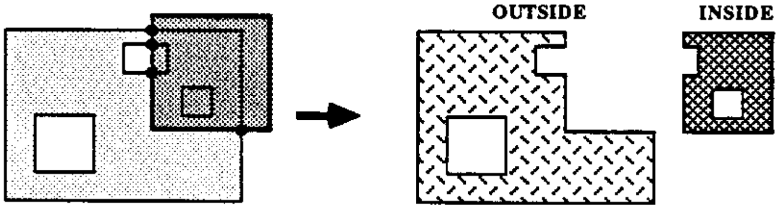


Figure B-16: Subject and reference overlap

Figure B-17 summarizes all the different cases described in this section. The nine shaded boxes indicate the final output for each of the nine cases.

The subject polygon has an outer-boundary S and n holes H_1, H_2, ...H_n (which is expressed as S [H_1..H_n]).
The reference polygon has an outer-boundary R and no hole.

R' designates a copy of the reference polygon, but whose order of vertices has been reversed (*i.e.* R' is a hole that has the same vertices as R).
h designates the new hole-boundary obtained by making the union of the reference with the holes of the subject.

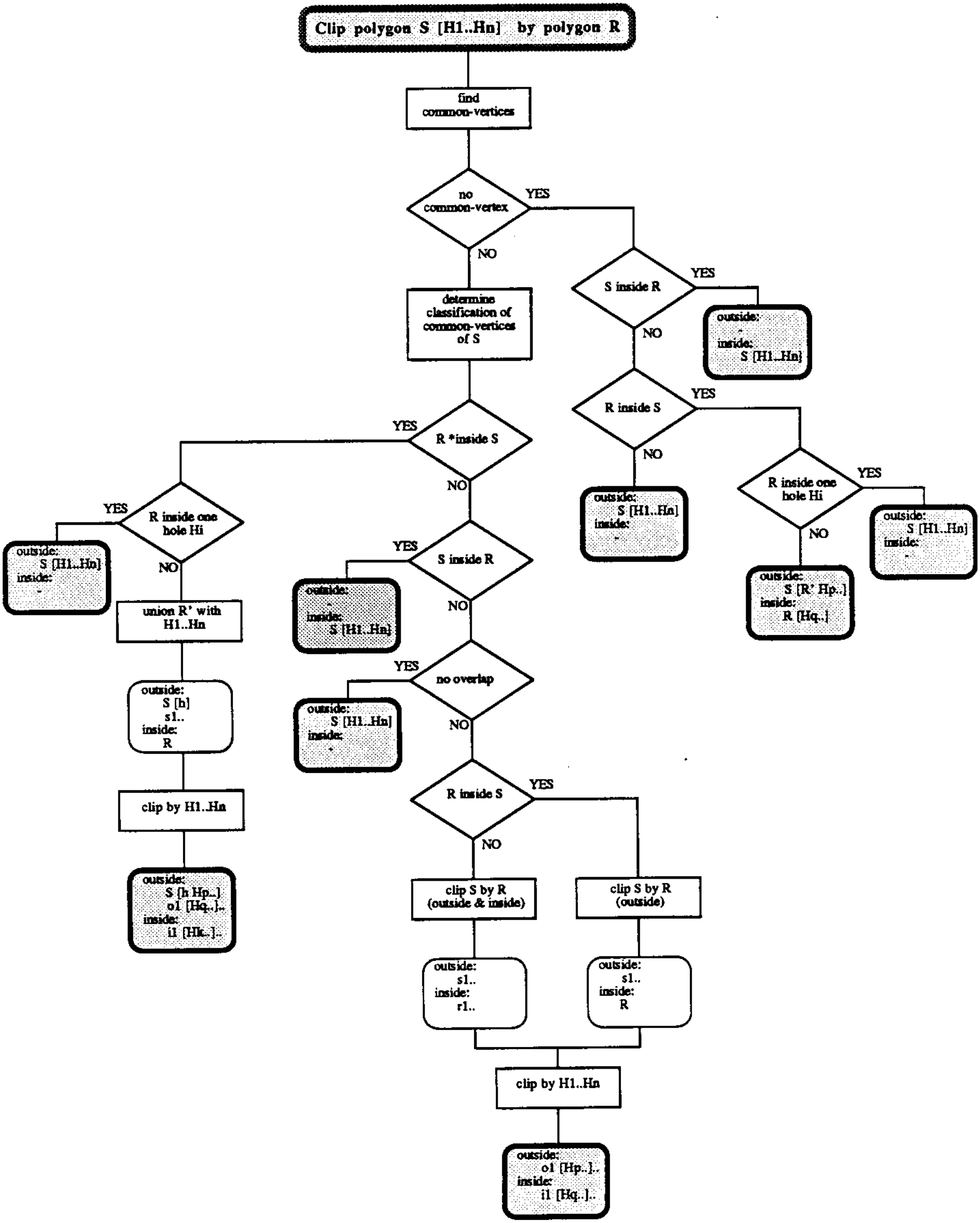

Clip polygon S [H1..Hn] by polygon R
find common-vertices
no common-vertex
YES
NO
S inside R
YES
outside: - inside: S [H1..Hn]
NO
R inside S
YES
NO
outside: S [H1..Hn] inside: -
R inside one hole Hi
YES
outside: S [H1..Hn] inside: -
NO
outside: S [R' Hp..] inside: R [Hq..]
determine classification of common-vertices of S
R *inside S
YES
NO
R inside one hole Hi
YES
outside: S [H1..Hn] inside: -
NO
union R' with H1..Hn
outside: S [h] s1.. inside: R
clip by H1..Hn
outside: S [h Hp..] o1 [Hq..].. inside: i1 [Hk..]..
S inside R
YES
outside: - inside: S [H1..Hn]
NO
no overlap
YES
outside: S [H1..Hn] inside: -
NO
R inside S
YES
NO
clip S by R (outside & inside)
clip S by R (outside)
outside: s1.. inside: r1..
outside: s1.. inside: R
clip by H1..Hn
outside: o1 [Hp..].. inside: i1 [Hq..]..

Figure B-17: Clipping process

B.5 Clipping of overlapping polygons

The case of overlapping polygons requires a subdivision of the subject polygon into outside and inside parts. The subject polygon and the reference polygon considered here have no hole. Their common-vertices are known, and classified with respect to the reference.

The basic idea of the clipping method described here is to generate new boundaries by starting at a common-vertex, then by following the boundaries of the polygons, possibly switching between subject and reference when a common-vertex is encountered.

At any given time, the polygon that is being followed is known as the *current-polygon*. It can be the subject, the reference, or the reference in reverse order (the vertex list is followed in reverse order).

B.5.1 Clipping algorithm

The following steps describe how to obtain the portions of the subject polygon that are outside the reference polygon. The same method is used to generate the portions of the subject polygon that are inside the reference polygon (replace "OUTSIDE" by "INSIDE" in the following operations).

1. Determine the list of vertices whose outgoing-edge is OUTSIDE (called *OUTSIDE-vertices*). This list contains the vertices that can be used as *starting-vertex* of a new boundary.

2. Choose one OUTSIDE-vertex as starting-vertex (for instance the first of the list), and remove it from the list. Store the starting-vertex in a new list that will eventually contain the new boundary.

3. The current-polygon is the subject polygon, and the current-vertex is the starting-vertex.

4. Starting from the current-vertex, follow the current-polygon until the next common-vertex is reached. Add every traversed vertex to the new boundary.

5. When a common-vertex is reached, determine the appropriate operations to perform according to the classification of the vertex, using the OUTSIDE rules listed below. The current common-vertex becomes the new current-vertex, is removed from the list of OUTSIDE-vertices (except if the rules specifies to keep it), and is added to the new boundary. The selected rule determines the new current-polygon.

6. Repeat steps 4 to 5 until the starting-vertex-has been reached. At this point, a new boundary has just been closed.

7. Repeat steps 2 to 6 until the list of OUTSIDE-vertices is exhausted. At this point, the generation of new boundaries is complete.

B.5.2 Rules to determine which polygon to follow

The new current-polygon that must be followed after reaching a common-vertex is designated as SUBJECT (boundary of the subject polygon), REFERENCE (boundary of the reference polygon), or REVERSE-REF (boundary of the reference polygon, followed in reverse order).

The rule giving the new current-polygon is determined by the classification of the current-vertex. In some cases, an additional test is required to determine, near the current-vertex, the local position of the boundary of the reference relatively to the subject (test for instance whether the middle-point of one of the two edges of the reference connected to the current-vertex is inside or outside the subject).

The following rules list only the cases that require to switch from one boundary to another. The current-polygon does not change in all other cases. Figure B-6 illustrates all the possible geometric configurations.

Rules for generating an OUTSIDE boundary

current-vertex	**new current-polygon**
INSIDE/INSIDE	REVERSE-REF
OUTSIDE/OUTSIDE	SUBJECT
INSIDE/PARALLEL	REVERSE-REF
PARALLEL/INSIDE	REVERSE-REF
PARALLEL/OUTSIDE	SUBJECT
INSIDE/OUTSIDE Reference locally inside	• if current-polygon is REVERSE-REF, jump to SUBJECT. • if current-polygon is SUBJECT, jump to REVERSE-REF and do not remove the vertex from the OUTSIDE-vertices list.

B.5.2.1 Rules for generating an INSIDE boundary

current-vertex	**new current-polygon**
OUTSIDE/OUTSIDE	REFERENCE
INSIDE/INSIDE	SUBJECT
OUTSIDE/PARALLEL	REFERENCE
PARALLEL/OUTSIDE	REFERENCE
PARALLEL/INSIDE	SUBJECT
OUTSIDE/INSIDE Reference locally inside	• if current-polygon is REFERENCE, jump to SUBJECT. • if current-polygon is SUBJECT, jump to REFERENCE and do not remove the vertex from the INSIDE-vertices list.

B.5.3 Example

Figure B-18 shows an example of clipping for two overlapping polygons.

B.6 Clipping by holes

In case that the subject and the reference overlap (page 49), the clipping of the subject by the reference (section B.5) is performed without taking into account the possible holes of the subject. The boundaries generated by the clipping process must then be clipped (outside) by the holes of the subject.

This for also true for the general case of page 47, where the reference and the boundaries (except the new hole) generated during the iterative union operations (section B.7) must be clipped (outside) by the holes of the subject.

B.6.1 Iterative clipping by the holes of the subject

To clip a boundary by a set of holes, first clip (outside) the initial boundary by one of the holes, then, for all remaining holes, clip (outside) the outer-boundary resulting from the previous clipping operation by the next hole (see Figure B-19).

The clipping of a boundary by a hole does not require to examine all the cases listed in section B.4, since here the subject does not have any hole. Only the following cases are possible:

1. No overlap (with or without common-vertex): the resulting polygon has the boundary as outer-boundary, and no hole. boundary remains unchanged
2. The hole is *inside the boundary (with or without common-vertex): the resulting polygon has the boundary as outer-boundary, and the hole as hole.
3. Otherwise: the resulting polygon has as outer-boundary the boundary generated

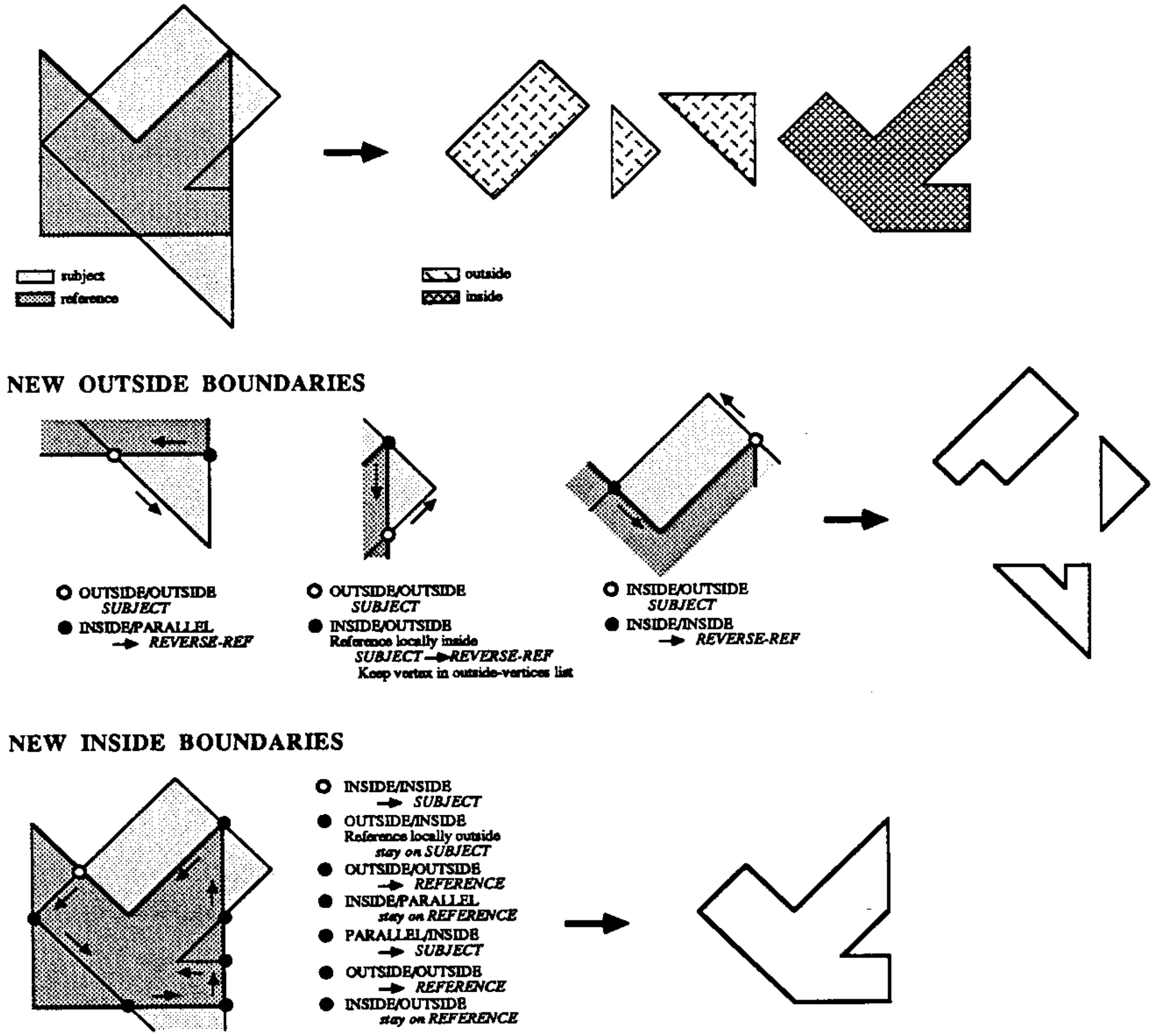


Figure B-18: Clipping of overlapping polygons

by applying the algorithm described in section B.5, and no hole.

Note: Before clipping a boundary by a hole, the order of the vertices of the hole should be reversed. The order should be reset to a clockwise order after completion of the clipping operation.

B.6.2 Example

Figure B-19 presents an example of clipping in the case B.4.3. First, the outer-boundary of the subject is clipped by the reference. The resulting outside polygon and inside polygon are then clipped (outside) by the holes of the subject.

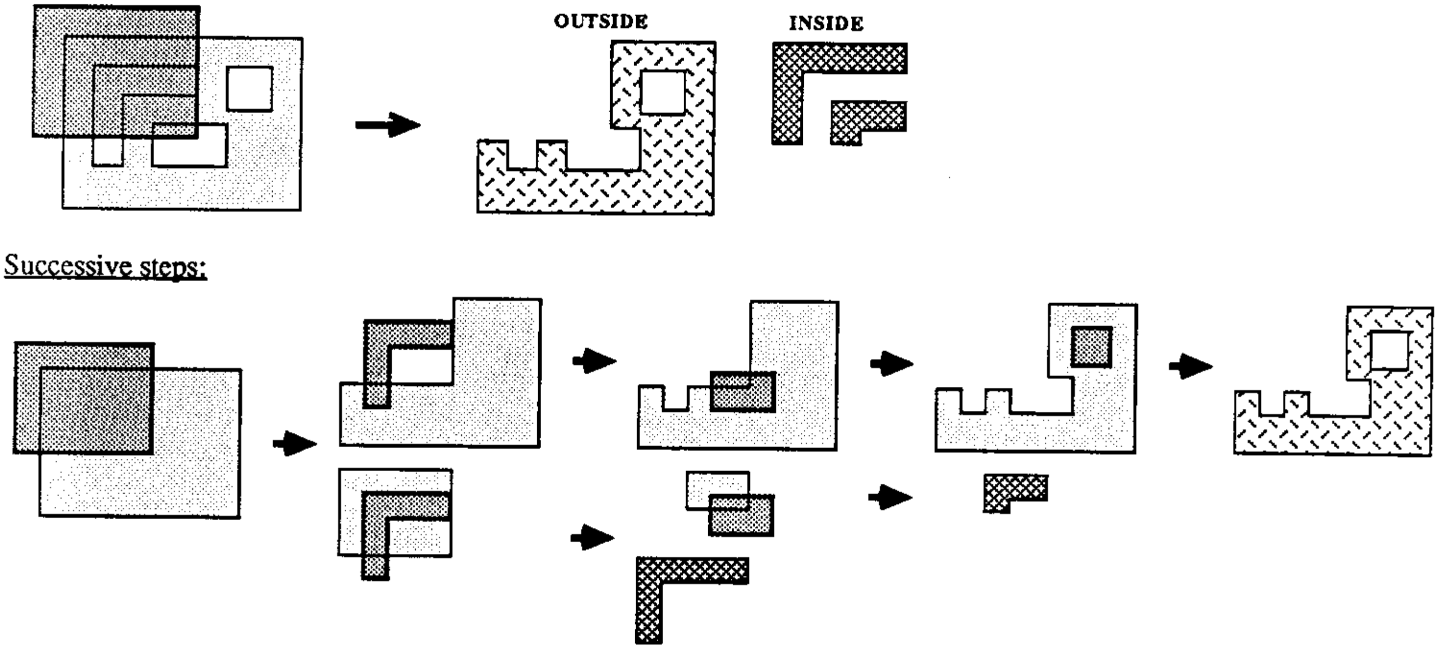


Figure B-19: Iterative clipping

B.7 Union with holes

This section explains how to generate the outside portions resulting from the clipping of the subject by the reference, in the general case in page 47.

In this case, the resulting outside part consists in:

- One polygon, whose outer-boundary is the outer-boundary of the subject, and whose hole-boundaries are the holes of the subject that are outside the reference and that do not have any edge in common with the reference, plus a new hole which is the **union** of the reference and of all the holes of the subject that are not inside the reference and that overlap or have at least one edge in common with the reference.
- Possibly one or several polygons that are inside the new hole defined above.

B.7.1 Iterative union of holes

Let us suppose that the subject has an outer-boundary S and n holes H_1, H_2, ... H_n, and that the reference has an outer-boundary R and no hole. In the case mentioned above (R *inside S), we need to generate the union of R with H_1, H_2, ... H_n.

A way to perform this union is to start by making the union of R and H_1, then to make the union of the outer-boundary resulting from the previous union operation with H_2, and to repeat this process until H_n.

The outer-boundary obtained after the last union operation is the new hole-boundary of the outside polygon that has the outer-boundary of the subject as outer-boundary. Other hole-boundaries may be added when clipping by the holes of the subject (section B.6).

All the hole boundaries obtained in any of the successive union operations define the outside parts of the subject that are inside the new hole. They still need to be clipped by the holes of the subject (see section B.6).

Note: Because of the input requirements of the union procedure described below, it is necessary to reverse the order the vertices of R and of H_1, H_2, ... H_n before performing the union operations. Do not forget to reset H_1, H_2, ... H_n to a clockwise order after the union operations are completed.

B.7.2 Union of two polygons with no holes

The union of two polygons with no holes is performed by using the clipping algorithm described in section B.5, applied with a subject polygon ordered in a clockwise way (like a hole). The outside part resulting from the actual clipping of a subject polygon considered as a hole (ordered in a clockwise way) by a reference polygon is the union of the subject and the reference.

The rules applied to switch forth and back between subject and reference (page 52) are the same as for a regular clipping operation, except for the one corresponding to the case INSIDE/OUTSIDE. For a union operation this rule applies only if the reference is locally outside of the subject (instead of inside)

The resulting polygon may have zero, one or several holes (see Figures B-20 and B-21). Since the clipping algorithm is applied with a subject ordered in a clockwise way (like a hole), the order of the boundaries of the resulting polygon is reversed (*i.e.* the order of the outer-boundary is clockwise, and the order of the hole boundaries is counter-clockwise).

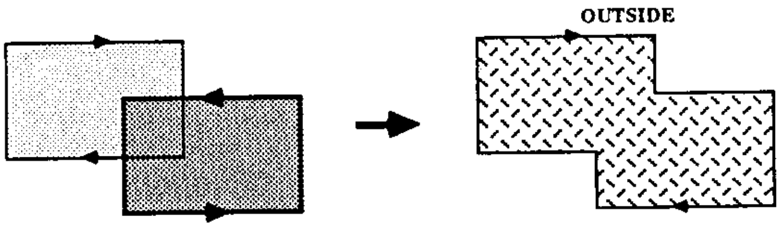


Figure B-20: Union of 2 polygons (no hole generated)

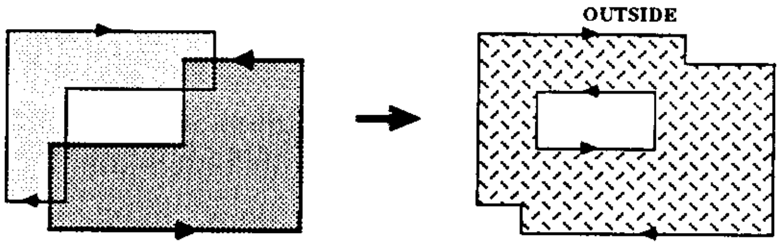


Figure B-21: Union of 2 polygons (hole generated)

B.7.3 Example

Figure B-22 presents an example of clipping that requires to make the union of the reference with the holes of the subject (case B.4.2).

The subject has two holes, and the reference is *inside the subject.
The resulting outside part is composed of three polygons.

First, we generate the union of the reference with one hole of the subject. The resulting polygon has two hole-boundaries. The outer-boundary obtained after the first union operation is then used for the next union operation, with the second hole of the reference. The polygon resulting from the union has no hole. Its outer-boundary is the boundary of the new hole. Two hole-boundaries were generated during the iterative union operations. They will give the two outside polygons that are inside the new hole, after they get clipped by the holes of the subject.

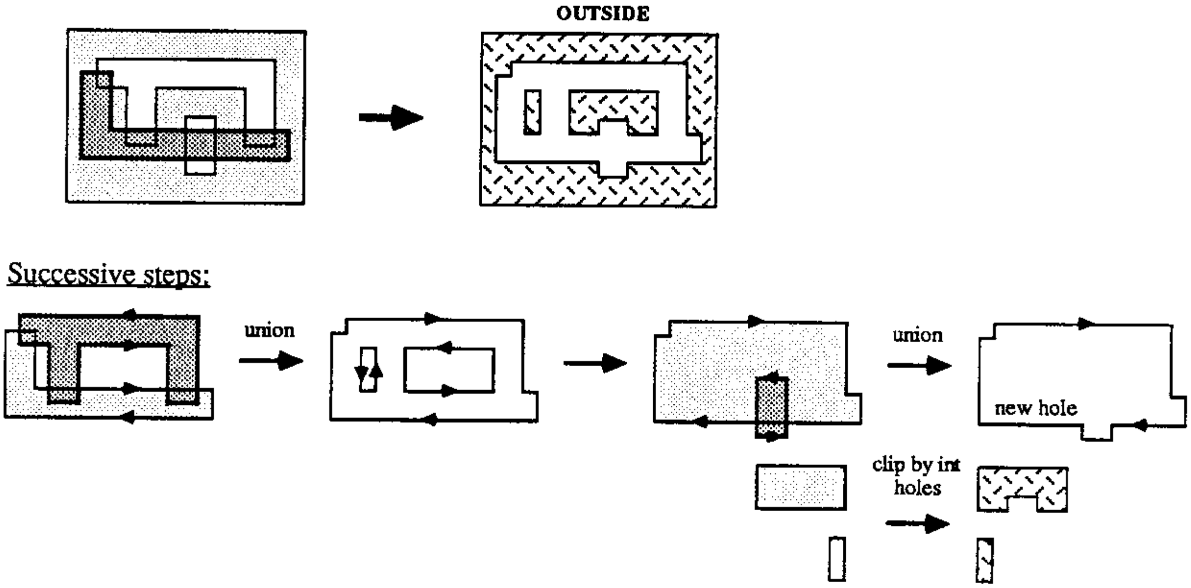


Figure B-22: Iterative union

References

[1] P. Balakumar, J.C. Robert, R. Hoffman, K. Ikeuchi, and T. Kanade. *VANTAGE: A Frame-Based Geometric/Sensor Modeling System – Programmer/User's Manual V1.0.* Technical Report, Carnegie Mellon University, Robotics Institute, 1989. (In preparation).

[2] B.G. Baumgart. *Winged edge polyhedron representation.* Technical Report STAN-CS-320, Stanford University, Artificial Intelligence Laboratory, 1972.

[3] P.J. Besl. Geometric modeling and computer vision. *Proc. of the IEEE,* 76(8):936–958, August 1988.

[4] P. Bezier. *Numerical control: mathematics and applications.* John Wiley, New York, 1972.

[5] R.C. Bolles and P. Horaud. 3DPO: a three-dimensional part orientation system. *The International Journal of Robotics Research,* 5(3):3–26, 1986.

[6] J.W. Boyes and J.E. Gilchrist. GMSolid: interactive modeling for design and analysis of solids. *IEEE Journal of Computer Graphics and Applications,* 2(2):27–40, March 1982.

[7] C. Hansen and T. Henderson. CAGD-based computer vision. In *Proc. IEEE Computer Society Workshop on Computer Vision,* pages 100–105, IEEE Computer Society, Miami Beach, FL, December 1987.

[8] K. Ikeuchi. Generating an interpretation tree from a CAD model for 3-D object recognition in bin-picking tasks. *International Journal of Computer Vision,* 1(2):145–165, 1987.

[9] K. Ikeuchi and K. S. Hong. *Determining Linear Shape Change: Toward Automatic Generation of Object Recognition Program.* Technical Report CMU-CS-88-188, Carnegie Mellon University, Computer Science Department, 1988.

[10] K. Ikeuchi and T. Kanade. Modeling sensors: toward automatic generation of object recognition program. *Computer Vision, Graphics, and Image Processing,* 1989. (Accepted for publication).

[11] T. Kanade, P. Balakumar, J.C. Robert, R. Hoffman, and K. Ikeuchi. Overview of geometric/sensor modeler VANTAGE. In *Proc. the International Symposium and Exposition on Robots*, The Australian Robot Association, Sydney, Australia, November 1988.

[12] F. Kimura and M. Hosaka. *Program Package GEOMAP Reference Manual.* Computer Vision Section, Electrotechnical Lab., 1977.

[13] K. Koshikawa. *SOLVER reference manual.* Computer Vision Section, Electrotechnical Lab., RM-85-33J edition, 1984. (In Japanese).

[14] K. Koshikawa and Y. Shirai. A 3-D modeler for vision research. In *Proc. Intern. Conf. on Advanced Robot (ICAR85)*, pages 185–190, Robotics Society of Japan, 1985.

[15] T. Lozano-Perez. Task planning. In M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Perez, and M.T. Mason, editors, *Robot motion*, chapter 9, pages 473–498, The MIT Press, Cambridge, MA, 1982.

[16] M.E. Mortenson. *Geometric Modeling*. John Wiley, 1985.

[17] A. A. G. Requicha. Representations for rigid solids: theory, methods, and systems. *ACM Computing Surveys*, 12(4):437–464, December 1980.

[18] A.A.G. Requicha and H.B. Voelcker. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 9–24, march 1982.

[19] J.C. Robert. *Polygon Clipping: Description of a General Algorithm.* Technical Report, Carnegie Mellon University, Robotics Institute, 1989. in preparation.

[20] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Tipplett, editor, *Optical and Electro-Optical Information Processing*, pages 159–197, MIT Press, Cambridge, MA, 1965.

[21] A. Storr and J.F. McWaters. *Off-line programming of industrial robots.* North-Holland, Amsterdam, 1986.

[22] I.E. Sutherland. Sketchpad: a man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference: 23*, pages 329–349, 1963.

[23] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. In *Proc. of SIGGRAPH*, 1977.