National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Canada

THREE-DIMENSIONAL MODEL CONSTRUCTION FROM MULTIPLE SENSOR VIEWPOINTS

Stéphane Aubry

B.Sc. Northeastern University 1984

M.Sc. Stanford University 1985

Department of Electrical Engineering
McGill University, Montréal

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

February 1994

ISBN 0-315-94575-3

Canada

A mes parents

Abstract

This work addresses the automatic construction of geometric models of real scenes, from multi-view three-dimensional sensor data.

We review range data acquisition, multi-view integration, and solid modeling. We show that knowledge about the data acquisition procedure yields not only the coordinates of the acquired points, but also additional geometrical information. We use that information to draw a geodesic proximity graph with respect to the surface the data points lie on. Such a graph is useful because it provides the connectivity information necessary for subsequent differential-geometric processing, and three-dimensional surface modeling. We say that the graph is a least-commitment boundary representation, because it does not involve the use of higher level or cognitive processes.

We specialize the concept to the common case of line-of-sight optical sensors. We give formal definitions of graph validity using assumptions of object opacity and object rigidity, and we demonstrate that Euclidean proximity graphs drawn on the data points are not valid when the data is sparse with respect to surface concavities.

We describe a sub-quadratic incremental view integration algorithm that assumes the data is highly-organized. It guarantees graph validity under restrictive conditions. Therefore, we present another non-incremental algorithm with no assumption on the input data organization. It is based on an iterative carving of the graph faces, starting

with the convex hull of all points as the initial model. It builds a hierarchy of models, each of which is internal to the previous one. We test the algorithm with real data on an object homeomorphic to a sphere, and incorporate heuristics designed to yield more geometrically pleasing results.

Résumé

Cette thèse a pour sujet la construction automatique de modèles géométriques de scènes réelles à l'aide de capteurs télémétriques.

Nous faisons une étude bibliographique de l'acquisition de données et de la modélisation tri-dimensionnelles, et de l'intégration de vues multiples. Nous montrons que la connaissance des détails de la procédure d'acquisition offre non seulement les coordonnées des points de surface, mais aussi de l'information géométrique supplémentaire. Nous utilisons cette information pour tracer un graphe de proximité géodésique par rapport à la surface sur laquelle les données sont échantillonnées. Ce graphe est utile car il offre la connexité nécessaire au traitement différentiel subséquent, et enfin à une représentation surfacique tri-dimensionnelle. Nous disons que le graphe est une représentation de surface à engagement minimum, étant donné qu'il ne requiert pas l'usage de processus cognitifs ou de haut niveau.

Nous spécialisons ce concept au cas courant des capteurs optiques. Nous offrons des définitions formelles de la validité du graphe en postulant des objects opaques et rigides, et nous prouvons que les graphes euclidiens de proximité ne sont pas valides lorsque la résolution des données est faible par rapport aux concavités de la surface à modéliser.

Nous décrivons un algorithme incrémental d'intégration de données qui présume que

les données sont organisées sous forme matricielle. L'algorithme peut garantir la validité du graphe résultant sous certaines conditions restrictives. En conséquence, nous présentons un autre algorithme, cette fois non incrémental, qui ne présume aucune organisation particulière des données. Cet algorithme sculpte les faces du graphe de façon itérative, avec comme point de départ l'enveloppe convexe de tous les points. Il construit une hiérarchie de modèles, chacun étant strictement inclus dans le précédant. Nous testons l'algorithme sur un object homéomorphique à une sphère et nous incorporons des heuristiques afin d'obtenir des résultats perceptuellement meilleurs.

Acknowledgements

L'achèvement de cette thèse est une étape importante. Son parcours a été long et tumultueux, avec ses peines et ses joies.

Je dois des remerciements chaleureux à trop de personnes pour être en mesure de toutes les nommer. Comme il est de coutume, je m'excuse à l'avance de toute omission qui ne serait que fortuite.

En premier lieu, je tiens à remercier mon directeur de travaux, le Dr. Vincent Hayward, pour son enthousiasme sans borne, et ses encouragements indéfectibles. Grâce à sa créativité communicative, j'ai découvert de nombreux domaines qui me seraient sans doute restés inconnus. Puisse-t-il continuer à transmettre sa passion à des générations d'étudiants!

Je remercie aussi mon comité de thèse pour l'intérêt porté à mes recherches. En particulier, les Dr Levine et Ferrie qui m'ont souvent offert l'occasion de discuter de mes travaux. Je remercie Jacques Labelle et Mike Walsh de l'Université du Québec à Montréal pour avoir pris la peine de confirmer un résultat de combinatoire auquel j'étais arrivé.

L'équipe des "rangers" du McRCIM m'a souvent aidé lors de mes expériences. Mentions spéciales à André Lejeune, Pierre Tremblay, Gilbert Soucy et Wu Kenong. Je

suis aussi reconnaissant à Mike Parker, Peter Whaite, Lee Iverson et Ajit Nilakantan qui se sont à maintes reprises montrés disponibles pour me sortir d'une impasse informatique.

D'autres personnes ont coloré mon séjour au McRCIM. J'espère que nos chemins se croiseront de nouveau. Je pense à Christian Consol, John Darcovich, Guy Godin, Fayçal Kahloun, Robert Lucyshyn, Tareef Al-Mahdawi, Abdul-Reza Mansouri, Chafye Nemri, Mark Readman, Finn Wredenhagen, et mes fidèles compagnons de bureau André Foisy et Yasmine Ghallab.

Enfin et surtout je remercie ma famille proche, pour sa compréhension et son respect constant de mes choix.

Cette thèse a bénéficié du soutien financier du CRSNG (Conseil de recherche en Sciences Naturelles et en Génie), du FCAR (Fonds pour la Formation des Chercheurs et l'Aide à la Recherche) et de la Direction Générale de l'aide financière aux étudiants.

Claim of Originality

We made the following original contributions:

- We reviewed the state-of-the-art in range data acquisition, multi-view integration, and solid modeling and showed the connections between these disciplines.
- We determined that a least-commitment representation that retains all available three-dimensional data is a desirable intermediate representation for multi-view range data integration. We proposed the graph as such a representation.
- We developed a formalism for the validity of a model with respect to its sensing data.
- We derived a formula for the number of labeled 3-connected maximal planar maps.
- We defined conditions for graph validity based on line-of-sight sensing of objects homeomorphic to spheres.
- We developed a sub-quadratic incremental algorithm for highly-structured range view integration, and gave conditions for its validity.
- We developed a non-incremental algorithm for non-structured range view integration and tested it on real data.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 An Automated Scene Description Tool

Many technological applications such as robotics, CAD/CAM, computer graphics, or virtual reality, require geometric descriptions of the surrounding three-dimensional world. Such accurate descriptions are often assumed to be readily available, thus overlooking the difficulty of obtaining them. This thesis is a step towards the goal of automating the task of describing a three-dimensional *environment* or *scene*.

For example, imagine a robot manipulator having to reach a point of its workspace through an environment cluttered with obstacles. Planning a collision-free path requires a map of the environment. Data files such as CAD specifications or other pre-existing prototypes may provide geometrical *models* of some of the environment's features. Yet these existing models may be unusable by the path-planner. They may be inaccurate or of a too low resolution, or their actual pose and attitude in the environment may be undetermined. Hence actual *sensing* of the physical scene is necessary to create or complement these models.

One solution is to manually measure and enter the three-dimensional information into a solid modeler. The task is painstaking, repetitive and prone to error. There are also instances when one cannot even afford the luxury of manual data acquisition. For example, a nuclear reactor is an environment too hazardous for an operator to enter. A small or microscopic environment, or a cavity accessible only through a narrow opening, as in geological exploration, makes it outright impossible for a human to perform the task.

An *Automated Scene Description Tool* (ASDT) is a tool which performs the scene description task automatically or semi-automatically; the latter term means that some operator intervention and/or guidance is necessary. Its purpose is to facilitate, speed up, and improve the reliability of the scene description task.

In the next section, we give examples where the availability of an ASDT is very valuable. We begin with robotics, as it was the driving impetus behind this work.

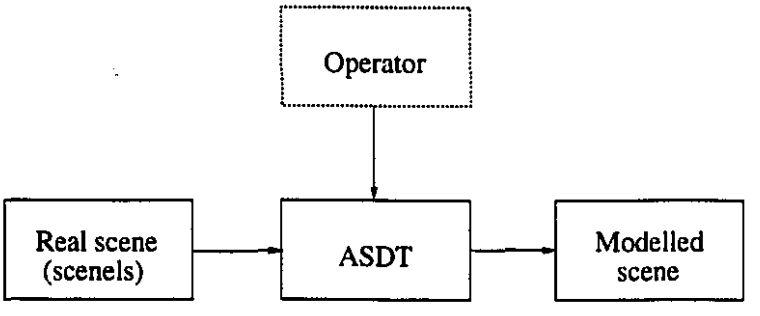

Figure 1.1 Simple ASDT input/output Relationships. Hashed lines indicate optional paths or blocks.

1.1.1 Robotics

1.1.1.1 From Off-line Programming ...

Industrial robots have long been programmed with the help of a "teach pendant", whereby an operator merely guides the robot through a series of moves which are recorded as a history of joint positions. Although simple and effective, this method of teaching-by-showing is increasingly being replaced by off-line programming.

In off-line programming, the task the robot must execute is described in a specialized programming language such as VAL, AML, or RAIL, or with robot-control primitives, as in RCCL (Hayward, 1984). Off-line programming frees up equipment and improves the operator's safety because it separates the robot and the surrounding manufacturing facilities from the learning task. Off-line simulation runs help validate the program and allow specification of optimal parameters. Further, the task can be easily redefined by altering the program. The program itself can be automatically generated from a high level task description.

1.1.1.2 ... To Scene Description

While many techniques contribute to the realization of off-line programming systems, the task of planning the robot motions draws on recent results of artificial intelligence (Whitesides, 1985; Latombe, 1991).

Accurate digital models of the geometry of both the robot and its environment must be available in order to plan the robot path. In particular, the robot must "know" the shape and location of obstacles, in order to avoid colliding with them, and of the parts and fixtures, in order to perform its task.

For a given manufacturing cell, the robot parameters are usually known from manufacturer's data (or they are experimentally determined), and do not change with time. In contrast, the environment may frequently change with new production runs, retooling or rejigging. Because the ASDT decreases the lead-time necessary to describe the newer environment, it is even more beneficial if the environment changes frequently.

So far we avoided the issue of *local* vs. *global planning.* The planning task we refer to is global. It answers the question: "What steps does the robot have to follow in order to accomplish the desired task?" Global planning is a geometrical task and requires a map of the environment. Local planning, in contrast, is used to close the loop inside the global plan, and uses on-line sensor-based feedback information, rather than off-line geometrical models. We illustrate this with some examples.

In a welding application, the welding head is to follow a weld line while maintaining a critical distance. This constraint can be met through local feedback with the aid of various process parameters. However, local on-line information does not address the problem of locating the weld-line, and of maneuvering the robot through the task without collisions. Such global planning tasks require a global model of the environment.

Robotic painting is another important robotic application, particularly for the car manufacturing industry. As in the welding application, fine-tuning the end-effector's distance from the painted part can be done with sensor feedback, but gross motion still requires a map of the environment.

Assembly is also a prevalent robotic task. Much research goes on in the area of fine motion planning for assembly tasks (Lozano-Perez et al., 1984; Mason, 1984; De Schutter and Brussel, 1988). The prevalence of jigs and fixtures in the scene often

makes the environment quite complex.

In all of these applications, an ASDT is a very valuable tool for global motion planning.

1.1.2 Other Applications

An ASDT can also be used in a host of other fields unrelated to robotics, whenever a three-dimensional description is to be obtained from a physical model.

The medical field offers many such applications. For example, an important task is the modeling of the human body, or parts of the human body, as in tomography, dentistry, or prosthesis design.

Other applications requiring the one-time automated acquisition of three-dimensional models are as diverse as geological exploration, nuclear reactor maintenance, archiving, industrial process control, or baggage handling.

1.2 The Components of an ASDT

We call *scenels* the component elements of the scene [1]. The definition of a scenel is very loose, the only restriction being that a scene be comprised of at least one scenel. A scenel may be a physical object, such as a table, or a screwdriver. Alternately, a scenel can be any easily identifiable part of an object, such as a tabletop, or a screwdriver shaft. Yet still, a scenel can be a combination of objects, grouped by virtue of their physical attachment, such as a motor and its gear drive, by virtue of

[1] Other researchers have used the same term to refer to sets of differential and reflectance properties associated with local scene elements (Breton et al., 1992).

their function, such as a computer terminal and its detached keyboard, or by virtue of their visual appearance, such as an object occluding another one.

As shown in Figure 1.2, the ASDT itself is split into three components: *acquisition*, *integration* and *representation*. All three are interdependent when designing an ASDT (Hayward and Aubry, 1987), but we decouple them hereunder to expose their salient features.

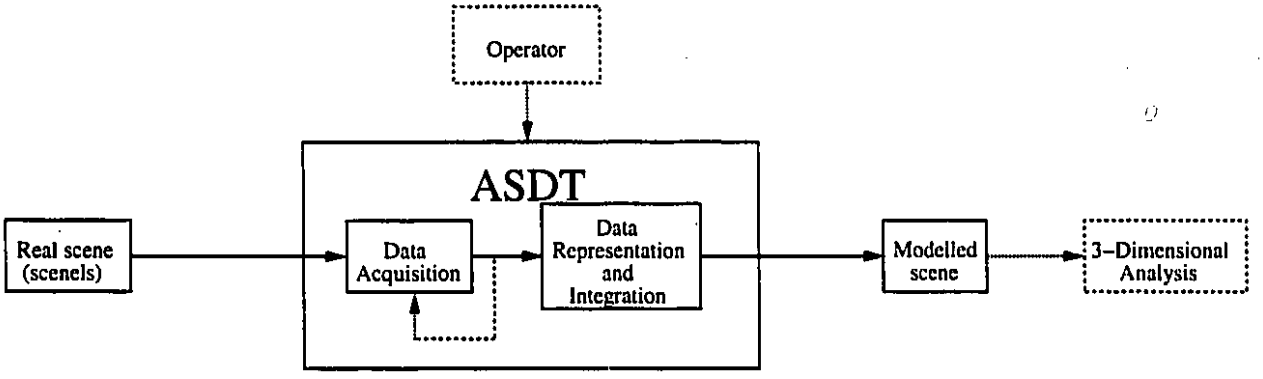

Figure 1.2 The ASDT paradigm. Hashed lines indicate optional paths or blocks.

1.2.1 Data Acquisition

The data acquisition phase senses the scenels' *surfaces* and extracts the three-dimensional coordinates of these surfaces. Methods used to gather surface three-dimensional data can be classified under two main labels (Jarvis, 1983). *Passive methods* use the techniques of computer vision (Ballard and Brown, 1982; Levine, 1985; Nevatia, 1982) and recover the desired three-dimensional information by applying any of the so-called *shape-from-X* techniques. *Active methods* acquire *telemetric* data; their output is trivially mappable to spatial three-dimensional information. In Section 2.1, we give a review of the different techniques, and show how the method of data acquisition

influences the scene description paradigm. In the system we built as a result of this work, real data is acquired with an active two-axis synchronized triangulation-based laser range scanner (Rioux, 1984).

1.2.2 Data Integration and Representation

Because sensor data are always noisy, the first processing task is to filter them, generally at the cost of resolution. We consider filtering to be part of data acquisition because it encompasses many mature techniques (Pratt, 1978; Gonzalez and Wintz, 1977), and because it is best tackled when knowledge about the sensing mechanism is taken into account.

Instead, the data processing task we concern ourselves with is that of *data integration.* For a given scene, a data acquisition sequence generally yields several *data sets*, with each set coming from a different *data source.* Each raw data set must then be correlated to the others in order to arrive at a global description of the scene. We distinguish two types of data integration:

Sensor fusion This term usually refers to the case where multiple sensor readings reduce the uncertainty associated with each. It is common to treat each reading as the value of a random variable and to deduce the most likely estimate for the underlying physical parameter using Kalmann filtering techniques (Durrant-Whyte, 1987; Moutarlier and Chatila, 1989). When different sensing modalities are used, it is common to use prediction-verification paradigms (Aggarwal and Magee, 1986; Krotkov and Kories, 1988).

Multi-View Integration In general, it is necessary to position the sensing apparatus at several locations in order to completely acquire the desired scene. For example, opacity prevents optical sensors from "seeing" beyond occlusions. Hence, several optical sensors "looking" at a given scene but from different locations acquire different but complementary information. In this case, the data sets must be put in mutual relation, or *integrated.* In particular, we must establish data *connectivity.* In Section 3.1, we give a full review of existing techniques for doing so.

A major contribution of this thesis is how to accomplish the integration task by incorporating information about the data acquisition procedure.

Finally, data representation, or data storage, refers to the maintenance of the filtered and integrated data into a digital computer. The main (and often conflicting) objectives of the many forms of data representation are ease of input, compactness and ease of use. *Solid Modeling* is the field that studies these issues and we review it in Section 2.2. Because the data integration algorithm depends in a large part on which representation one chooses, data integration and data representation are displayed together in Figure 1.2.

1.2.3 Global vs. Incremental Processing

Figure 1.2 features an optional feedback loop around the ASDT. When the loop is not active, we say that the scene description process is *global*: All data sets are acquired before the ASDT builds the model. When the loop is active, the scene description process is *incremental*: intermediate models are built after each data set acquisition.

The incremental property [2] is in general desirable because the evolution of the model

[2] Also called *on-line property* in the Computational Geometry literature.

trough successive sets of data is better controlled. First, it allows the use of a stopping criterion for the number of data sets to be acquired and integrated, based on the quality of the current model. The fit can be purely empirical and based on the judgment of the operator, or an analytic test for model convergence can be established. Another advantage of the incremental property is that the data may not all be available at once. In this case, it is preferable to build a model with only partial information, and to complement the model whenever new information becomes available, without having to rebuild it from the ground up. As we see in this work, not all integration algorithms possess the incremental property.

1.2.4 Three-Dimensional Analysis

Figure 1.2 features an optional post-processing block labeled *three-dimensional analysis*. It refers to the process of isolating and describing the subcomponent shapes, notably by isolating the discontinuities in the surface data. Three-dimensional analysis is a large field (Ferrie, 1986; Besl and Jain, 1988; Leclerc, 1989; Ferrie et al., 1990). It leads to a much more concise representation for the scenels, while still capturing their major features. Object recognition (Besl and Jain, 1985; Lowe, 1987; Fan et al., 1989) is a typical example of an application requiring compaction into such a terse model. For reasons of tractability, three-dimensional analysis is also generally performed for most of the applications mentioned in Section 1.1.1.

We consider three-dimensional analysis to be outside the scope of the ASDT. The ASDT's function is to output a faithful, rather than compact, model of the scenels. In this sense, the ASDT paradigm embeds a *least-commitment* principle, whereas three-dimensional analysis commits to choosing significant features and shape parameters among a large set of possible ones. Obviously, the model the ASDT outputs must be

amenable to further three-dimensional analysis. Depending on the model's ultimate use, three-dimensional analysis may then be performed, as indicated by the post-processing block.

While Figure 1.2 illustrated the ASDT's paradigm, Figure 1.3 illustrates the paradigm for which the result of three-dimensional analysis is the goal. In the latter figure, three-dimensional analysis is performed *before* the resulting model is output. Data integration must still be performed if data is acquired from separate sources. In Chapter 3, we further expose why the integration procedure differs in both paradigms, as a result of their different objectives. Finally, note that the optional incremental loop is also featured in this paradigm.

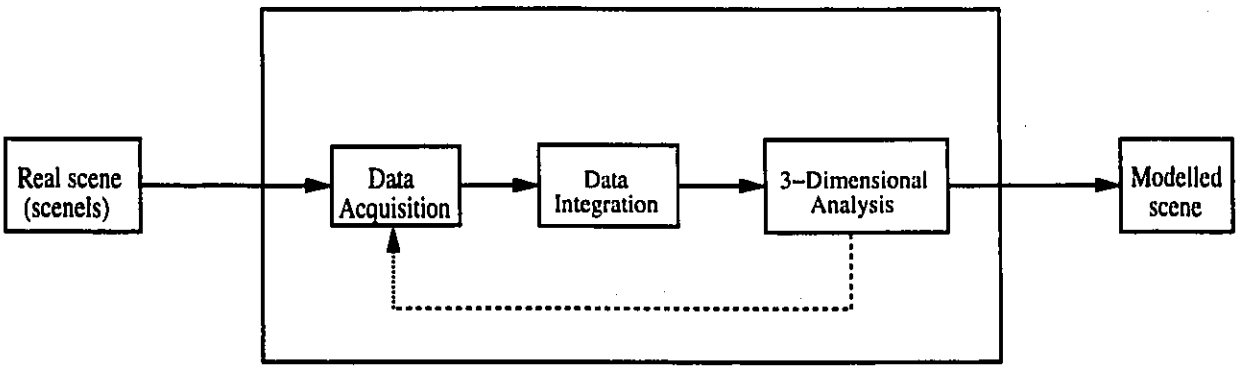

Figure 1.3 The Three-dimensional Analysis Paradigm. Hashed lines indicate optional paths or blocks.

This concludes the description of the ASDT's components. The next section addresses the specifications of these components.

1.3 The Specifications of an ASDT

1.3.1 Sources of Variation for Data Acquisition

We first consider data acquisition. For each set of scenels that is to have its surface scanned by one or more sensors, an infinity of different data sets can result. One source of variation is sensor noise. Clearly, noise is undesirable and is directly related to the type of sensor in use. Yet noise is always present, as no sensor reading is perfectly repeatable, but rather is a random variable. In the following discussion, we disregard noise to concentrate on the other sources of input data variation one has control over, given a particular sensing device.

We assume that the data is in the form of discrete three-dimensional *points* that lie on the surfaces of the scenels. In Chapter 2, we show that this assumption is consistent with both direct and indirect methods of three-dimensional data acquisition. The points are a spatial sampling of the surface under consideration. The finer the sampling, the more faithful the output model. Sampling size is fully under control of the operator, up to the resolution of the sensing device. Hence sampling size is a controllable source of variation. Another controllable source of variation is the extent of the scene acquisition. Namely, a choice exists as to which scenels and which scenel parts are to be acquired. Other controllable sources of variations for sensing are where to position the sensors, how many independent readings to take, etc...

1.3.2 Consistency of Data Integration

A given scene may give rise to many different data sets, because of the above-mentioned sources of variation. Figure 1.4 illustrates the ASDT processes more formally. S is the set of possible world scenes, and D is the set of $< v, w >$ pairs where:

- v is a three-dimensional vector expressing the position of a scenel surface point in a world reference frame $\mathcal{W}$,
- w is a composite structure containing information about the type of sensor used to acquire v, and the sensor's position and attitude with respect to $\mathcal{W}$.

Each sensor-acquired surface point yields a $< v, w >$ pair and constitutes a datum for the purposes of this presentation. Thus, datum $< v, w >$ is distinct from datum $< v, w' >$ if $w \neq w'$. As this work illustrates, the sensor-related details w of how the value v is derived contains invaluable information that can be used during the later data integration stage. From the above discussion on the sources of variation of the data acquisition procedure, we model data acquisition as a one-to-many relation $\mathcal{F}$ from S to $P(D)$, where S and D are defined as above, $P(D)$ is the power set of D, and $(s\ \mathcal{F}\ d)$ if and only if it is physically possible to obtain the set $d \in P(D)$ by sensing the surfaces of the scene $s \in S$. We also model the data integration and representation procedure as a mapping $\mathcal{G}$ from $P(D)$ to M, where M is the sets of models available with the chosen representation. $d\ \mathcal{G}\ m$ if and only if the integration algorithm outputs the model $m \in M$ given the data set $d \in P(D)$.

We also define the subset $S' \subset S$ of *ideal* scenes. An ideal scene is a scene that can be modeled *exactly* by a model of M. Since S is the set of real scenes, the probability that a given scene belongs to S' is zero. Note that the larger the expressive power

of the chosen representation is, the larger S' also is. Then we define the "natural" bijection $\mathcal{B}$ from M to S', which to each model $m \in M$ associates the ideal scene $s' \in S'$ which m models exactly.

We are now ready to define representation consistency. We say that the model m is a *consistent* representation of the scene s given a particular data set d acquired on s (hence $s \mathcal{F} d$) if and only if

$$\Big(\mathcal{G}(d) = m\Big) \quad \Longrightarrow \quad \Big(\mathcal{B}(m)\, \mathcal{F}\, d\Big).$$

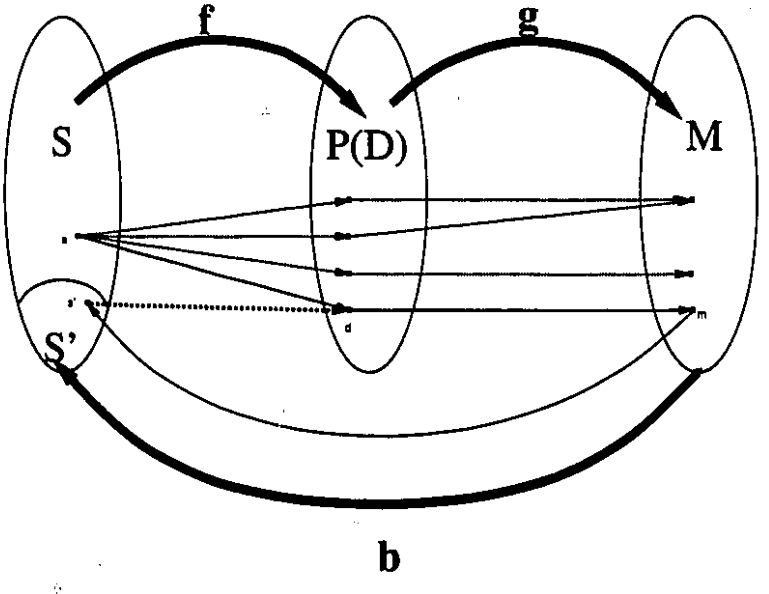

Figure 1.4 Illustration of:
- The set-theoretic relations between the ASDT's components.
- The ASDT's consistency condition. If $s' \not\mathcal{F} d$ (shown as a thick hashed arrow in the figure) then m is not a consistent representation of s.

In other words, the consistency condition guarantees that if the real scene s was replaced by an ideal instance $\mathcal{B}(m)$ of the model m output by the ASDT, then the same data acquisition procedure could still be performed on that instance *and* yield

exactly the same results d, up to noise. Although consistency does not guarantee any degree of "goodness" for the resulting model m, it is a predicate that partitions M into acceptable and unacceptable models for s, based on physical characteristics of the scene and of the acquisition procedure. An inconsistent model $\hat{m}$ is so because the physical characteristics of data acquisition preclude m from being an exact model for the scene at hand. We also say that the data integration and representation step, and by extension the ASDT itself, is consistent if and only if

$$\forall s \in S,\ \forall d \in P(D)\ ,\quad \left(\begin{array}{c} s\ \mathcal{F}\ d \\ \wedge \\ \mathcal{G}(d) = m \end{array}\right) \Longrightarrow \mathcal{B}(m)\ \mathcal{F}\ d,$$

meaning that a consistent integration and representation stage outputs consistent models for all scenes and for all data acquisition procedures. ASDT consistency places a limit on the amount of distortion the model can ever introduce. The limit is that if the real scene was to be replaced by its model, and measurements were to be made on the model, while following the same acquisition procedures as for the original data, then the same numerical data should be obtained, up to measurement errors.

This formal definition of consistency is the basis for the data integration methods we develop in this work for the case of telemetric range finders. These set-theoretic concepts are illustrated in Figure 1.4.

1.3.3 ASDT Efficiency

The ASDT must acquire and model scenes in a "reasonable" amount of clock time, as hours or even days taken on building a single scene seriously restricts its use.

Yet, unless one is modelling time-varying objects, such as people milling about in a room, "real-time" is a clearly desirable but not generally essential specification for an ASDT. Hence a good compromise is that the ASDT be able to model a scene in minutes, or tens of minutes at most. This is the assumption we base our discussion on in Chapter 2, when debating the practicability of different techniques. The speed of the first step, data acquisition, depends almost entirely on the sensing device and will not be addressed beyond the review given in Chapter 2.

The speed of the data integration step, in contrast, depends on the integration algorithm used. We measure it both in terms of actual clock time used on a particular platform and in terms of the number of necessary symbolic operations with respect to the size of the input data. In this thesis, we develop worst case asymptotic complexity relationships (the "big O" notation), as well as empirical expected performance for the algorithms we describe.

1.4 Methodology

In the previous sections, we exposed the need for an ASDT, and we described its components. Then we argued that the ASDT must:

1. **Output a least-commitment representation.** Such a representation retains all available information and delays choosing between competing compacted models until a later three-dimensional processing phase. This is in part because the choice of optimal compacted representation depends on the model's ultimate use, which in turn depends on the particular application (Requicha, 1983). The least-commitment representation provides an *intermediate* representation, amenable to further compaction into a more structured one. In Section 2.2, we

introduce representational issues, and further discuss this point.

Another aspect of the least-commitment principle concerns the representation of uncertainty. Unless the representation's language allows for the explicit encoding of uncertainty, arbitrary choices must sometimes be made between competing equally-plausible models. In this case, we favor *maximal-volume* models. The reason stems from the model's likely ultimate use. Most of the applications we detailed in Section 1.1 require some degree of collision-avoidance between objects. Models which *contain* the objects they represent are said to be *conservative* and are therefore preferred for obvious safety reasons.

2. **Perform multi-view integration.** Because acquiring the three-dimensional surface data of a given scene requires several views, an essential component of the ASDT is the integration of these views into a view-independent model. The integrated model is expressed in a world-reference frame, and includes the data from all views.

3. **Incorporate knowledge about the sensing modality at the data integration stage.** Three-dimensional data acquisition is still an emerging technology (See Section 2.1). Many techniques exist, and use different working principles. Although these technologies are constantly improving, three-dimensional data acquisition is overall a rather slow and imprecise process. It follows that in an unstructured environment where little *a priori* information is available, one is often limited to acquiring *sparse* surface information.

   With such data, surface connectivity is difficult to establish: this is known as the "connect-the-dots" problem and is illustrated in Figure 1.5 for a two-dimensional contour. In the figure, the contour forms the boundary between the object and the free space, and a line-of-sight sensor samples the contour with discrete points. The underlying contour connectivity between the sampled

points cannot be established on the basis of their coordinates alone. However, it *can* be approximated if the geometry of the line-of-sight segments is taken into account, as we show in this work.

In Section 3.1.2.3, we review existing methods for determining connectivity between surface points, in both the two- and three-dimensional cases. In later chapters, we propose a general paradigm to determine point connectivity, which considers not only the raw data, but also the way in which that data was acquired. Namely, we use the additional information offered by the knowledge about the data acquisition procedure, in order to infer point connectivity. For this reason, we say that we use the knowledge about the sensing modality at the integration stage. The next paragraphs expound on the point.

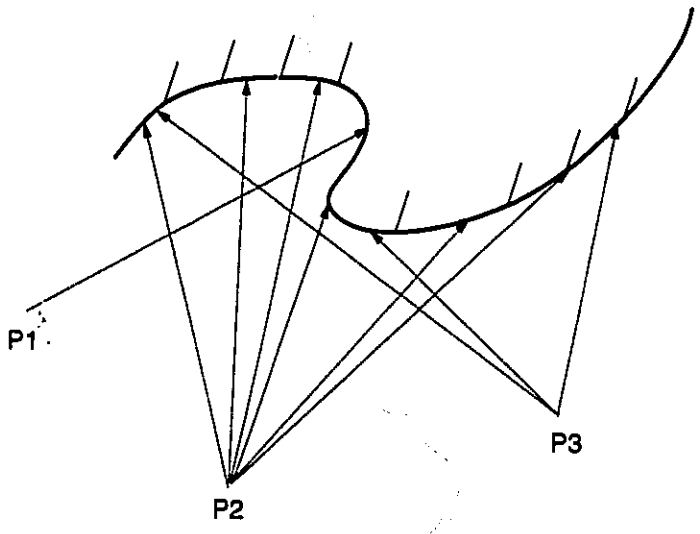

Figure 1.5 A two-dimensional connect-the-dots problem. P_1, P_2 and P_3 are three camera positions. The connectivity between the sampled contour points cannot be determined from the points' coordinates alone.

As explained in Section 1.3.2, we assume each datum is a $< v, w >$ pair. The crucial information contained in the structure w is a (possibly implicitly encoded) region of

the three-dimensional space that is known to be free of any obstacle or scenel by virtue of the physical data acquisition procedure. This region naturally includes the physical location of the sensor itself, as well as a *visibility region* [3] E, where E lies "between" the sensor and the scenel being sensed.

In the case of contact sensing, for example, E is empty, so the free-space region encoded in w is simply the physical space taken by the sensor. In the case of optical sensing, which is of particular interest for this work, E includes one or more directed line-of-sight *segments* joining the imaging system's optical center(s) to the surface point v. Obviously, these segments cannot cross the scenels' surfaces, for if they did, the scenel would not be opaque. But the acquired surfaces *must* be opaque in order to reflect light and to allow acquisition with such a system. In other words, either the surfaces are opaque, vindicating the assumption that they cannot be intersected by the segments, or they are not, and the acquisition process itself yields spurious information. Therefore, we assume that all scenels of interest are opaque.

The integration procedure is then as follows: the information provided by the sensor isolates areas that are known to lie entirely in free space. Then the loci of these areas are refined by performing sets of *geometrical* tests on the data. Additionally, the refinement process determines data connectivity, from which a representation for the scenels is inferred in turn.

The geometrical tests use both the surface coordinates (v) and the acquisition details (w). They are designed to guarantee that no part of the resulting model intersect any of the E regions.

The output of the integration procedure is a graph whose vertex set is the the set of surface points v, and the edge set represents the connectivity between the points.

[3] We employ the term visibility very loosely in this general introduction.

Every edge of the graph must belong to at least two cycles, or *faces*, so that the resulting graph is a *tesselation* of a surface.

The faces of the tesselation can be modeled as patches. When the tesselation is a triangulation, the simplest such patch is the first order approximation, namely the planar patch. For lack of better knowledge of the differential-geometric properties of the underlying surface, this is the representation we choose by default.

Figure 1.6 illustrates an example on which the algorithm was run. The resulting model is shown in Figure 1.7.

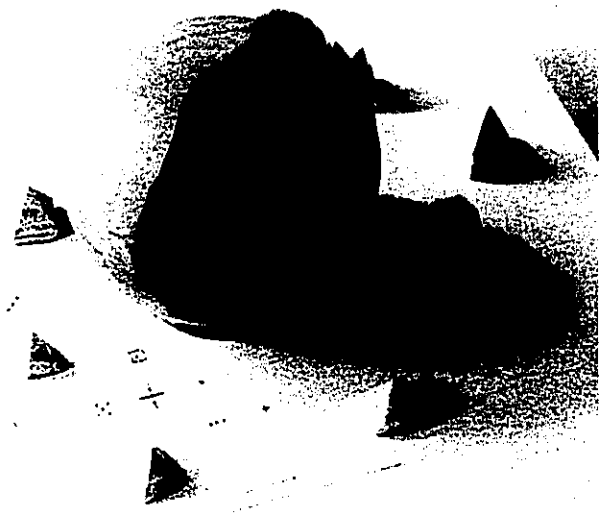

Figure 1.6 Photograph of a pencil holder. The small pyramids are not modeled, but are used as fiducial marks, as explained on Page 123.

1.5 How to Read this Thesis

This chapter introduced the ASDT, and described some of its uses. In particular, we argued that off-line programming requires being able to plan the robot motions,

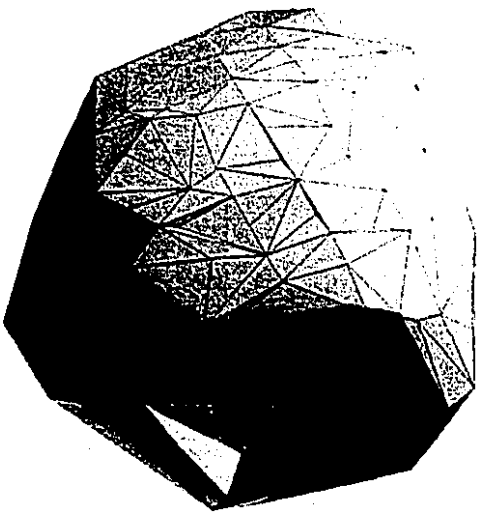

Figure 1.7 The resulting connectivity graph on a set of sparse three-dimensional surface points, as computed by the algorithm described in Chapter 5.

that planning required the existence of a model of the environment, and that such a model is very difficult to obtain without an ASDT. Then we gave requirements for the ASDT, and we used them to claim that the ASDT must perform multi-view integration, must use a least-commitment representation, and must make full use of the available knowledge about the data acquisition procedure.

In the next two chapters, we do a literature review of the state-of-the-art for these issues, namely data acquisition, data integration, and data storage. Most readers can skip the surveys of range data acquisition and solid modelling given in Chapter 2. Chapter 3 reviews the state-of-the-art for the multi-view integration problem and introduces the problem this work addresses. It should be read to put the later chapters in perspective with previous work.

Chapter 4 describes a data integration algorithm that uses an incremental paradigm. We did not implement that algorithm because we believed it did not represent a good

balance between efficiency, simplicity, and the generality of the problem it can successfully solve. For this reason, the reader may skip Chapter 4 without compromising the understanding of the later chapters.

In Chapter 5, we describe an alternate more general algorithm, which does not however possess the incremental property. We implemented this algorithm and tested it on real data. Finally, Chapter 6 discusses the results, and describes further work.

Chapter 2

Literature Review

In Chapter 1, we described the ASDT's three main issues: data acquisition, data representation and data integration. We also argued that the relationship between these components must be kept in mind while designing an ASDT. In this chapter and the next, we review existing techniques to accomplish that goal.

2.1 Data Acquisition

Sensing the scene with an apparatus capable of effecting dimensional measurements is obviously the first step of the description process, and a host of methods exists for performing such measurements (Beckwith et al., 1981, Chapter 11). This section investigates and identifies candidates for use by an ASDT.

2.1.1 Data Acquisition Specifications

A data acquisition technique must combine speed and precision, while being practical and providing a fair level of robustness. Below, we give rule-of-thumb numerical values for these specifications.

In Chapter 1, we mentioned that the ASDT must complete its task in no more than tens of minutes of clock time. This upper bound on total scene building time implies a time bound at least as stringent on the data acquisition phase. The requirement can easily be met with current technology.

We also saw in Chapter 1 that a consistent ASDT builds models which are as accurate as its data acquisition system allows. Hence, the latter determines the accuracy of the final model. Since robotics is one of the major motivations for this work, the accuracy of the model, and by extension of the data acquisition system, should be at least as good as that of the robotic workstation. Most industrial robots achieve a resolution accuracy in the order of the millimeter. This is the yardstick we use to select viable data acquisition systems. These requirements (acquisition time and accuracy) are roughly in line with those given in (Jain and Jain, 1989, Table 1).

Finally, an ASDT must achieve the above requirements with a range of depth and a field of view in the order of the meter. This measure grossly corresponds to the size of industrial robots and of their work envelope.

From these specifications, we can eliminate near-contact or contact methods.

Near-contact methods use proximity sensors, whose output depends on the sensed material's electrical or light-reflective properties. For example, pneumatic comparators and optical flats measure dimensions with great precision and without assuming particular properties about the objects.

Contact methods record the displacements induced by the object' surface on a measuring device. Such a device is the CMM (Menq et al., 1992; Choi et al., 1990) which is mainly used as a manufacturing inspection tool. Recently developed miniature touch sensors can also be mounted on robot manipulators and used to model the environment by way of groping (Allen and Michelman, 1990).

Despite their very high resolution capabilities, both contact and near-contact devices are too unwieldy and impractical for the purposes of the ASDT, because of the large number of measurements (usually several thousand data) needed to model an entire scene. As a result, the time constraints given above appear difficult to meet, because of the considerable set-up time involved.

Long-range optical methods, such as *photometry* and *optical telemetry*, do not suffer from these restrictions, as they permit the acquisition of a large number of data in a short amount of time. Namely, each datum can be acquired in the order of microseconds to milliseconds, while overall setup time is in the order of minutes. This is a satisfactory speed for our application. These techniques are often called *range finding* techniques, since they determine the distance between the sensing element and the points on the scenel surfaces. We review them using the classification given in Chapter 1, separating active from passive methods. We will see that active methods are more suited to the purposes of the ASDT, but for completeness and because of the close relation between passive and active methods, we first give a detailed introduction to the latter.

2.1.2 Passive Methods

Passive methods are characterized by the prior acquisition of photometric or intensity *images*. An image is simply a two-dimensional array of light intensity values. The

passive methods are so called because no light is shone onto the scene. The light that reaches the array receptors is therefore due to ambient lighting only.

Such intensity images are difficult to analyze in general because they lead to ambiguous interpretations. They map intensity, an extrinsic characteristic of the three-dimensional world, onto the two-dimensional image plane along the lines of a perspective projection. The task of recovering the correct interpretation for a given image is then a formidable one since it requires that the perspective ambiguity be resolved from the intensity cue alone.

A simplified image formation model can be written

$$\mathcal{IM}(a,b) = I(x,y,z),$$

where I is the illumination-reflectance operator whose range is the set of scene surface points visible to the sensing elements, and $\mathcal{IM}(a,b)$ is the intensity value associated with the (a,b)-element of the image.

Inverting the operator I then recovers the three-dimensional coordinates of the surface points of the scene. It can be proven that this process is an ill-posed problem (Terzopoulos, 1988). Nevertheless, the various *shape-from-X* techniques that we quickly describe below are attempts to perform that difficult inversion.

2.1.2.1 Shape from Shading

These approaches compute the normal vector for the surface points that I map into the image. Knowledge of the depth of just one point then theoretically suffices to determine a depth map for all connected points, by successive integration.

Horn (Horn, 1975) proved that the perceived intensity is a function of the gradient

of the surface orientation, except at what he calls singular points, where the surface orientation can be locally determined from the intensity. Knowledge of the reflectance map and of the illumination model is necessary to reconstruct an approximation to the surface map. The solution involves solving a first-order non-linear partial differential equation and is essentially imprecise. It also requires establishing certain constraints, such as the sign of the second derivative at singular points. However it breaks down for some classes of discontinuities, such as those created by occlusions. Pentland (Pentland, 1986) arrived at a more robust solution by constraining surfaces to be locally spherical and by assuming Lambertian reflectance only, whereas Ikeuchi and Horn (Ikeuchi and Horn, 1981) dealt with surfaces with a high specular component. Smith (Smith, 1983) proved that surfaces cannot in general be *exactly* recovered from shading alone, while Ferrie (Ferrie, 1986) placed quantitative bounds on the limitations of shading analysis.

2.1.2.2 Shape from Texture

Here, surface texture is used as a clue to derive depth. For example, so-called gradient methods derive surface orientation from the tilt and slant parameters of the underlying surface. The parameters are estimated from the direction of maximum rate of change of the projection of a set of surface *primitives*. These methods assume that primitives of a known shape dot the surface, and that the surface is made up of planar patches of a sufficient size (Stevens, 1979; Kender, 1978). We note that the first condition can be reliably enforced by projecting the desired patterns of light onto the surface, as is done with structured lighting techniques (Will and Pennington, 1972; Potmesil, 1979; Wang and Aggarwal, 1989).

2.1.2.3 Shape from Focus (Horn, 1968; Jarvis, 1976; Krotkov and Martin, 1986)

This method is based on a very basic optical principle: any point of the scene is in focus for only one lens-to-image distance, and this distance is a unary function of the distance between the point and the lens (i.e. the range.) Therefore the absolute distance can be found by determining at which lens aperture a computed measure of focus quality is maximized. Since a well-focused image is synonymous to a sharp one, the existence of high spatial frequencies in the window of interest indicates a high quality focus and hence yields the range. The range at which the focus is optimal is found by dichotomic trial and error. Despite its simplicity, this method is slow since the image plane must be sequentially physically displaced and the focus quality computed for many possible range values in order to determine the sets of points in focus at each position (luckily focus quality is a unimodal concave function of aperture). Moreover, the method breaks down in the face of homogeneous regions since they are devoid of high frequency features. This technique was found useful for special-purpose applications, such as automated focusing for commercial cameras (Goldberg, 1982).

2.1.2.4 Shape from Occlusion Cues (Rosenberg et al., 1978)

This method infers depth relationships by building a depth graph based on the occlusion evidences appearing in a segmented image. Probabilistic relaxation labeling (Hummel and Zucker, 1983) is used to resolve contradictions in the partially constructed graph. Despite its elegance, this method suffers several drawbacks: it is fairly complex, it only yields relative distances and it depends on the existence of reliable occlusion clues as well as of prior reliable segmentation of the scene.

2.1.2.5 Shape from Stereo Disparity (Yakimovski and Cunningham, 1978; Levine et al., 1973)

Suppose a scene surface point projects on two different image planes. The *stereo disparity*, or distance between the two projections in image coordinates, is inversely proportional to the point's distance to the image planes, or range.

This simple principle forms the basis for stereo methods. All such methods rely on finding sets of feature points in the scene and matching them in the two (or more) images. This difficult process solves the *correspondence problem.* It is usually carried out by computing a correlation function over window pairs, with one window coming from each of the images. Once the correspondence is established, it is a relatively easy task to compute the depth of various points of the scene from simple geometric relations, since we assume that inter-camera distances are known.

As with the focusing approaches, the windows being analyzed must contain enough high frequency components for the correlation measures to be meaningful. If large areas of the scene are featureless, or if the scene's features are repeated (imagine the case of a macro-texture), window matching may turn out to be ambiguous and unreliable. This difficult problem is sometimes tackled by exploiting one or more of the following ideas:

- Not attempting to establish correspondence over windows but over chosen image features only, such as oriented edges (Baker and Binford, 1981), zero-crossing of the second derivative (Marr and Poggio, 1979), or surface differential-geometric properties (Ferrie, 1986). This allows us to compute the depth of all "interesting" points, whereas that of the other areas of the scene can be found by interpolation.

- Improving the reliability of the correlation match by using a hierarchical search strategy similar to that of Tanimoto's pyramid (Tanimoto and Pavlidis, 1975). The idea is to perform the matching starting at a low resolution, to isolate the matched image areas, and to repeat the process on those very areas after expanding them to the next higher resolution (Moravec, 1979).

- Making use of cooperative algorithms which first perform matching at a local level and then "correct" the correspondence results on the basis of global continuity constraints (Baker and Binford, 1981; Marr and Poggio, 1979).

- Bypassing the correspondence problem by the use of structured lightning techniques (See Section 2.1.3.1).

2.1.2.6 Passive Methods: Conclusions

Many other methods were left out from the above survey, including for example shape from contour (Stevens, 1979; Witkin, 1980), shape from motion (Ullman, 1979; Skifstad and Jain, 1989), shape from parallax (Soneira, 1988), or shape from shadows (Raviv et al., 1989). The reader interested in a more complete presentation of passive ranging methods can refer to the authoritative survey of Jarvis (Jarvis, 1976).

The problem with passive methods is that non-geometric clues (occlusions, texture, blurring, etc...), or underdetermined geometric clues (epipolar disparities, apparent motion), are used to infer geometrical reality. As a result, all such methods require extensive and complex processing. In practice, computing shape descriptions on a few hundred or a few thousand points (pixels) on modern computers fail to meet the speed requirements laid out above by several orders of magnitude.

More importantly, they make serious restrictive assumptions about the observed

world. For this reason, no such method taken in isolation can achieve a high degree of robustness.

Biological vision systems provide a proof by existence that passive depth reconstruction is possible, and this has greatly motivated computer vision researchers to pursue such methods. However, they have long conjectured that deriving depth maps from intensity-based images necessitates the introduction of high-level cognitive processes, if robustness is to be achieved. Not surprisingly therefore, the above-mentioned *computational* techniques fail to adequately reconstruct the three-dimensional world. Interestingly, the numerous visual illusions we humans experience are a clue that we use imperfect, though powerful, methods to treat visual data, and the study of neurophysiology and psychophysics has greatly contributed to computer vision (Levine, 1985; Marr, 1982). For example, it has been shown that humans make errors in orientation estimates based *solely* on shading cues (Horn, 1975).

2.1.3 Active Methods

In opposition to photometric images, range or telemetric images embody an explicit representation of the geometry of the scene and therefore allow us to bypass the computationally difficult inversion of the illumination-reflectance operator I. The accuracy of the range information is then solely limited by the that of the sensor since telemetric images are an intrinsic representation of the scene's geometry. In what follows we will use the terms telemetric images and range views interchangeably.

Range views are obtained through active vision. Instead of relying on ambient illumination, an active vision system shines light onto the scene at regular spatial intervals, thus artificially creating unambiguous features that greatly simplify the processing: the monitoring of the directed light yields two-dimensional geometrical information

whose processing is straightforward. By placing a receiver at a known position with respect to the emitter, the third dimension (the range) can be resolved from purely geometric methods.

In other words, the interpretation of the returned visual data is made considerably easier by the disambiguation the added degree of control provides. The price to be paid is the additional hardware requirement.

Methods of obtaining range data vary, both in their principle and in their implementation. Range imaging is still relatively new and for this reason it may well be that future technological evolutions improve the feasibility of some of the less promising methods described hereunder. A good survey of existing active range imaging systems can be found in (Besl, 1988). We give a brief review of these techniques below, starting with the most-widely used of them, triangulation.

2.1.3.1 Range from Triangulation

Figure 2.1 illustrates the triangulation principle. A light source O, usually a laser, beams a ray towards a scene point P, which then reflects it back. If the surface is Lambertian or nearly so, all directions will pick up a significant return signal. By placing a receiver, usually a CCD camera, at Q such that the emitted and reflected rays form a non-null angle, one can determine the distance, or range, of P from simple trigonometry arguments: Since both the origin and the angle of the emitted ray are known, the equation of the emitted ray is fully determined. Similarly, the reflected ray is constrained to pass through both the center of the lens Q and the lit camera element IM_x, thus determining its equation. Both rays intersect at P, thus determining its coordinates.

By choosing a coordinate system positioned at Q and whose z-axis is orthogonal to

the image plane, we can write:

$$d_1 = z \cot \alpha \tag{2.1}$$

$$d_2 = \frac{z I M_x}{f} \tag{2.2}$$

$$D = d_1 + d_2, \tag{2.3}$$

where D is the epipolar distance, f is the focal distance, IM_x is the offset of the lit element in the camera image plane. The "range", or z-coordinate of P is thus

$$z = \frac{D}{\frac{IM_x}{f} + \cot \alpha}. \tag{2.4}$$

Triangulation range finding offers several advantages. First, it it based on a robust principle, and avoids many of the restrictive assumptions of the shape-from-X methods. Second, it offers a fairly good accuracy, usually considerably better than 1% of measured range. This means that accuracies in the order of the millimeter can be obtained by limiting the range to under 1 meter. Finally, because it only requires geometrical computations rather than knowledge-based processing, three-dimensional data acquisition is greatly sped up, as hardware becomes the factor limiting acquisition speed. Although not as fast as intensity imaging, acquisition of several thousand three-dimensional data can usually be done in seconds or less. This point is further discussed below.

For these reasons, we propose triangulation as the method of choice for automating the process of scene description.

The recurrent problem encountered when using triangulation for range determination is the *missing part problem.* Since the emitter and the receiver are not coaxial, the

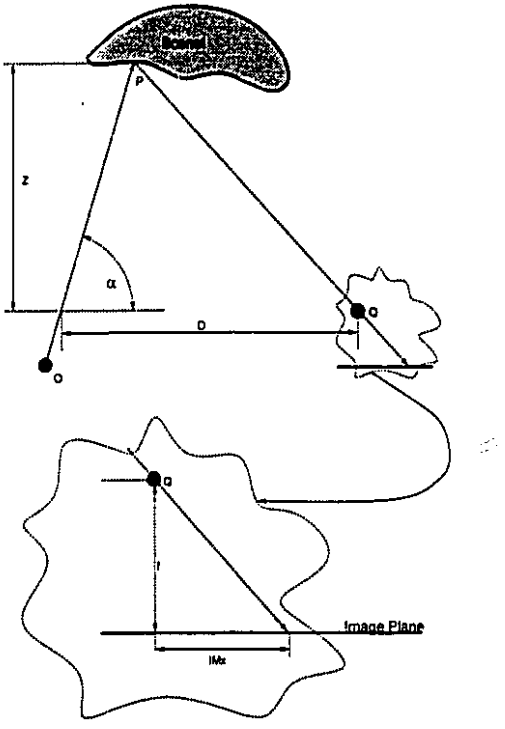

Figure 2.1 The triangulation principle.

image perceived at the receiver does not register with the set of points illuminated by the active lighting system. Consequently, missing parts, or *shadows* appear at locations of the range image where no z component can be computed.

Shadows are illustrated in Figure 2.2 and are of two types. Type I shadows arise when the illuminating ray does not reflect back to the receiver (because of object self-occlusions.) Type II shadows arise when the illuminating ray cannot reach a point of the object which is visible from the receiver. The smaller the angle between the emitter and the receiver, the smaller the shadows, but also the less accurate the measurements.

Type I shadows can be reduced by introducing an additional receiver (Bhanu, 1984)

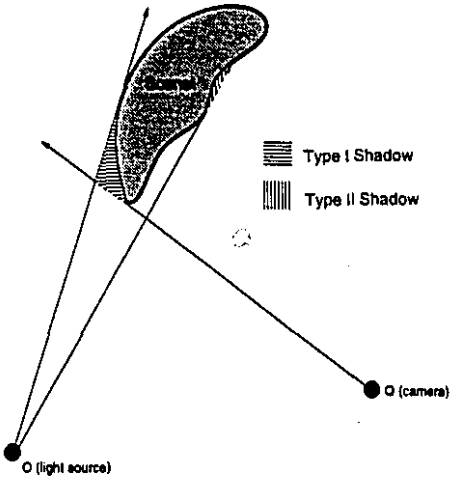

Figure 2.2 The missing data problem.

whereas type II shadows can be reduced by the introduction of an additional ray emitter (Sato et al., 1982). In principle, there is no limitation to the number of extra emitters or receivers that can be introduced in order to reduce the shadows to a minimum, although total shadow elimination can never be guaranteed. Because of the additional hardware involved, the preferred setup remains the single emitter-receiver combination. Further, missing data are just as easily recovered by displacing the range finder to other locations and acquiring new views.

We now turn our attention to the scene illumination method. The illuminating ray is deflected onto selected areas of the scene with a combination of opto-electronic shutters or of rotating mirrors. Some popular configurations include:

Point Triangulation (Ishii and Nagata, 1976) The original and simplest method. A light spot is sequentially directed towards all sampled points. The mechanical movements involved in directing the spot slow down the acquisition process. In their original paper, Ishii and Nagata reported a time of almost one minute for

acquiring a 128 × 128 image. Fortunately, recent advances in hardware have considerably sped up the acquisition process. For example, Rioux (Rioux, 1984) built a sensor that can acquire a similar image in less than a second.

Slit triangulation (Agin and Binford, 1976; Oshima and Shirai, 1983) This method reduces the number of necessary mechanical movements by scanning light slits, instead of spots, across the scene. A peak detector attached to the camera allows to rapidly detect the set of pixels illuminated by the projected slit. In (Oshima and Shirai, 1983), the total time for obtaining the range image was reduced to a few seconds for a 128 × 128 image. A more recent paper by Ozeki *et al.* (Ozeki et al., 1986) in which range computations are done in hardware claims a processing time of half a second for a 50 × 50 image, or 3 seconds for a 128 × 128 image.

Space Encoding Whereas point triangulation and slit triangulation require n^2 and n distinct samples respectively for an $n * n$ range image, structured lightning methods cut the number of necessary samples further. Judicious encoding of the projected patterns must be used to ensure that the correspondence between the projected illuminant and its position on the receiver's image plane is maintained. Altschuler *et al.* (Altschuler et al., 1981) describe a system that projects a set of binary coded masks on the scene with the help of an electro-optic encoder. $\log_2 n$ masks are sufficient to entirely recover an $n * n$ image. Sato and Inokuchi (Sato and Inokuchi, 1985) use a Gray code rather than a plain binary one, thereby making the method less error-sensitive. The bottleneck of this method resides in the large number of digital memory accesses necessary to store and retrieve the patterns.

Space encoding approaches culminate with methods that encode the scene with a *single* lighting pattern. In (Carrihill and Hummel, 1985; Tajima, 1987), the

single pattern contains varying intensity lightning subpatterns. In a general environment, however, the more robust patterns are those with only two intensity levels, "bright" and "dark", such as the color-coded stripe patterns of Boyer and Kak (Boyer and Kak, 1987), or the overlapping binary patterns of Vuylsteke and Oosterlinck (Vuylsteke and Oosterlinck, 1990).

2.1.3.1.1 Other Triangulation Scanners Various techniques have been developed to improve the accuracy (Harding and Goodson, 1986), to expand the range of depth (Bickel et al., 1985) or the field of view (Rioux, 1984), or to realize a compact design (Rioux and Blais, 1986). We describe the last two techniques below.

The Synchronized Scanner increases the field of view with a variant of the point by point triangulation. It comprises three rotating mirrors, two of which rapidly move in unison while the third one rotates at a much slower pace. The faster mirrors sweep the light spot along a line, whereas the slower one increments the sweep line in an orthogonal direction, as in the slit triangulation method described above.

The originality of this scanner resides in the addition of the second fast mirror which moves synchronously with the first one and deflects the ray. A geometrical analysis reveals the advantages of the setup:

- All light rays are returned to a single line segment and a linear sensor is then sufficient to collect them. By comparison, a full two-dimensional camera is needed in most other setups. The bulkiness of the ranging head is reduced.
- The light rays hit the sensor over a shorter range than they would using other geometries. This permits the use of a lens with a longer focal length, and hence a better resolution, without sacrificing on the field of view. Alternatively, the same resolution can be kept but with a smaller emitter-to-receiver angle. As

mentioned above, a smaller angle reduces shadow effects.

- The short range over which the light hits the position sensor permits the use of a small analog sensor, such as a lateral effect photodiode, instead of a slower CCD array coupled to a peak detection circuit. The high dynamic range of the former device allows capture and analysis of a 128×128 image in less than half a second. In addition, such a sensor also outputs the amount of light received, hence "automatically" yields a registered intensity image. However, the advantages of such a sensor have to be weighed against the higher sensitivity, the higher resolution and the absence of electronic drift of CCD's.

In the compact 3-D camera of Rioux and Blais, a small mask pierced with two holes is placed in front of a CCD camera. Planes of light are shone on the scene, hit the object, pass through the mask, and hit exactly two camera elements. The separation of these elements is inversely proportional to the degree of focus of the object and this automatically yields the z-coordinate of the point (the range); the absolute position of the elements yields the x- and y- coordinates. Because of its extreme simplicity, this camera is very light and inexpensive.

This concludes the presentation of triangulation techniques. For completeness, the next subsections review other active methods of obtaining range information.

2.1.3.2 Range from Time-of-Flight (Lewis and Johnston, 1977; Nitzan et al., 1981; Moring et al., 1987; Svetkoff et al., 1984)

The idea of time-of-flight ranging is simple: one directs a ray of light onto the scene, and measures the elapsed time until the signal returns. Since the emitter and the receiver are coaxial, this method is immune to the missing part problem of triangulation-based techniques. For the same reason, a registered intensity image can be obtained

along with the range image (Nitzan et al., 1981). The availability of registered intensity and depth images can prove a very valuable tool if one is to later perform processing on the image, notably as a way to reinforce discontinuity cues (Gil et al., 1983).

Time-of-flight range finders meet the resolution, work envelope and speed specifications laid out in Section 2.1.1. For example, the most recent sensor designed at ERIM offers a resolution of .25 mm over a distance of 1 meter, and can acquire a $512 * 512$ image in one second (Jain and Jain, 1989).

Time-of-flight however is hampered by the specialized hardware it requires. The speed of light demands electronics with picosecond discrimination. Powerful lasers, which are expensive, bulky and unsafe, must be used in order to improve signal-to-noise ratio. This method therefore appears to be difficult to apply in an industrial environment. However, should the required hardware become available at a non-prohibitive cost, this method of obtaining range maps could become very popular, especially if, for obvious safety reasons, the power of the laser can be kept low (below 10 mW).

A variation on the time-of-flight idea consists of using acoustical (ultrasound) rather than light waves (Moravec and Elfes, 1985; Acampora and Winters, 1989; Audenaert et al., 1992). Even though sound waves travel considerably slower than light, thus avoiding the drawbacks of the previous method, they present other problems. First, unlike light which can be made coherent, sound is hard to focus, offering a poor resolution. For the same reason, reflections off surfaces other than the one being gauged can easily confuse the sensor. Finally, frequent recalibration is needed as the speed of sound is very sensitive to changes in both ambient temperature and humidity. Yet ultrasound ranging is fast and inexpensive. Because of its higher reliability at close range (even if still inaccurate), it has been found to be adequate for collision

avoidance in automated navigation applications (Borenstein and Koren, 1988).

2.1.3.3 Range from Moiré Gratings

In this method, light is shone through a grating and the returned ray (after it hits the object) passes through another grating with the same pitch. The laws of interferometry (Idesawa and Yatagai, 1980) tell us that the output yields contour lines of equal range whose spacing depends on the Moiré pitch. The method is simple and generally offers a resolution at least as good as triangulation range finders. Unfortunately, it only yields relative distances and cannot handle occlusions or range discontinuities. Further, it requires a dark room and specialized optical equipment. Moiré methods are reviewed in Reid (Reid, 1986).

Many other specialized active methods have been described in the literature, for example holographic methods (Tozer et al., 1985), which are used to detect very fine surface variations. Okada (Okada, 1982) also describes a short-range active sensor which detects range based on the quantity of reflected light.

2.1.4 Ranging Methods: Conclusions

We have reviewed the main current 3-D imaging techniques, and seen that the so-called passive techniques are still a long way from performing with the kind of assurance that humans effortlessly display at the task. For this reason, we believe that active techniques, and in particular triangulation methods are still the most promising ones for scene acquisition. Their main drawbacks, mainly their relative slowness and incomplete data acquisition (i.e. the missing part problem) are slowly being resolved with the emergence of faster, more compact and more reliable hardware, as well as with numerous refinements of the basic operating principle.

Finally, we note here that laser time-of-flight is still impractical for an industrial environment, but could turn out to be the most practical method in a decade or so, with the emergence of faster hardware, more sensitive photo-receptors, and higher discrimination timing devices.

2.2 Solid Modeling

We have so far seen how to obtain the raw data for the scene. However, the scene description process does not stop there as we still have to:

1. Combine the data from the several views,
2. Convert that information to a form suitable for a computer representation.

It turns out that these two tasks are closely interrelated. For completeness, we present in this section a review of Solid Modeling, the field of research which deals with item 2 above. Authoritative reviews on the subject can be found in (Requicha, 1980; Requicha, 1983).

The issues involved in choosing a representation scheme are storage and computational requirements, ease of input, and the versatility of the representation for executing various algorithms. Unfortunately, these issues are often conflicting, which is why many schemes are in use today. Representations can be classified into a hierarchy of models forming a continuum between structured (or compact ones) and enumerative (or redundant ones) (Hayward, 1986). The following survey proceeds from enumerative representations to structured ones.

2.2.1 Volumetric Occupancy Schemes

2.2.1.1 Voxel Lists

Each volume element is represented by one logical bit, indicating whether the element is full or empty. This method is simple but very storage intensive since it makes no use of object coherence. It is usually inadequate for scene modeling.

2.2.1.2 Octrees

Octrees are a compact form of voxel lists which makes use of object coherence through a recursive decomposition of space (Meagher, 1982a; Meagher, 1982b). Octree encoding imposes a sorting on the 3-D space in both space and size, speeding up many operations. Since only one primitive, the cube, is used, most processing tasks bear some uniformity. Finally, octree computations naturally lend themselves to parallel processing.

Still, the construction of octrees is computationally expensive, and much memory is required to store the abundance of pointers. Alternate ways of encoding octrees alleviate the latter problem but complicate the algorithms that operate on them. Finally, the octree representation is extremely dependent on position and even more so on orientation, making it highly non-unique.

The segment representation is an interesting volumetric representation variant. It was introduced by Martin and Aggarwal (Martin and Aggarwal, 1983) and corresponds to a double discretization of the scene into equally-spaced segments. This representation is about as space consuming as octrees are, but is much more intuitive and less orientation dependent. It is easily constructed from a series of photometric views and is also easily converted to a surface representation. A major disadvantage of this

scheme however is its poor graphical appearance on a display due to its lack of both conciseness and structure.

2.2.1.3 Cell Decomposition

Cell decomposition comprises representations that decompose the space into instances of a chosen primitive, such as tetrahedra (Faugeras et al., 1984), spheres (O'Rourke and Badler, 1979), parallelepipeds (Kim and Aggarwal, 1986), or segments (Martin and Aggarwal, 1983). Unlike octrees, these representations do not build a hierarchical description. They have been found to be useful for many applications. For example, spheres are invariant under projections and this property is useful for fast graphical display. A tetrahedral representation is easily built with a general-purpose three-dimensional Delaunay Triangulation algorithm (Preparata, 1985, Chapter 5). Parallelepipeds have found applications in robotic path planning (Lozano-Pérez and Wesley, 1979).

2.2.2 Boundary Representations

Boundary representations (B-Rep's) are a family of representations where objects are made up of faces which are bounded by edges, themselves bounded by vertices. It is a well-behaved generalization of the wireframe representation (See next subsection). Several facial primitives can be used: triangles, general planar polygons, quadtrees (Samet, 1984) or parametric surfaces (Faux and Pratt, 1979).

B-Rep's make it possible to concisely represent arbitrarily complex shapes and they are well suited to computer graphics applications. Regrettably, it is difficult to interactively edit a B-Rep model that contains a large number of faces, in part because they encode local information which bears little relation with the global structure of

the model. The reader interested in a more detailed discussion of each primitive's advantages and disadvantages should refer to (Requicha, 1980).

2.2.3 Wireframe

The wireframe representation is simple and compact. It only encodes objects' vertices and edges. Many application programs already exist for computer graphics and for fast, interactive use. Unfortunately, it is ambiguous in three dimensions and can lead to the creation of nonsense objects. For this reason, it is inadequate for the internal representation of an object in a solid modeler.

2.2.4 Constructive Solid Geometry

The difficulty that humans experience entering or editing a B-Rep model have led to the development of Constructive Solid Geometry (CSG). CSG represents a large step in conciseness from B-Reps. It decomposes the scene into a set of simple global primitives (usually cubes, wedges, fillets etc...). It is thus a high level description. Note that CSG is very different from the cell decomposition representation because in the latter, the shape of the cells is arbitrarily chosen ahead of time and bears no relation to the high-level shape of the sub-objects.

CSG is well-suited to being manipulated either by a human or by an algorithm, whether for purposes of creation or of alteration. CSG can represent large classes of objects and supports Boolean operators. However, free-form surfaces are hard to represent with CSG, and conversion to other representations is difficult. Finally, we believe it is less suited to automatic scene description since global descriptions are harder for machines to generate.

2.2.5 Other Representations

There are many other representation schemes, including popular ones such as Sweep Representations (Agin and Binford, 1976) and Skeletons (Udupa and Murthy, 1977), or more obscure ones such as Generalized Blobs (Mulgaonkar et al., 1982) or Prism Trees (Faugeras and Ponce, 1983). We believe none of these offers much promise for the task of scene description as they either lack generality or ease of manipulation.

2.2.6 Solid Modeling: Conclusions

We gave a brief survey of solid modeling techniques. The most common ones used by solid modelers are CSG, B-Rep's and to a lesser extent, octrees. We saw that B-Rep's and octrees are more versatile and that CSG is well suited to human input. Because of the different advantages offered by each representation, solid modelers commonly store several representations in parallel, picking the one that is most suited to the particular operation to be performed.

2.3 Chapter Summary

We reviewed current methods of range data acquisition and solid modeling techniques. for the purposes of building a description of a scene.

Based on these techniques, the ASDT must tackle the task of multi-view integration. In the next chapter, we give a background review on that task and set the stage for the algorithms we describe in succeeding chapters.

We will see that the assumptions about the data acquisition techniques play an important role on data integration, and that the applicable algorithms directly depend

on the solid modeling technique one chooses.

Chapter 3

Multi-View Integration: A review

3.1 Data Integration

3.1.1 Finding the Inter-Frame Transform

As we saw in Chapter 1, the different views must be merged into a unified three-dimensional model, a process which we called data integration. The first task of multi-view integration is to find the geometric transform between the different views. This step depends in part on the assumptions one is willing to make regarding the environmental setup.

The simplest case occurs when the range finder's coordinate frame is registered with respect to the scene, and this is true regardless of the actual agent of movement. For example, the range imaging device can be kept stationary while the object(s) of the scene are rotated on a monitored motor-driven turntable (Faugeras and Pauchon, 1983; Bhanu, 1984). If the scene is too large or bulky to be easily displaced, one can equivalently elect to displace the imaging device instead.

An alternative strategy is to position fiducial marks around the scene. By identifying the fiducials across the several views, the inter-view transformation parameters are obtained through the solution of a least-squares problem (Potmesil, 1979). The identification of the fiducials is itself an instance of the correspondence problem (See Section 2.1.2.5), but the problem is normally easy to solve because of the highly-structured nature of fiducials.

In some cases, one can also resort to operator-assisted feature identification. For example, Hasegawa (Hasegawa, 1982) manually guides a laser spot through identical feature points across a series of views, while a triangulation range finder determines the coordinates of these features. Hence, correspondence is unambiguously established. Operator feedback is in the form of a superimposition of the laser spot on a video image of the scene.

Finally, one can estimate the inter-frame transform automatically by determining which one minimizes feature differences between overlapping parts of the object. This is done by evaluating a *matching* cost function for a set of chosen evaluation points, subject to a number of constraints. The evaluation points are usually image features, the cost function is based on local similarities of those features, while the constraints are that adjacency relationships between features are respected and that the transform be a rigid motion.

In (Ferrie, 1986), intensity image windows taken around distinguished surface points are correlated and consistency is verified using a relaxation labeling procedure. In (Herman, 1985), trihedral polyhedral vertices are matched using geometrical and topological constraints, the latter with a Waltz-like (Waltz, 1975) procedure. Potmesil (Potmesil, 1983) performed a heuristic search in the transformation parameter space, based on positional, orientation and curvature differences between points of maximal curvature. Chen and Medioni (Chen and Medioni, 1991) used a similar technique, minimizing

distances from points to tangent planes. Parvin and Medioni (Parvin and Medioni, 1991) used a multi-layer neural network, which computes surface matching costs using geometric and adjacency constraints between surfaces.

These computational approaches do not rely on external artifacts for their solution. They are then more general, although computationally expensive. Nevertheless, because they necessitate the early computation of useful features, their overall cost is lessened in view of the savings they introduce for the (eventual) later processing stages.

3.1.2 Merging the Several Scenes Together

After the camera position is precisely determined, we must merge the views together. This consists in finding a description for the scene in a world-coordinate frame, starting from a collection of views which are all expressed in their own viewer-coordinate frame. The methods reviewed below show that the algorithm used to perform the task depends in a large part on the representation scheme we adopt.

3.1.2.1 Shadow Intersection Merging

The most common method of merging the scenes is the "shadow intersection" method. This method is based on the assumption that the various scenels in the scene can be unambiguously separated from the background. This is often an artifice but it leads to simple algorithms and it can be used with both intensity and range images. The scenels then project into "blobs", or silhouettes, on the image plane. By back-projecting these silhouettes into space, one creates semi-infinite conic sections (or semi-infinite prisms if the orthographic projection is used) whose base is on the image plane. This process is equivalent to assuming that every occluded area of the scene

is "full", namely that the occluding scenel occupies that area. Similarly, it is as if a "shadow" was extended from the scenel to the image plane and from the scenel infinitely far behind it.

As we obtain additional information from subsequent views, these shadows are "shrunk" to their maximal plausible size, namely their common intersection. Figure 3.1 shows a simple shadow intersection example illustrating the principle. The process stops when "enough" views are taken, presumably when the whittled-down shadow exactly matches the scenel being modeled. This can either be decided by the operator, or some measure of convergence can be defined. One such measure is to stop when the volume reduction ratio introduced by additional views drops below a certain value.

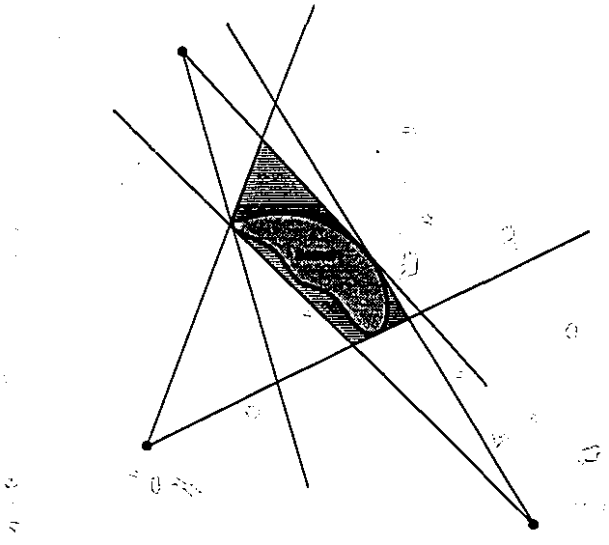

Figure 3.1 The shadow intersection method. A two-dimensional scenel is acquired from three viewpoints. Its tangents of visibility determine semi-infinite shadow cones whose intersection determines the resulting model.

Octrees are often used in conjunction with the shadow intersection method. For example, Chien and Aggarwal (Chien and Aggarwal, 1986) built the octree of an object seen from three orthogonal *intensity* views. Veenstra and Ahuja (Veenstra and Ahuja,

1986) extended the method to up to 13 standard views. Hong and Schneier (Hong and Shneier, 1985) presented a more general method that makes no assumption on the view positions. The already-built octree is projected onto the image plane of each new view, and is intersected with the scenel's projection for that view. Noborio *et al.* (Noborio et al., 1988) later sped up the method by performing intersections in three-dimensional space instead. Similar techniques are presented in (Potmesil, 1987; Srivastava and Ahuja, 1990).

Alternate representations can also be used with the shadow intersection method, influencing the intersection algorithms. For example, Martin and Aggarwal (Martin and Aggarwal, 1983) use the segment representation to merge shadows, using the scan-lines of the last view as the arbitrary direction of orientation for the segments. Srinivasan *et al.* (Srinivasan et al., 1989) discretized the blobs appearing in the image plane along arbitrary stacked planes, such that the eventual intersection tests be between two-dimensional polygons.

The references given just above all used intensity images. With such images, object concavities (such as the inside of a cup) cannot be recovered, since the method only considers object projections, or silhouettes. When used with range images however, the method does not suffer from this defect and it also converges faster. In (Connolly, 1984), Connolly constructed a quadtree in each image plane of a set of synthetic range views and used ray-tracing techniques to assimilate the quadtrees into a global octree. In (Stenstrom and Connolly, 1986), line segments are extracted from range images, and are then swept arbitrarily far from their respective image plane, thus constructing a wireframe model for each range view. The wireframe models are "fleshed out" using the Markowsky and Wesley algorithm (Markowski and Wesley, 1980), and the resulting polyhedra are intersected to yield a solid polyhedral model. In (Wang and Aggarwal, 1989), relative depth is obtained by a structured lighting technique, and

is combined with passive sensing in yet another variation of the shadow intersection method. The authors claim that the construction process is thus made more accurate for a fixed number of views.

In summary, the shadow intersection method is a general and simple image integration paradigm which works with both intensity and range images, with the proviso that concavities can only be acquired with the latter image type. It also requires a volumetric representation scheme since it encodes the inside of the object (or what is presumed to be its inside) rather than its surface.

The need for an a priori separation of scene from background restricts its applicability. This separation is often hard to establish automatically, since object-background separation is an instance of the unsolved general segmentation problem. Further, the separation cannot be obtained nor can the sensor rotate around the scene in some instances. Imagine the case where the sensor is limited to moving around and scanning the inside a cavity or a room. Such a scene has no "background" to speak of and therefore does not admit a shadow intersection method.

We will in the next sections turn our attention to methods that encodes *surface* rather than *volume* information. Since the acquired data is itself in the form of surface information, we believe that B-Rep's are a more natural representation for the models built by the ADST. We distinguish two main methods, surface- and point-based merging methods.

3.1.2.2 Surface Merging

In surface merging, individual views are processed as sets of surface components and are then merged together. This technique clearly uses a boundary representation. For each new view, four processing steps are taken:

1. Identification of surfaces.
2. Transformation of the surfaces into a common reference using the known inter-frame transform.
3. Surface overlap elimination.
4. Surface parameter update.

The first identification step may be done at a symbolic or at a data level. Symbolic level merging is normally performed within the three-dimensional analysis paradigm of Figure 1.3: one first detects features in the individual views, then these features are matched across the views with a matching algorithm similar to those discussed in Section 3.1.1.

For example, Ferrie (Ferrie, 1986) uses shape-from-shading techniques to identify and to segment surfaces based on differential-geometric properties, Dane (Dane, 1982) fits quadric patches using a region-growing approach, and Parvin (Parvin and Medioni, 1991) detects range data discontinuities using zero-crossings of the second derivative. These approaches build a composite graph, whose nodes are the identified surfaces and arcs indicate adjacency relationships.

Surface identification can also be done at a data level. In this case, individual views are not "processed" and only a low-level description is arrived at. This is more attuned to the ASDT least-commitment paradigm of Figure 1.2, where we do not seek a trimmed high-level representation from which it is all but impossible to recover the original data. Rather, the choice of the final structured representation is left up to the application program that subsequently uses the ASDT's output.

For example, Potmesil (Potmesil, 1983) obtains range data by projecting grids of orthogonal lines on the scene, and matching grid junctions across the views. He

obtains a network of bicubic parametric patches which are used as the basic surface elements. Using a triangulation range finder, Bhanu (Bhanu, 1984) simply aggregates range data into small planar facets.

The surface descriptions are then transformed to a common reference using the inter-view transform, which is either known or estimated as described in Section 3.1.1. After this step, the actual surface merging is performed. One of the difficulties consists in eliminating the overlapping sections, namely the scene parts which appear in more than one image. This process also depends on the surface representation employed.

In (Potmesil, 1983) overlapping surface segments are reparametrized by mapping a new parametric grid on them. Soucy (Soucy, 1993) performs a similar reparametrization based on range images, except that the information coming from each view is weighted by an estimate of its quality. In (Ferrie, 1986; Parvin and Medioni, 1991), the composite graph is updated by recomputing the boundaries of the surfaces, while in (Dane, 1982) the quadric parameters are modified based on the newly-acquired information.

3.1.2.3 Point Merging

In the last section, we saw that a boundary representation can be extracted from a scene at the data level, and that this paradigm was consistent with the ASDT's description. In general however, range elements or *rangels*, are in the form of discrete three-dimensional points, as seen in Chapter 2[1].

Many parametric methods exist for constructing smooth surfaces passing through or near a collection of boundary points (Faux and Pratt, 1979). The simplest method

[1]There are exceptions to this, as some techniques use the close proximity of rangels to aggregate them into *lines* for further processing (Oshima and Shirai, 1983).

however does away with the smoothness assumption and approximates the desired surface with a collection of planar facets. For any triplet of adjacent points, we approximate the surface by a face which passes through those three points. Processing the point set is hence reformulated as defining adjacency relationships between the data points. This is a three-dimensional connect-the-dots problem, as mentioned in Chapter 1. The goal is to find the "most natural" *triangulation* of the data set, based on a combination of contextual information and of heuristic measures. More formally, we wish to build a *connectivity graph* G spanning the set V of data points, such that G embeds a kind of proximity graph in a Riemannian, or geodesic sense.

In the simplest case, suppose we acquire n points that are known to lie on the surface of a connected object that contains no hole: topologically, we say that the object is homeomorphic to a sphere. We seek a triangulation, or a *spanning, maximal, 3-connected, planar* graph on V. The collection of faces of the triangulation forms the surface of a simple *polyhedron*. In Appendix A, we derive a numerical formula for $\phi(n)$, the number of admissible such graphs:

$$\phi(n) = \frac{(4n-11)!}{(3n-6)!}\binom{n}{2}. \tag{3.1}$$

Clearly, the enormous growth rate of the above formula precludes all attempts at enumeration[2]. Thus, clever algorithms must be designed to prune the candidate graphs.

O'Rourke (O'Rourke, 1981) defines the polyhedron of minimal area as the most natural model for the set V. He also gives a heuristic "greedy" algorithm to compute a good approximation for such a polyhedron. The algorithm selects the convex hull of V as an initial approximation to the desired shape. The shape is then incrementally

[2] $\phi(10) > 400$ million and $\phi(120) > 10^{304}$. Of course, the overwhelming majority of these graphs represent non-simple polyhedra, namely polyhedra with self-intersections.

"carved" by inserting its internal points into its boundary. The selection of the internal points and how to insert them into the current shape is based on local criteria only, yielding an $O(n^3)$ algorithm.

An alternative method uses the three-dimensional Delaunay triangulation as a seed structure (Boissonnat, 1984). Note that triangulation here is a misnomer, since the structure is actually a *tetrahedrization* of the convex hull of V. The tetrahedra are then eliminated based on a local geometrical criterion that measures how "internal" to the current shape the tetrahedron is. As with O'Rourke's method, the process stops after all the internal points are on the boundary of the shape. The final result is a cell decomposition representation. Its boundary is a triangulation of the shape.

Hoppe (Hoppe et al., 1991) reconstructs surfaces from very dense data by first constructing a *Riemannian Graph*, a special type of proximity graph he derives from the Euclidean Minimum Spanning Tree. Good results are given using synthetic data.

Despite its name, the Riemannian graph, like the Delaunay Triangulation, is a proximity graph based on Euclidean distance considerations only. One problem with such graphs is that near points in a Euclidean distance sense may not be neighbors in a Riemannian, or geodesic sense. In other words, two vertices may be near in the three-dimensional space, and far from each other if one was to travel along the two-dimensional surface on which they lie. Yet a Euclidean proximity graph will incorrectly join such vertices. Of course, the geodesic distance cannot be computed, for if it were, we would know the underlying surface.

The two-dimensional example of Figure 3.1 illustrates the above problem. The shape shown in (a) is Delaunay-triangulated in (b). The two-dimensional analogue of Boissonnat's algorithm will remove triangle B instead of triangle A from the initial triangulation, thus obtaining the result shown in (c). Since the convex hull is contained in

the Delaunay Triangulation, the two-dimensional analogue of O'Rourke's algorithm also yields the same result: Because creating A increases external perimeter by a greater amount than creating B does, the shape of (c) is again favored. Yet, the preferred perceptual connectivity is shown in (d).

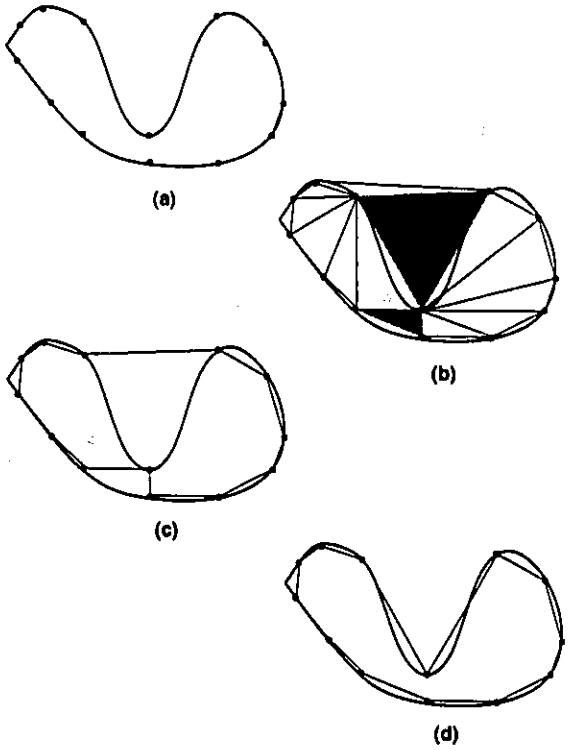

Figure 3.2 (a) A two-dimensional closed shape with a set of boundary points on its contour.
(b) The Delaunay Triangulation on those points.
(c) The graph obtained by either of (O'Rourke, 1981; Boissonnat, 1984)
(d) The desired connectivity graph.

3.1.2.3.1 Constrained Point Merging The connectivity problem we just outlined does not lie with the particular algorithms, but rather with the ill-posed nature of the connect-the-dots problem. A "cloud" of points does not make a well-defined object. Contextual information is needed to help find the desired shape.

For example, Boissonnat (Boissonnat, 1982) developed an algorithm for triangulating the surface of an object provided that a label is available at all surface points, indicating whether or not the surface is locally convex. However, the convexity boolean is derived from the signs of the principal curvatures, which are second derivative features. Because derivative-taking implies the prior knowledge of connectivity, the label assumption only points to the existence of a "chicken-and-egg" problem.

A more realistic case occurs when the data is organized into a set of stacked planars sections, and each section contains one or more polygonal contours. This form of organization is typical of tomographic data, where each plane corresponds to an acquisition scan-line. The contours are then used as subgraphs of the desired connectivity graph. Subsequent linking of adjacent contours yields a triangulation of all points, where each triangle is composed of two adjacent points from a given contour, and one point from an adjacent contour. Still, linking adjacent contours is a constrained connect-the-dots problem. Choosing the contour-to-contour connections was shown in (Keppel, 1975) to be reducible to finding a shortest-length path in a directed graph, where the weights of the graph arcs optimize the heuristic criterion of choice. Keppel chooses to minimize the volume of the polyhedra bounded by adjacent contours and by the contour-to-contour triangles, thus minimizing the overall object volume. At any stage of the triangulation, only two choices are present: either triangulate "upwards" (with two vertices of the newly-created triangle on the upper scan-line) or triangulate "downwards"(with the two same vertices now on the lower scan-line). Fuchs (Fuchs et al., 1977) gives a similar, but more complete method which minimizes the object's surface area. Wang and Aggarwal (Wang and Aggarwal, 1986) sped up Fuchs' method by using the well-known graph search A^* algorithm (Hart et al., 1968). In (Boissonnat, 1988), a different method is used: contour-to-contour triangular transitions are built with the three-dimensional Delaunay Triangulation. This method naturally handles the case of disconnected contours (*i.e.* more than one

contour per planar section).

Another type of contextual information is the knowledge that a set of *edges* must belong to the connectivity graph. This happens when the data consists of *stereo edges*, rather than stereo points, and the surface triangulation is then constrained to include these edges (Boissonnat et al., 1988).

The last example supports our claim of Page 15 that much is to be gained using the knowledge of the data acquisition procedure right at the data integration stage. In Section 1.4, we saw that the line-of-sight segments of triangulation range finding are known to entirely lie in free-space, and that this knowledge can be used to discriminate between shapes based on the consistency of physical observations.

For example, Faugeras and Pauchon (Faugeras and Pauchon, 1983) first built closed planar contours using the line-of-sight information. As above, they assume the data is partitioned into planar sets of points. However, the planar polygons are not known a priori and must be built from the original point set before an algorithm such as Keppel's is applied. By solving the connectivity problem along parallel contours independently, the connect-the-dots problem is made two-dimensional. For every plane, the goal is to build a *simple* polygon (a spanning cycle in graph terminology) from the set of points associated with that plane.

This two-dimensional connect-the-dots problem has also been studied in isolation. For example, O'Rourke *et al.* (O'Rourke et al., 1987) solved the problem by building a *minimal spanning Voronoi tree*, while Edelsbrunner *et al.* (Edelsbrunner et al., 1983) proposed an algorithm based on a generalization of the convex hull.

Rather, Faugeras and Pauchon's algorithm used contextual information as follows: For each acquisition plane, the sensor incrementally rotates around the object. For a given plane, two criteria are used to establish point connectivity. For points acquired

from the same sensor position, or view, adjacency is established following the angular adjacency of these points with respect to the sensor (Refer to Figure 1.5). For points acquired from different views, adjacency is established depending on whether the line-of-sight of a given data point intersects the edge joining the points of another view. This last condition fulfills the opacity condition (See Section 1.4).

Alevizos *et al.* (Alevizos et al., 1987) perfected the above idea and gave an algorithm for building a simple polygon starting from a set of points, where each point lies at one end of a finite segment. They prove that the solution, when it exists, is unique, and they obtain it in the optimal worst-case asymptotic complexity of $O(n \log n)$ time, for an n-sized input. When the data comes from a physical source, the existence of a solution is guaranteed (up to sensor noise). The algorithm first builds the convex hull of the data points. In that respect, it resembles both O'Rourke's and Boissonnat's three-dimensional algorithms. Yet it differs in the way the subsequent "carving" of the convex hull is performed. Here again, the intersection of acquisition segments with edges of the polygonal contour is the decision criterion used in breaking adjacency relations.

An attractive feature of both Faugeras and Pauchon's and Alevizos *et al.*'s algorithms is that the graph construction does not call for closeness measures in the Euclidean metric sense, as the Delaunay triangulation does, nor does it require additional heuristics or assumptions about the surface texture. The former method assumes that the data is organized as a set of separate views, while the latter also does away with that assumption.

In the next chapters, we will extend the ideas contained in these algorithms to the general three-dimensional connect-the-dots problem. Here, we give a simple graphical example to illustrate the nature of the problem. Imagine an object with a deep concavity is scanned with five acquisition segments.

Equation 3.1 states that 10 different planar maximal triangulations can be drawn on these vertices. The maximal triangulation on five vertices has $3*5-6=9$ edges, but there are only $\binom{5}{2} = \phi(5) = 10$ possible edges. Hence, each graph is differentiated by which edge is "missing" from its edge set. We use this fact to label graphs in the following discussion. For example, (A, B) is the graph with no edge joining A and B, but with an edge joining every other vertex pair.

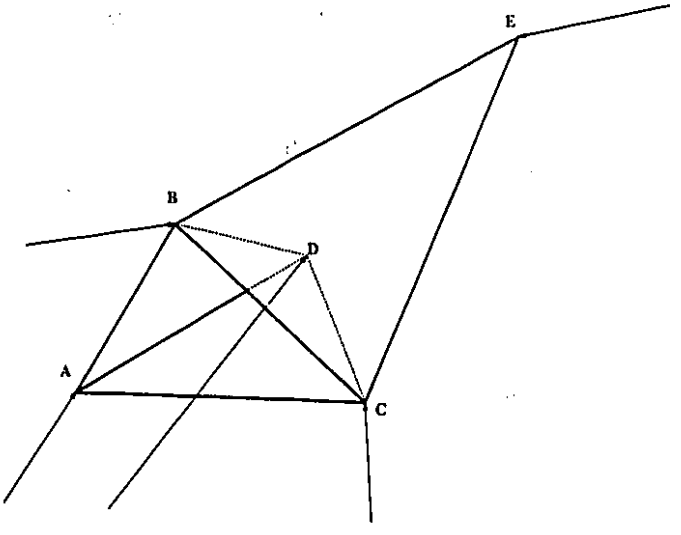

Figure 3.3 A simple example with 5 vertices and their acquisition segments. Imagine the graph shown in the figure is the most "perceptually-consistent" graph for the underlying object, with point D lying at the bottom of the object's concavity.

Figure 3.3 shows graph (D, E) superimposed on the set of vertices and their segments. The model based on graph (D, E) is consistent in the sense given in Chapter 1 since, without any further information, the object could very well be exactly represented if each graph face was replaced by a planar facet. On the other hand, any graph with face (A, B, C) in its face set is not consistent since the acquisition segment of D intersects that face, thus violating the opacity assumption.

By enumerating all ten possible graphs, we see that only four are consistent with the

data acquisition, the other six containing face (A, B, C) in their face set. The four consistent graphs are $(A, B), (B, C), (A, C)$, and (D, E). Figure 3.4 shows such an alternate graph.

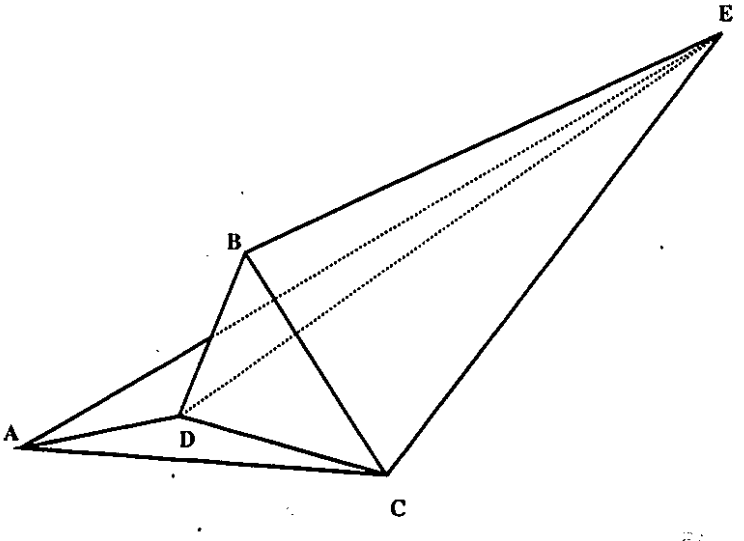

Figure 3.4 (A, B) is an alternate consistent graph for the 5-vertex data set.

This simple example shows that contrary to the situation in the two-dimensional case, graph reconstruction of three-dimensional objects based on vertex sets and their acquisition segments does not admit a *unique* solution.

As well, note that the implicit visibility information provided by the acquisition segments is only that there exists an ϵ-diameter cylinder of free-space around each segment. Nevertheless, the question arises as to whether, or when, such input data is sufficient to construct a consistent least-commitment connectivity graph using purely geometrical tests.

3.1.3 Chapter Summary

We have reviewed the state-of-the-art in merging several views of a three-dimensional scene. We saw that the first step, finding the inter-frame transform between the views, is generally a semi-automated process.

Several techniques exist to merge views once the geometric correspondence is known. The shadow intersection method is the most simple and works well, but is restricted to *objects* with a clear object-background separation. It also requires the use of volumetric representations.

We also described some surface merging techniques, which are akin to the three-dimensional analysis techniques used in Computer Vision (Refer to Figure 1.3).

Finally, we reviewed point merging methods, which follow the ASDT paradigm (See Figure 1.2). Since point-merging methods do not perform three-dimensional analysis as described in Chapter 1, they implement the least-commitment principle introduced in that chapter.

These methods also use surface-based representation, but they first construct a graph by merging the sensor information purely at a data level, namely by inferring point-to-point connectivity. This connectivity is essential for further differential-geometric processing, since the latter requires the knowledge of neighbor relationships between points.

In most of the point-merging approaches, the graph is drawn from Euclidean proximity considerations only. This assumption breaks down when the data is sparse with respect to the surface concavities, which we illustrated with a simple example. Others have introduced the idea of using the implicit additional knowledge given by the sensor acquisition segments' paths. In (Alevizos et al., 1987), this idea was proven to be

powerful enough to unambiguously construct the graph of an object homeomorphic to a (two-dimensional) disk.

In this work, we investigate whether the equivalent statement can be made for three-dimensional graph reconstruction in general. Namely, can three-dimensional surface graphs of real objects be reconstructed solely from the geometry of surface-segment intersections?

The next chapter presents an algorithm for *incrementally* merging sets of data, each of which has a common acquisition center. The following chapter presents an algorithm for *globally* merging data without any assumption with respect to data organisation.

Chapter 4

An Incremental Merging Algorithm

4.1 The Acquisition Frames

In this chapter, we present a method to incrementally merge a series of range *views* into a graph embeddable on a sphere. We do not assume that the points coming from different range views are organised along parallel or non-intersecting lines, as was done in most of the data integration methods reviewed in Chapter 3. The reasons for doing so are two-fold.

First, obtaining such scanlines may be an undesirable restriction in some environment configurations. For example, we may wish to explore the environment using sensors positioned at arbitrary positions and orientations. This may help greatly for exploring certain types of concavities, and for the general problem of navigation in an unknown environment.

Second, although the scan-line approaches described in Section 3.1.2.3 (Keppel, 1975; Fuchs et al., 1977; Wang and Aggarwal, 1986) adequately solve the two-dimensional surface connectivity problem along one dimension, it does resort to using a Euclidean metric in the orthogonal direction.

We will assume in the following that we have at our disposal an arbitrary but bounded number of range images, whose position, orientation, and resolution are also arbitrary. Our objective is to merge those images in order to build a **surface connectivity graph** G. The vertex set of G is the set of data points (See figure 4.1).

Notation: We will adopt the following notational conventions:

- $\forall G$ s.t. G is a graph, $E(G)$ is the edge set of G.
- $\forall G$ s.t. G is a graph, $V(G)$ is the vertex set of G.
- $\forall G$ s.t. G is a graph, $F(G)$ is the set of faces of G.
- $\forall G$ s.t. G is a graph, $G = \langle A, B \rangle$ means that $A = V(G)$ and $B = E(G)$.
- $\overline{x,y}$ denotes the edge joining x and y.
- $(x_1, \ldots, x_n)$ denotes the graph cycle $x_1, \ldots, x_n, x_1$.
- $\overbrace{x_1, \ldots, x_n}$ denotes the closure of the $(n-1)$-dimensional simplex whose vertices are $\{x_1, \ldots, x_n\}$, minus those vertices.
- $\setminus$ is the usual set difference operation.

4.1.1 Definitions

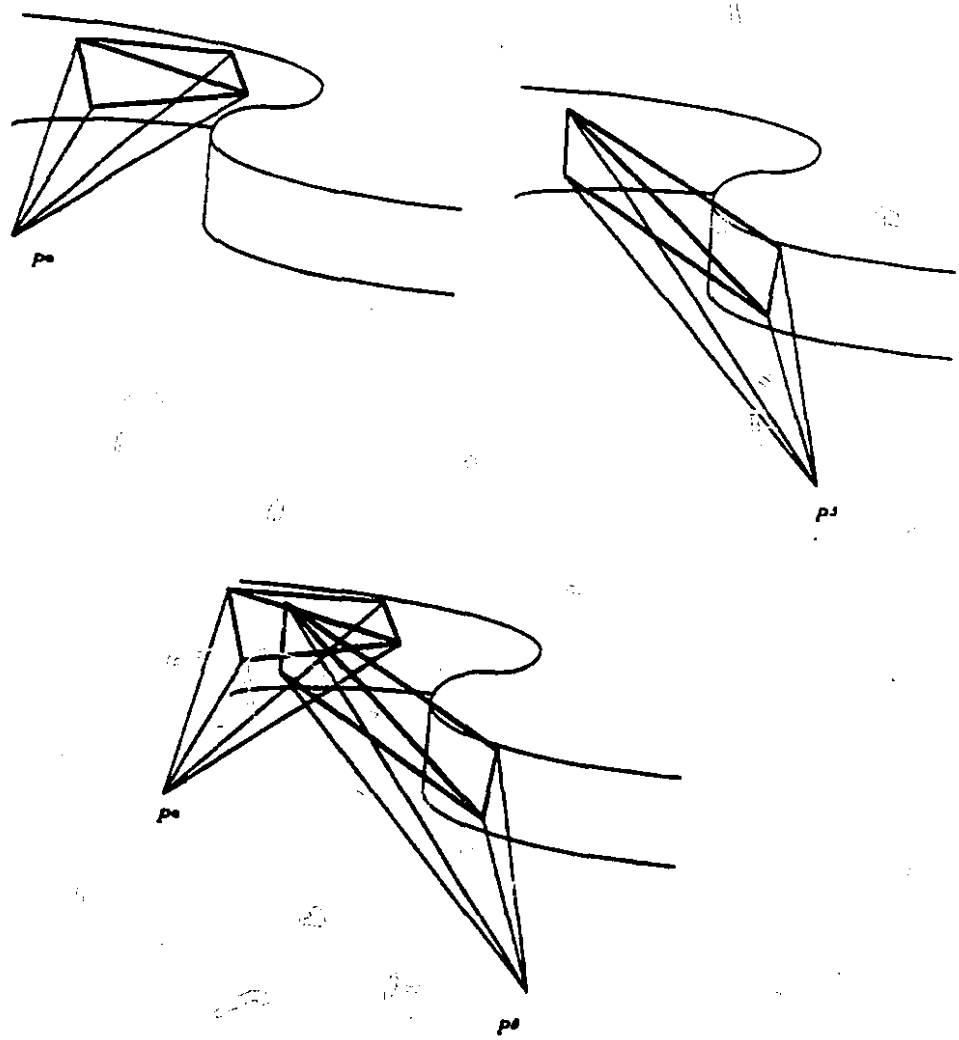

Figure 4.1 a)The inferred surface graph for frame $\mathcal{F}^{\alpha}$.
b)The inferred surface graph for frame $\mathcal{F}^{\beta}$.
c)The superposed graphs from both frames. The resulting graph remains to be found.

Definition 1 *A* **surface connectivity graph** *for a set V of points assumed to lie on a two-dimensional surface $\mathcal{M}$, is a graph whose vertex set is V and whose faces are embedded on $\mathcal{M}$.*

Definition 2 *A* **frame** $\mathcal{F} = \langle P, V, S \rangle$ *is a triplet consisting of the following geometric entities:*

- *A point P in free space.*
- *A set $V = \{x_1, \ldots, x_N\}$ of surface points such that $\forall i, j \in \{1, \ldots, N\}, \exists \lambda \geq 0, \overrightarrow{Px_i} = \lambda \overrightarrow{Px_j}$.*
- *A set $S = \{s_1, \ldots, s_N\}$ of directed line segments such that $\forall i \in \{1, \ldots, N\}, s_i = \overrightarrow{Px_i}$, with s_i lying entirely in free space.*

A *frame* embeds the information about $\mathcal{M}$ that can be obtained from a single viewpoint. In the case of range imaging, P is the imaging system's fixed lens center, S is the set of acquisition segments, no two of which are collinear, and V is the set of data points expressed in world coordinates. Alternately, P could be the fixed end of a thin mechanical sensing link, S the set of successive lines of action of the links, and V the set of successive positions of its moving end. We will refer to P as the **center**, or **acquisition center**, of the frame, to the elements of V as the **data points** of the frame, and to the elements of S as the **segments**, or **acquisition segments**, of the frame.

We assume in the following that there are ν frames to be merged. For each frame, the frame itself as well as its attributes are superscripted with Greek letters: For a given frame α, we'll write $\mathcal{F}^\alpha = \langle P^\alpha, V^\alpha, S^\alpha \rangle$. The superscript may be omitted when there is no risk of confusion. Finally, the cardinality of each V^α is assumed to be n^α, and $\sum_{\alpha=1}^{\alpha=\nu} n^\alpha = N$. Hence N represents the total size of the input.

Definition 3 *A* **visibility graph** *of frame $\mathcal{F}^\alpha$ is a graph whose vertex set is V^α. The graph will be denoted G^α, and its face set as embedded on $\mathcal{M}$ will be denoted F^α.*

The visibility graph of a frame corresponds to the partial connectivity that can reliably be inferred from a unique viewpoint. We will show in the next subsection that one can easily construct a visibility graph for an isolated frame $\mathcal{F}^\alpha$. To that end, we first formally define the notion of *validity* for a surface connectivity graph.

Definition 4 *A visibility graph G^α will be called* **2 1/2-valid**, *or simply* **valid** *if and only if it is:*

- *A triangulation.*
- **2 1/2-Consistent.** *G^α is said to be 2 1/2-consistent, or consistent, if and only if:*
  - *No acquisition segment intersects the closure of any face of G^α, except at its vertices. Formally,*

$$\forall f = (x_1, x_2, x_3) \in F^\alpha, \forall s \in S^\alpha, (s \cap \widehat{x_1, x_2, x_3}) = \emptyset. \tag{4.1}$$

  - *Each face of G^α can be linked to the acquisition center P^α of the frame, such that the closure of the resulting* **visibility tetrahedron** *contains no data point, except at its vertices. Formally,*

$$(\forall f = (x_1, x_2, x_3) \in F^\alpha), \widehat{P^\alpha, x_1, x_2, x_3} \cap V^\alpha = \emptyset. \tag{4.2}$$

The first validity condition relates to the fact that the graph can be embedded on a given surface without self-crossings, so the faces of the graph form a *covering* of the surface. Formally, we want G to be embedded on a compact, connected, and

orientable two-dimensional surface $\mathcal{M}$. It turns out that such a surface $\mathcal{M}$ can always be triangulated (Aleksandrov, 1956).

To enforce the planarity condition, we impose the additional restriction that the Euler number of $\mathcal{M}$ must be equal to 2. Hence the surface must either:

- Be closed and of genus 0 (a sphere).
- Have one boundary curve (a disc).

Any such surface can be triangulated in such a way that the resulting graph is isomorphic to a planar triangulation (Giblin, 1981). In summary, the first validity condition is that G be isomorphic to a planar triangulation.

The second validity condition, the consistency condition, ensures that the assumed connectivity is consistent with the data points, their respective acquisition viewpoints, and the assumption of object opacity.

The first consistency condition, (4.1), ensures that all concavities are accounted for. When it is fulfilled, no data point x can be visible from a given center P located in free-space, while its segment s, namely the line of visibility joining x to P, crosses a face of G. In other words, x cannot at the same time be both visible and invisible (hidden by a face) from P.

The second consistency condition, (4.2), ensures that all convexities (protuberances) are accounted for. When it is fulfilled, no data point x can "stand" between a face f of G and the acquisition center P. In other words, f, as seen from P, cannot at the same time be visible and obscured by x.

Hence, consistency ensures that the graph G uses all the information provided by the sensor segments, namely that they lie entirely in free space and that they are

terminated at the boundary of free space and solid space.

4.1.2 Building the Frame Visibility Graph

In this section, we show that for a particular frame $\mathcal{F}^\alpha$, one can easily infer a "natural", valid visibility graph.

Let E^3 be the three-dimensional world space, with its origin at P^α and with an orthonormal basis $\mathcal{B}$. Let S^2 be a sphere centered at P^α. $\forall y \in E^3$, let (θ_y, ϕ_y) be the polar coordinate pair of y around P^α with respect to $\mathcal{B}$. Let $\mathcal{P}$ be the polar angle transformation around P^α

$$\mathcal{P} : E^3 \longrightarrow S^2,$$

$$y \mapsto (\theta_y, \phi_y).$$

Let $\mathcal{P}^\alpha$ be the restriction of $\mathcal{P}$ to V^α. Let $V^{\alpha\prime} = \{x' | \forall x \in V^\alpha, x' = \mathcal{P}^\alpha(x)\}$. Let $G^{\alpha\prime}$ be the graph such that $V(G^{\alpha\prime}) = V^{\alpha\prime}$ and

$$\forall x \forall y \in V^\alpha, (\overline{x,y} \in E(G^\alpha)) \Leftrightarrow \left(\overline{\mathcal{P}^\alpha(x), \mathcal{P}^\alpha(y)} \in E\left(G^{\alpha\prime}\right)\right). \tag{4.3}$$

We can now state the following theorems:

Theorem 1 *Let G^α be any graph whose vertex set is V^α. Let $G^{\alpha\prime}$ be the graph on the image set of G^α by the polar angle transformation $\mathcal{P}^\alpha$ such that (4.3) holds. If $G^{\alpha\prime}$ is a planar triangulation, then G^α is valid.*

Proof: See Appendix A.7.1.

Theorem 2 *Let $\mathcal{Q}$ be a projection of E^3 onto a plane Π not passing through P^α and whose center of projection is P^α. Let β be the solid angle around P^α spanned by the elements of V^α. Let $V^{\alpha\prime\prime}$ be the image set of V^α through $\mathcal{Q}$.*

If $\beta < 2\pi$ steradians, $\exists\Pi$ such that Π intersects all the elements of S^α. Further, any triangulation $G^{\alpha\prime\prime}$ in Π of the mapped set $V^{\alpha\prime\prime}$ is valid.

Proof: See Appendix A.7.2.

In general, range images are acquired by sweeping the scene along two (sometimes orthogonal) directions u and v, and by sampling at some, possibly regular, intervals along those directions. Hence, the data points form a complete grid of parallelograms on the projection (focal) plane Π (See Figures 4.1a, 4.1b, and 4.9). From theorem 2, any triangulation on Π is valid. In particular, the following triangulation is straightforward to construct: Let $x^{i,j}$ designate the datum point acquired at the ith position along the u camera motion direction and at the jth position along the v camera motion direction. For all four-tuples $x^{i,j}, x^{i,j+1}, x^{i+1,j}, x^{i+1,j+1}$, we let the following edges belong to $E(G^\alpha)$:

$$\overline{x_{i,j}, x_{i,j+1}},$$

$$\overline{x_{i,j+1}, x_{i+1,j+1}},$$

$$\overline{x_{i+1,j}, x_{i+1,j+1}},$$

$$\overline{x_{i,j}, x_{i+1,j}},$$

$$\overline{x_{i,j}, x_{i+1,j+1}} \text{ or } \overline{x_{i+1,j}, x_{i,j+1}}.$$

Figure 4.9 on Page 88 illustrates this process. In section 4.2, we shall use the special properties of this arrangement to facilitate the process of merging the different frames together.

4.1.2.1 Graphs Drawn from the Trace of a Camera Path

The previous example showed that for a particular arrangement of the data points and an immobile camera, one can easily infer a "natural" connectivity graph without any computations. An interesting question would be the determination of whether other "easy" valid graphs can be obtained in cases where the camera moves about.

4.1.2.1.1 Sliding the Sensor along its Line of Support. We can make an immediate observation regarding the interpretation of P in definition 2. Suppose the camera lens center O moves to a location P_i along the segment s_i in such a way that the following holds:

$$\forall x_i \in V^\alpha, \exists \lambda > 0, \overrightarrow{P_i x_i} = \lambda \overrightarrow{P^\alpha x_i}. \tag{4.4}$$

Then the new frame $F^{\alpha\prime} = (P^{\alpha\prime}, V^\alpha, S^{\alpha\prime})$ is different from F^α only in the length (but not in the sense) of the elements of S^α and $S^{\alpha\prime}$. Since the lengths of the segments s_i are irrelevant to the derivation of theorems 1 and 2, the latter still hold for frame $F^{\alpha\prime}$. In other words, the actual locations of O are irrelevant to the graph construction process if the semi-infinite segments terminated at the data points all intersect in a common point P (See Figure 4.2). In the remainder of this Chapter, we will continue to refer to P^α as the center of acquisition of the frame, but the reader may wish to keep in mind that P^α could alternately be defined as any point in free space verifying (4.4).

4.1.2.1.2 Moving the Sensor along a Curve or Path. We now wish to investigate whether the definition of a frame can be relaxed even further, while still offering us the possibility of inferring "natural" and valid visibility graphs. Suppose

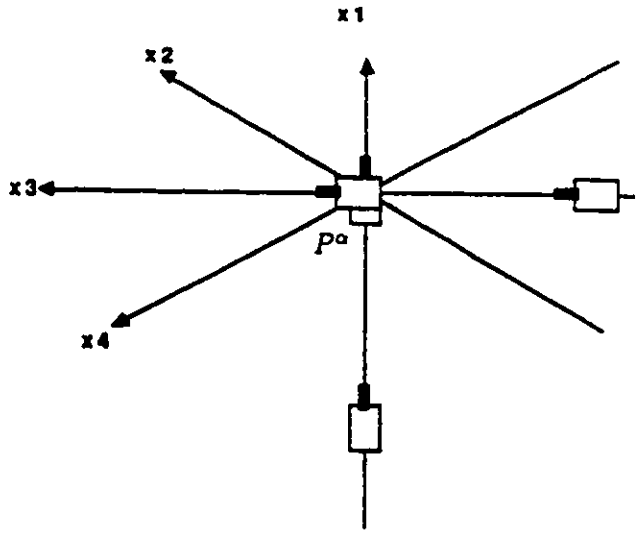

Figure 4.2 The actual camera locations are not important as long as all semi-infinite segments do intersect in a common point.

the acquisition center O moves along a path $\mathcal{C}$ and acquires a datum at N positions $P_i (i \in \{1, \ldots, N\})$, where i is an index increasing with the parameter of $\mathcal{C}$, and where the acquisition segments are all tangent to $\mathcal{C}$. Our goal is to find whether conditions on $\mathcal{C}$ can be enunciated, such that the visibility graph obtained by joining *adjacent* data points along $\mathcal{C}$ is valid.

Remark 1 *Suppose O follows the trace of a three-dimensional curve $\mathcal{C}$. Then the elements of S have the trace of $\mathcal{C}$ for an envelope.*

Proof: The elements of S are tangents to $\mathcal{C}$ and constitute a subset of the tangents of $\mathcal{C}$. Since a curve is an envelope of its tangents (Pogorelov, 1965, p. 35), it is also an envelope of a subset of its tangents. □

Suppose now that for each acquisition point P_i, x_i is its datum point, s_i is its acquisition segment, and r_i is the semi-infinite segment terminated at x_i and whose support is s_i.

We generalise (4.4) to the case where each $r_i (i \in \{1, \ldots, N-1\})$ intersects its neighbour r_{i+1} at a point P_i^{i+1}. Formally, we say that that there exists a relation $\prec$ on S, such that

$$\forall i \in \{1, \ldots, N-1\}, (s_i \prec s_{i+1}), \tag{4.5}$$

where $(s_i \prec s_j) \Leftrightarrow ((\exists \lambda_i, \lambda_j > 0), (\exists P_i^j \in E^3), (\overrightarrow{P_i^j x_i} = \lambda_i \overrightarrow{P_i x_i}) \wedge (\overrightarrow{P_i^j x_j} = \lambda_j \overrightarrow{P_j x_j}))$.

Obviously, every point $P_i^j (j = i+1)$ defines its own degenerate frame

$$\mathcal{F}^{ij} = \langle P_i^j, \{x_i, x_j\}, \{s_i = \overrightarrow{P_i x_i}, s_j = \overrightarrow{P_j x_j}\}\rangle. \tag{4.6}$$

with the degenerate triangulation $G^{ij} = \langle \{x_i, x_j\}, \{\overline{x_i, x_j}\}\rangle$.

From (4.6) however, every $x_i (i \in \{1, \ldots, N-1\})$ belongs to exactly two frames, namely to

$$\mathcal{F}^{(i-1)i} = \langle P_{i-1}^i, \{x_{i-1}, x_i\}, \{s_{i-1}, s_i\}\rangle.$$

and to

$$\mathcal{F}^{i(i+1)} = \langle P_i^{i+1}, \{x_i, x_{i+1}\}, \{s_i, s_{i+1}\}\rangle.$$

Hence,

$$\forall i \in \{1, \ldots, N-1\}, G^{i(i+1)} = \langle \{x_i, x_{i+1}\}, \overline{x_i, x_{i+1}}\rangle.$$

If we now set $G^\alpha = \bigcup_{i=1}^{N-1} G^{i(i+1)}$, G^α becomes a connected chain (see figure 4.3).

A legitimate question is whether G^α then is valid. Unfortunately, the chain structure of G^α does not constitute a very strong result since a valid connectivity graph *should*

be a triangulation. The chain being a degenerate triangulation, its faces are edges and the visibility tetrahedra defined in (4.2) are visibility triangles. If the segments and data points are in "general position", no segment will intersect any graph edge and no data point will lie in the triangles defined just above. Hence the chain is trivially valid whenever no set of three segments and no set of three data points are coplanar.

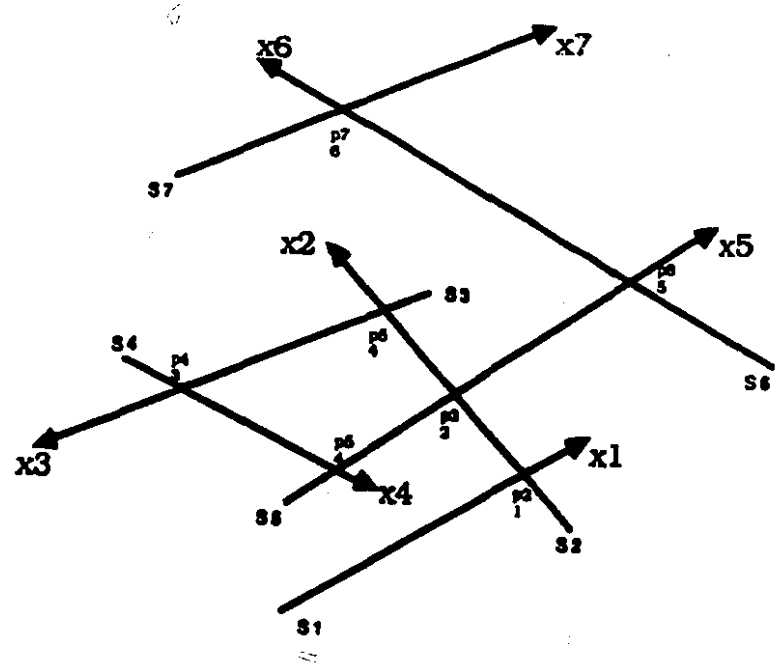

Figure 4.3 Adjacent semi-infinite segments intersect in three-dimensional space. When the data points and the segments are in general position, the chain $x_1, x_2, \ldots, x_6$ forms a degenerate valid graph.

We note however that we imposed the triangulation condition on our definition of validity because we wanted to model a simple two-dimensional surface. If on the other hand we wish to model a one-dimensional contour, then the chain should be the valid graph-theoretic structure. The chain is open if homeomorphic to a segment and is closed if homeomorphic to a circle.

Hence, we reformulate the definitions and theorems of the previous paragraph in a setting for which the desired graph is a chain. This setting brings us back to the two-dimensional case (Alevizos, Boissonnat, & Yvinec, 1987). If we assume that the set V of data points lies in a plane E^2, and that we are to construct a **contour connectivity** graph G for which $V(G) = V$, then the following definitions and theorems apply:

Definition 5 *The plane graph G whose edge set is the set of data points will be called* **2-valid** *if and only if it is:*

- *A chain.*
- **2-consistent**. *We define G to be 2-consistent if and only if:*
  - *(a) No acquisition segment intersects any edge of G, except at its vertices.*
  - *(b) Each edge of G can be linked to the acquisition center of the frame, such that the closure of the resulting triangle contains no data point, except at its vertices.*

In a manner similar to that of the previous paragraph, we define S^1 to be a circle centered at P^α, for a given frame $\mathcal{F}^\alpha$ embedded in E^2. $\forall x_i \in E^2$, let ψ_i be the polar angle of x_i around P^α with respect to an orthonormal basis $\mathcal{B}$. Let

$$\mathcal{R} : E^2 \longrightarrow S^1,$$

$$x \mapsto \psi_x.$$

Let $\mathcal{R}^\alpha$ be the restriction of $\mathcal{R}$ to V^α. Let $V^{\alpha'''} = \{x'''|\forall x \in V^\alpha, x''' = \mathcal{R}^\alpha(x)\}$. Let $G^{\alpha'''}$ be the graph such that $V(G^{\alpha'''}) = V^{\alpha'''}$ and

$$\forall x \forall y \in V^\alpha, (\overline{x,y} \in E(G^\alpha)) \Leftrightarrow \left(\overline{\mathcal{R}^\alpha(x), \mathcal{R}^\alpha(y)} \in E\left(G^{\alpha'''}\right)\right). \quad (4.7)$$

Theorem 3 *Let G^α be any graph whose vertex set is V^α for a fixed frame F^α embedded in E^2. Let $G^{\alpha'''}$ be the graph on the image set of G^α by the polar angle transformation $\mathcal{R}^\alpha$ such that (4.7) holds. If $G^{\alpha'''}$ is a chain, then G^α is 2-valid.*

Proof: See Appendix A.7.3.

Theorem 4 *Suppose that O follows a planar, closed, convex curve C such that each acquisition segment is tangent to C. Then if i is an index increasing with the parameter of C, the graph obtained by joining each $x_i (i \in \{1, \ldots, N-1\}) \in V$ to x_{i+1} is 2-valid.*

Proof: See Appendix A.7.4.

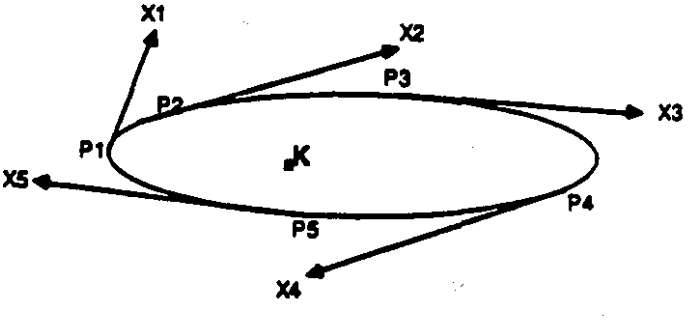

Figure 4.4 O follows the convex,closed curve C. The graph obtained by joining adjacent elements of V is 2-valid.

4.1.2.2 Summary

In summary, we showed in this section that the definition of a frame and the conditions for validity can be somewhat relaxed in some special cases. In particular,

- The actual sensor locations are not important as long as (4.4) holds.
- Suppose the sensor follows a three-dimensional path such that (4.5) holds, that it acquires data tangentially to the path, and that the data points and segments are in general position. Then the graph drawn by joining adjacent data points is degenerate but valid.
- Suppose the sensor follows a two-dimensional, closed, convex curve and acquires data tangentially to its path. Then the graph drawn by joining adjacent data points is 2-valid.

These observations link the process of data acquisition with that of guaranteeing the validity of connectivity inferences. Hence one may wish to put them to practical use by incorporating such knowledge right at the early data acquisition stage.

4.2 Merging the Frames

We showed in Section 4.1.2 how we could construct a valid visibility graph for an isolated frame. The next challenge is to combine the valid visibility graphs obtained from the different views into a valid surface connectivity graph for the object $\mathcal{M}$.

4.2.1 Preliminaries

4.2.1.1 The Tetrahedral Bundles

Notation:

- $\forall \alpha \in \{1, \ldots, \nu\}$, $\forall f = (x_1, x_2, x_3) \in F^\alpha$, we define t_f to be the visibility tetrahedron $\overbrace{x_1, x_2, x_3, P^\alpha}$. We say that f is the cycle (or face) **associated** with t_f.

Definition 6 *The planar polygons*

$$U_i^\alpha = (P^\alpha, x_{i,1}^\alpha, x_{i,2}^\alpha, \ldots, x_{i,m}^\alpha)$$

$$\left(\text{resp. } V_j^\alpha = \left(P^\alpha, x_{1,j}^\alpha, x_{2,j}^\alpha, \ldots, x_{n,j}^\alpha\right)\right),$$

are called the **sheets** *of frame $\mathcal{F}^\alpha$ (See Figure 4.8).*

The data subscripts in the above definition indicate their positions of acquisition within $\mathcal{F}^\alpha$, with n angular increments along direction u, and m angular increments along direction v.

By construction (see the discussion following Theorem 2), the vertices of the sheets, other than P^α, form a connected chain which is a subset of G^α. Further, this chain is monotone in the direction of constant u (resp. v). Such a monotone planar chain can always be trivially triangulated with respect to P^α. We shall make use of this fact later.

Definition 7 *We call $U_1^\alpha, U_n^\alpha, V_1^\alpha, V_m^\alpha$ the* **bounding sheets** *of frame $\mathcal{F}^\alpha$.*

Definition 8 *The* **tetrahedral bundle** $\mathcal{T}^\alpha$ *of frame* $\mathcal{F}^\alpha$ *is defined as the set of all visibility tetrahedra* t_f, *with* $f \in F^\alpha$.

We also loosely refer to $\mathcal{T}^\alpha$ as the polyhedron whose set of faces is the union of F^α and of the bounding sheets of $\mathcal{F}^\alpha$. The precise meaning is clear from the context.

We also introduce the following notation:

- $u^\alpha_{i_j}$ is the triangle $\widehat{P^\alpha, x^\alpha_{i,j}, x^\alpha_{i,j+1}}$.
- $v^\alpha_{j_i}$ is the triangle $\widehat{P^\alpha, x^\alpha_{i,j}, x^\alpha_{i+1,j}}$.
- Δ^α is the set of triangles of the bounding sheets of $\mathcal{F}^\alpha$.

$$\Delta^\alpha = \bigcup_{j=1}^{m} u_{1,j} \cup \bigcup_{j=1}^{m} u_{n,j} \cup \bigcup_{i=1}^{n} v_{1,i} \cup \bigcup_{i=1}^{n} v_{n,i}.$$

4.2.1.2 Orienting the Graph Faces

The object being modeled, $\mathcal{M}$, is embedded in the three-dimensional space E^3, or in a portion thereof. Since $\mathcal{M}$ is orientable, it is the boundary within E^3 of two open sets, one of which we call the **interior**, and the other the **exterior**. One of these two sets is the free space, in which the sensor evolves, while the other is solid space. The free space may be the interior of $\mathcal{M}$, such as when the boundary is the set of connected walls of a room. Alternately, it can be the exterior of $\mathcal{M}$, such as when the boundary is that of a solid object isolated from the background on which it rests. For now, we define the notions of exterior and interior for the faces of the visibility and connectivity graphs by assuming the center of acquisition P lies in $\mathcal{M}$'s exterior:

Definition 9 *Let $\mathcal{F}^\alpha$ be an isolated frame. $\forall f \in F^\alpha$, let Π_f be the plane in which f lies. Then we define the exterior of f as the semi-space of Π_f in which P^α lies. The interior of f is the other semi-space. We also define the* **normal** *vector $\vec{n}(f)$ of f as the vector perpendicular to f, and which points towards the exterior of f.*

Contrarily to the faces of the visibility graphs, the faces of the connectivity graph have vertices which do not all belong to the same frame. The acquisition point P^α is not defined in that case. So we need a different procedure for determining the orientation of those faces.

In the next section, we describe how the frames are merged together and how new faces are built. We delay until then the description of how to determine the exterior and the interior of all faces of the connectivity graph G.

Definition 10 *We say that a segment $s = \overrightarrow{P^\beta x}$* **back-crosses** *a face $f \in G = G^{1,\ldots,(\beta-1)}$ if and only if s intersects f and P lies to the interior of f (see Figure 4.5). Conversely, we say that s* **front-crosses** *f if and only if s intersects f and P lies to the exterior of f.*

Definition 11 *Let $s = \overrightarrow{P^\beta x}$. Let F be a set of oriented faces. Then we define $f^F(s)$ be the first face of F, starting from P^β, that s intersects, if such a face exists. Formally,*

$$f^F(s) = \left\{ f_i \middle| \left(\forall f_j \in F, \text{s.t. } s \cap f_j = \{ y_j = \lambda_j s \}, \lambda_i = \min_j \lambda_j \right) \right\}.$$

Definition 12 *We say that a segment $s = \overrightarrow{P^\beta x}$ back-crosses the graph G if and only if $f^{F(G)}(s)$ exists and s back-crosses it. Conversely, we say that s front-crosses the graph G if and only if $f^{F(G)}(s)$ exists and s front-crosses it.*

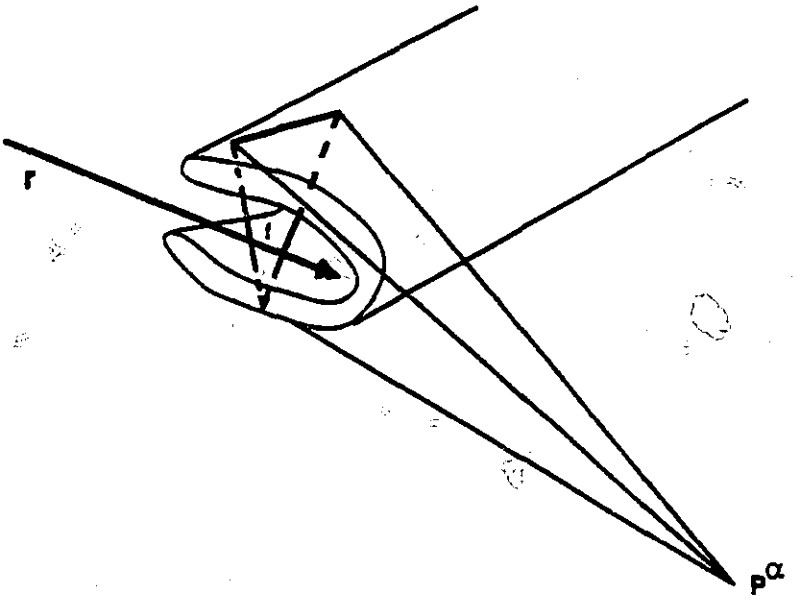

Figure 4.5 The segment r back-crosses the face $f \in F^{\alpha}$.

4.2.2 Overview of Merging Algorithm

Because segments do not in general intersect in three-dimensional space, neither Alevizos *et al.*'s method nor the particular examples of section 4.1.2.1 generalise to the solid model case. In the method we describe below, we start from the visibility graph for frame $\mathcal{F}^{\alpha=1}$ and we incrementally *insert* into its faces the data points from the other frames $\mathcal{F}^{\beta}(\beta \in \{2,\ldots,\nu\})$.

4.2.2.1 Inserting Points into Faces

Suppose we have constructed a graph $G = G^{1,\ldots,(\beta-1)}$ and that we acquire a new frame $\mathcal{F}^{\beta}$. In this section, we describe how to *insert* the data points of V^{β} into a particular *face* of the current graph G.

Suppose we wish to insert only one point x_n of V^β into f. We do it by simply splitting f into three new graph faces f_1, f_2, and f_3, all with a common vertex at x_n (see Figure 4.6). After insertion of x_n, the set of faces of the graph is modified such that $F(G^{1,\ldots,(\beta-1),x_n}) = F(G) \cup \{f_1, f_2, f_3\} \setminus f$.

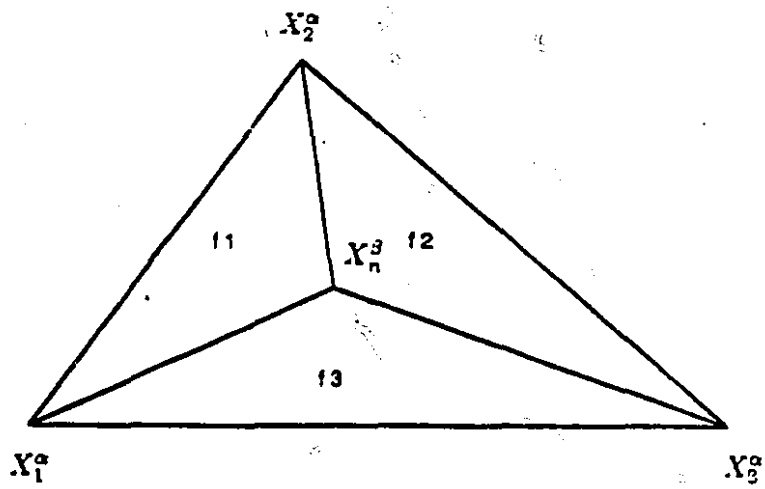

Figure 4.6 Inserting $x_n \in V^\beta$ into $f \in F(G)$.

Suppose now that we wish to insert more than one datum point of V^β into the same given face $f = (x_1, x_2, x_3)$. Let P be that set of points and let $P' = P \cup \{x_1, x_2, x_3\}$. Then P is inserted into G by building a valid (with respect to P^β) triangulation T on P'. Then, $F(G^{1,\ldots,(\beta-1),P}) = F(G) \cup F(T) \setminus f$.

In practice, we may build T as follows. We first build a valid triangulation on P, as described in Section 4.1.2. Since the triangulation has an embedding, its boundary is well-defined. This boundary is the set of projections of the frame's bounding sheets.

The next step is then to build a valid triangulation between the vertices of the boundary of P and the vertices of f. We call the resulting triangulation the **triangulation**

of face f. This process is illustrated in Figure 4.7.

As mentioned in section 4.2.1.2, we need to define the exterior and interior of those faces whose vertices come from different frames. This is done as follows. Suppose f is a face of a visibility graph; its exterior and normal vector are well-defined. Then suppose we are to insert a set of data points into f, so as to create new faces. Then for each such face f', its (exterior pointing) normal vector is defined as the vector perpendicular to f' and which points in the same general direction as f. In other words, $\vec{n}(f)$ is such that $\vec{n}(f) \cdot \vec{n}(f') > 0$.

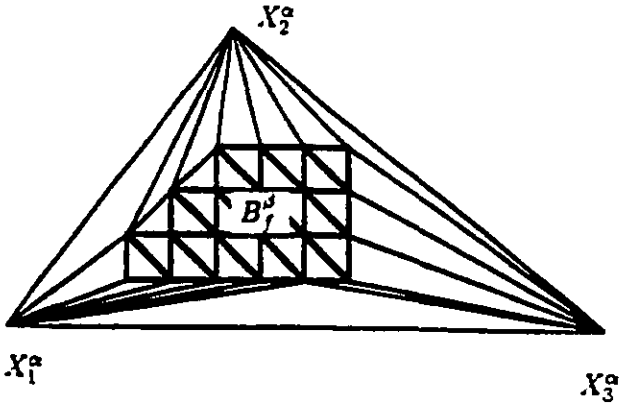

Figure 4.7 A set of points can be easily triangulated within face f.

4.2.2.2 The Insertion Criteria

We now address the question of determining which points to insert into which faces of the graph.

A necessary condition for a given datum point $x \in V^\beta$ to be inserted into a face of the graph $G = G^{1,\ldots,(\beta-1)}$ is that either:

$$s = \overrightarrow{P^{\beta}x} \text{ front-crosses } G, \tag{4.8}$$

or that

$$\exists \alpha \in \{1, \ldots, \beta - 1\}, \exists t \in T^{\alpha}, x \in t. \tag{4.9}$$

In the first case, the first consistency condition is violated and we say that the point x corresponds to a **concavity**. We eliminate the consistency violation by inserting x into the *first* face that s front-crosses.

Otherwise, we look for violations of the second consistency condition. If there exists a visibility tetrahedron t of a previous frame such that x lies inside t, we say that x corresponds to a **convexity**. We remove the consistency violation by inserting x into the face that t is associated with.

Note that if a segment back-crosses the graph, we do not take any action, even though it is a violation of condition (4.1).

4.2.2.3 Determining the Insertion Face

It remains to determine into which face of the graph the update is to be made.

If (4.8) applies, we insert x into $f^{F(G)}(s)$. We note that if $f = f^{F(G)}(s)$, then the last tetrahedron intersected by s prior to intersecting f is t_f.

Similarly, if (4.9) applies, x is inserted into the face associated with the tetrahedron in which x lies. This tetrahedron is also the last one intersected by s.

So in both cases we can find the face $f \in G$ into which $x \in V^{\beta}$ is to be inserted

by finding, for all previous frames $\mathcal{F}^\alpha(\alpha < \beta)$, the last tetrahedron of $\mathcal{T}^\alpha$ that the segment intersects.

Interestingly, f is found in the same way whether the datum point corresponds to a concavity (violation of the first consistency condition) or to a convexity (violation of the second consistency condition). In practice, f is found by performing intersection checks between the segments of S^β and the polygonal sheets formed by the tetrahedra of $\mathcal{T}^1, \ldots, \mathcal{T}^{(\beta-1)}$ (See Figure 4.8).

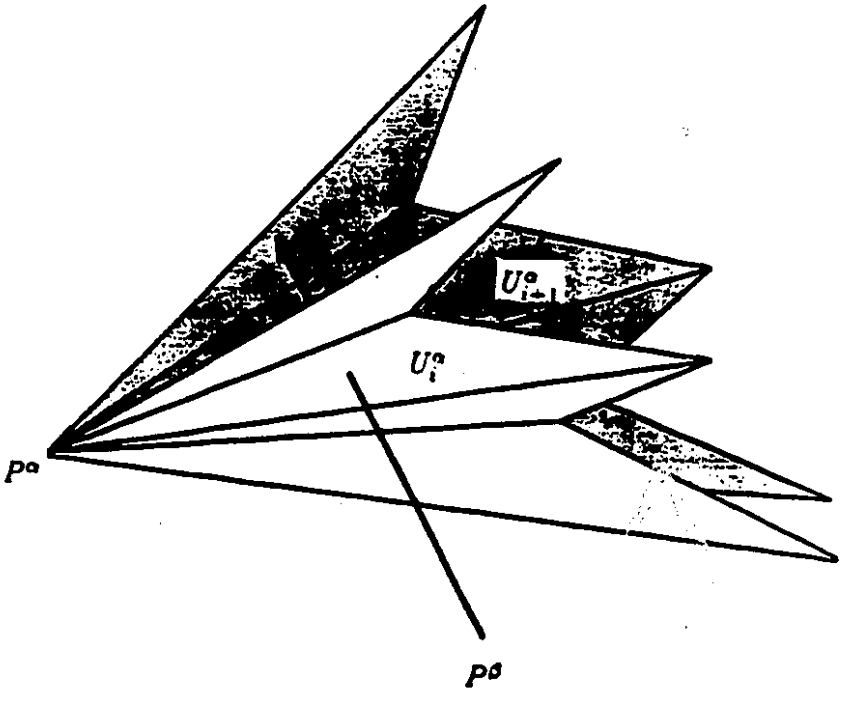

Figure 4.8 Determination of where to insert the new datum point is made by finding the last $t \in \mathcal{T}^\alpha$ that the segment s_i intersects. In the picture, s_i crosses U^α_i but does not cross U^α_{i+1}. A second search in the orthogonal direction would normally be performed before t is actually determined.

Suppose that at a given stage of the algorithm, we determine in the manner explained

above, that $x \in V^\gamma$ is to be inserted into a face $f \in F^\alpha$. If $f \in F(G)$, we simply insert x into f. If however, there exists one or more data points from one more intermediate frames (say $\mathcal{F}^\beta$, with $\gamma > \beta > \alpha$) that have already been inserted into f, then $f \notin F(G)$, as explained in section 4.2.2.1. In this case, we should insert x into a face f' of the triangulation of f. We determine f' in the same manner as we did for determing f, namely by performing intersection checks between the segment terminated at x and the sheets of bundle $\mathcal{T}^\beta$.

Hence we see that each face f of the original graph G^α becomes the root of a hierarchy of triangulations. This hierarchy can be expressed in the form of a tree. Each node of the tree is a face of the partial graph, constructed after aggregation of some of the frames. The children of each node are the faces of the triangulation of their parent face. Only the leaves of the tree are faces of the final graph, while the nodes are 3-cycles of that graph. For each such node f, we call G_f the subgraph rooted at f.

In the next section, we see that concavity points are not inserted at the same time as convexity points. A consequence is that the depth of the tree is at most 2ν: each frame may at most increase the depth of the tree by 2.

In the following, we give a more formal presentation of the algorithm, and we analyse the validity of the resulting graph.

4.2.3 Partitioning a New Frame

Suppose we have already constructed a valid graph $G^{1,\ldots,(\beta-1)}$ and we wish to aggregate the data contained in a new frame F^β into the augmented graph $G^{1,\ldots,\beta}$.

We first construct a partition $\mathcal{P}$ of V^β; this subsection indicates how to perform the partition. Following that, we insert the elements of $\mathcal{P}$ into $G^{1,\ldots,(\beta-1)}$.

4.2.3.1 Partitioning a Frame with Respect to Another Frame

For a given previous frame $\mathcal{F}^\alpha$, we partition V^β into five disjoint sets $A_\alpha^\beta, B_\alpha^\beta, A_\alpha^{\beta'}, B_\alpha^{\beta'}$, and O_α^β. These sets are illustrated in Figure 4.9 and are defined as follows:

A_α^β is the set of points of V^β whose acquisition segments front-cross G^α.

$$A_\alpha^\beta = \{x \in V^\beta, \text{s.t. } \overrightarrow{P^\beta x} \text{ front-crosses } G^\alpha\}.$$

$A_\alpha^{\beta'}$ is the set of points of V^β whose acquisition segments back-cross G^α.

$$A_\alpha^{\beta'} = \{x \in V^\beta, \text{s.t. } \overrightarrow{P^\beta x} \text{ back-crosses } G^\alpha\}.$$

B_α^β is the set of points of $(V^\beta \setminus A_\alpha^\beta \setminus A_\alpha^{\beta'})$ which lie in the tetrahedral bundle $\mathcal{T}^\alpha$.

$$B_\alpha^\beta = \{x \in (V^\beta \setminus A_\alpha^\beta \setminus A_\alpha^{\beta'}) | (\exists t_f \in \mathcal{T}^\alpha, x \in t_f)\}.$$

B' is the set of points of $(V^\beta \setminus A_\alpha^\beta \setminus A_\alpha^{\beta'} \setminus B_\alpha^\beta)$ whose acquisition segments do intersect $\mathcal{T}^\alpha$.

$$B_\alpha^{\beta'} = \{x \in (V^\beta \setminus A_\alpha^\beta \setminus A_\alpha^{\beta'} \setminus B_\alpha^\beta) | (x \cap \mathcal{T}^\alpha = \emptyset) \wedge (\overrightarrow{P^\beta x} \cap \mathcal{T}^\alpha \neq \emptyset\}.$$

O_α^β is the remaining set of points of frame β. They lie outside $\mathcal{T}^\alpha$ and so do their acquisition segments.

$$O_\alpha^\beta = (V^\beta \setminus A_\alpha^\beta \setminus A_\alpha^{\beta'} \setminus B_\alpha^\beta \setminus B_\alpha^{\beta'}).$$

For clarity in the following discussion, we drop the subscript and superscript of the elements of the partition. Some of these elements are further partitioned into disjoint subsets as follows:

A_f is the set of points of A whose acquisition segment front-crosses f of F^α.

$$A = \bigcup_{f \in F^\alpha} A_f \text{ where } A_f = \left\{ x \in A | f^{F^\alpha}(\overrightarrow{P^\beta x}) = f \right\}.$$

B_t is the set of points of B which lie within tetrahedron t of $\mathcal{T}^\alpha$.

$$B = \bigcup_{t \in \mathcal{T}^\alpha} B_t \text{ where } B_t = \{ x \in B | x \in t \}.$$

B'_u is the set of points whose **last** triangle of Δ^α their segment intersects is u.

$$B' = \bigcup_{u \in \Delta^\alpha} B'_u \text{ where } B'_u = \left\{ x \in B' \middle| \left[\left(s = \overrightarrow{P^\beta x} \right) \cap u = \{ y = \lambda s \}, \right. \right.$$
$$\left. \left. \wedge (\forall u_j \in \Delta^\alpha, \text{s.t. } s \cap u_j = \{ y_j = \lambda_j s \}, \lambda = \max j \lambda_j) \right] \right\}.$$

Remark 2 *The maximum number of triangles of Δ^α a given segment $s \in \mathcal{F}^\beta$ intersects is* 2.

Proof: The triangles of Δ^α all belong to the convex hull of $\mathcal{T}^\alpha$ since they are extreme in the directions $\{u, -u, v, -v\}$. Barring singularities, we know that a three-dimensional segment intersects a convex figure in at most two places. So, s intersects Δ^α in at most two faces. □

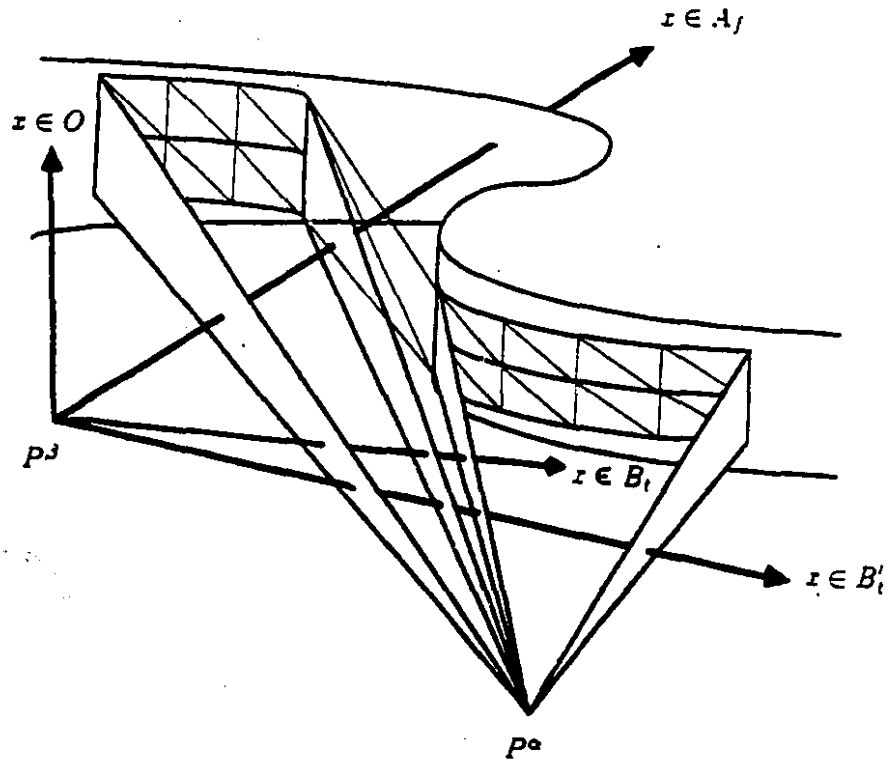

Figure 4.9 Illustration of the graph for frame $\mathcal{F}^\alpha$ as well as of the partitioning of V^β with respect to $\mathcal{F}^\alpha$.

In summary, we have:

$$V^\beta = A^\beta_\alpha \cup A^{\beta'}_\alpha \cup B^\beta_\alpha \cup B^{\beta'}_\alpha \cup O^\beta_\alpha = \bigcup_{f \in F^\alpha} A_f \cup \bigcup_{t \in T^\alpha} B_t \cup \bigcup_{u \in \Delta^\alpha} B'_u \cup A^{\beta'}_\alpha \cup O^\beta_\alpha = \bigcup p^\beta_\alpha,$$

where p^β_α is one of $\{A_f, B_t, B'_u, O, A'\}$.

4.2.3.2 Partitioning with Respect to All Frames

The previous section outlined the partitioning of V^β with respect to a given frame. Here we describe how to obtain a *global* partitioning of V^β with respect to all previous

frames.

We first partition V^β with respect to $\mathcal{F}^{\alpha=1}$. Then we subpartition the resulting subsets with respect to the other frames, in the order of frame acquisition. For a given partition element p_α^β this subpartitioning stops when all its data points are **definite**. A datum point $x \in V^\beta$ is said to be definite when either of the following can be unambiguously determined:

$$\exists \alpha \in \{1, \ldots, (\beta - 1)\}, s = \overrightarrow{P^\beta x} \text{ back-crosses } G^\alpha.$$

$$s \text{ front-crosses } G = G^{1,\ldots,(\beta-1)}.$$

$$\exists \alpha \in \{1, \ldots, (\beta - 1)\}, \exists t_f \in \mathcal{T}^\alpha, x \in t_f \wedge f \in F(G).$$

$$\forall \alpha \in \{1, \ldots, (\beta - 1)\}, (x \in O_\alpha^\beta) \vee (x \in B_\alpha^{\beta\prime}).$$

If (4.2.3.2) holds for x, the partitioning procedure is complete for x. We then say that $x \in A'_{global}$. Suppose now that (4.2.3.2) does not hold for x. Then we repeat the partitioning procedure for all subsequent frames, until a frame $\mathcal{F}^\alpha$ is found, such that either $x \in A_\alpha^\beta$, or $x \in B_\alpha^\beta$. In the first case, (4.2.3.2) holds, and let f be the face of F^α that s front-crosses. In the second case, (4.2.3.2) holds, and let f be the face associated with the tetrahedron of $\mathcal{T}^\alpha$ that x lies in.

If f is a face of the graph G, then we say that $x \in A_{global}^\beta$ if $x \in A_\alpha^\beta$, and that $x \in B_{global}^\beta$ if $x \in B_\alpha^\beta$. As explained in section 4.2.2.2, x is then to be inserted into f.

If f however is not a face of G, we need to find the face of G that x is to be inserted into. This is done by searching the face f' of the subgraph G_f that x should be inserted into. This process in turn determines if $x \in A^{\beta}_{global}$ or if $x \in B^{\beta}_{global}$. If x front-crosses face f', then $x \in A^{\beta}_{global}$. If, on the other hand, x lies in the tetrahedron associated with f', then $x \in B^{\beta}_{global}$.

We finally turn to the case of (4.2.3.2), where x cannot be inserted into any of the faces of G. We then say that $x \in O^{\beta}_{global}$.

4.2.3.3 Summary of Partitioning Procedure

In summary, the partitioning procedure is intimately linked to the process of determining which face of the current graph each new datum point is to be inserted into. This process iterates over the graph induced by the previously inserted frames. The graph is built in the form of a tree, which allows us to search it hierarchically.

The object of the search is to group the data points into subsets. The subsets such that the acquisition segments of their elements back-cross a face of a visibility graph are said to belong to A'_{global}.

Otherwise, the subsets such that the acquisition segments of their elements front-cross a face of the graph are said to belong to A_{global}. They are further subpartitioned with respect to the face of the graph that they front-cross.

Otherwise, the subsets such that their elements lie in a visibility tetrahedron t are said to belong to B_{global}. They are further subpartitioned with respect to the face of the graph associated with t.

The remaining subsets are said to belong to O_{global}. Both their elements and the acquisition segments of their elements lie outside the tetrahedral bundles.

In the following, the subscripts and superscripts are omitted for the elements of the partition. It is then clear from the context that the element names refer to the *global* partitioning.

4.2.4 Inserting into the Previous Frames

We now wish to insert the elements of the partition constructed above into $G^{1,\ldots,(\beta-1)}$, such that after insertion of every element p_α^β, the augmented graph $G^{1,\ldots,(\beta-1),p_\alpha^\beta}$ remain valid. How to do this insertion depends on whether $p_\alpha^\beta \subset A, A', B$, or O. Inserting a new frame is a three-step procedure.

First, the subsets corresponding to convexities are inserted into the graph. These subsets are those of B. The net effect of that insertion is to "enlarge" the modeled object.

Second, the subsets corresponding to concavities are inserted into the augmented graph. These subsets are those of A. The net effect of that insertion is to "carve up" the modeled object.

Third, the connected subsets made of data points outside the 'influence" of any of the current tetrahedral bundles are triangulated and inserted into the augmented graph. These subsets are those of O. The net effect of that insertion is to leave the graph disconnected. The number of connected components is equal to the number of connected components left after the insertion of the last frame, plus the number of

connected components of O^β.

Finally, the elements of A' are not inserted into the graph, as no valid graph that includes the back-crossing data points in its edge set can be constructed (See Figure 4.5). This corresponds to the case where not enough information is available, as noted in Alevizos *et al.* for the two-dimensional case.

We give a precise algorithm that performs the above tasks in Appendix A.8. We call that algorithm `merge-frames`. We now turn to the analysis of the validity of the resulting graph.

4.2.5 A New Definition of Validity

The previous definition of graph validity, namely 2 1/2-validity, turns out to be too restrictive for the multi-frame three-dimensional graph connectivity problem.

We have already seen that back-crossing of a face by a segment should not be considered a violation of consistency condition (4.1), and we ironed that difficulty out by not inserting into the graph the datum point associated with the back-crossing segment.

Another important difference is illustrated in Figure 4.10. There, a segment and its datum point violate both consistency conditions, although with respect to two different faces: s crosses f and lies inside $t_{f'}$. If we insert x into f, consistency condition (4.2) is still violated, while if we insert x into f', consistency condition (4.1) is still violated. In this case, we contend from visibility considerations that the datum point should be inserted into the face being crossed by the segment, namely into

f. In other words, maintaining consistency condition (4.1) takes precedence over maintaining consistency condition (4.2) because (4.1) of the strong assumption of object opacity.

At times, a segment may front-cross more than one face of the graph, as is also shown in Figure 4.10. In this case also, we contend from visibility considerations that the datum point should be inserted into the first face that the segment front-crosses.

Note that in the last two examples, some faces of the resulting graph G intersect and hence cannot model a simple polyhedron. Although a self-intersecting model is not geometrically correct, the *connectivity* information that G conveys is nonetheless correct.

Finally, because the elements of O and their associated segments entirely lie outside the tetrahedral bundles, there may not be any preferred portion of the existing graph with whom to associate them. In this case, it is preferrable not to "artificially" insert them into a particular face of the graph. Hence we obtain a disconnected graph whenever $O \neq \emptyset$.

Algorithm `merge-frames` accounts for the above considerations. We now couch them into the following validity definitions for connectivity graphs:

Definition 13 *We say that a 3-cycle $c = (x, y, z)$ of G is a* **bridge** *to face $f \in F(G)$ for vertex $x_1 \in V(G)$, if and only if every path from x_1 to a vertex of f passes through either x, y, or z (See Figure 4.11).*

Definition 14 *Let c be a 3-cycle of G. Then the* **pseudo-face** *of c (denoted f_c) is the face that passes through the vertices of c.*

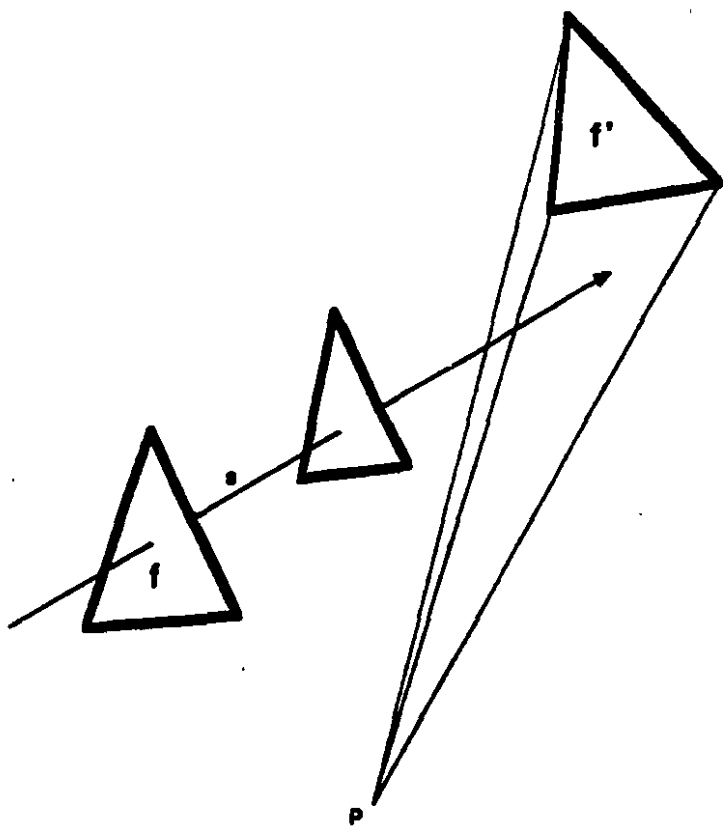

Figure 4.10 A given segment may violate both consistency conditions. It may also front-cross more than one face of the graph.

Definition 15 *A surface connectivity graph G will be called* **3-valid**, *if and only if it is:*

- *A triangulation or a set of triangulations.*
- **3-consistent**. *G is said to be 3-consistent if and only if, for each acquisition segment* $s = \overrightarrow{P^\beta x}$*:*
  - *If s front-crosses a set of faces* $F \subset F(G)$*, then there exists a 3-cycle c in*

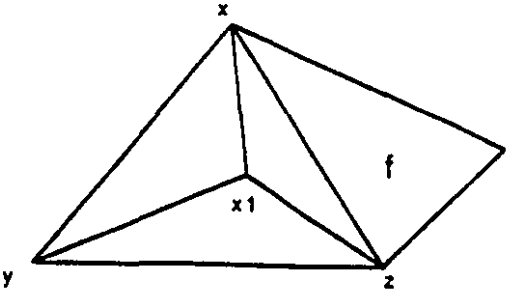

Figure 4.11 Cycle (x, y, z) is a bridge to face f for vertex x_1.

G such that s front-crosses the pseudo-face of c before it does any face in F, and c is a bridge for x to all faces of F. Formally,

$$\forall f \in F(G), \texttt{s.t. } s \in S \textit{ front-crosses } f, \exists \textit{ a 3-cycle } c \in G, \texttt{s.t.}$$

$$(f^{F(G)\cup f_c}(s) = f_c) \wedge (f_c \text{ is a bridge to } f \textit{ for } x). \tag{4.10}$$

– *If s does not front-cross G,then of all the prior visibility tetrahedra that x lies in, at least one has its associated face not in F(G). Formally,*

$$[f^{F(G)}(s) = \emptyset] \Longrightarrow$$

$$\left[\left(S = \left\{t_f | \left(t_f \in \cup_{\alpha=1}^{\beta-1} T^{\alpha}\right) \wedge (x \in t_f)\right\} \neq \emptyset\right) \Longrightarrow \left(\exists t_f \in S, \texttt{s.t. } f \notin F(G)\right)\right]. \tag{4.11}$$

As the following theorem shows, this new definition of validity is less restrictive than the previous ones.

Theorem 5 *If G is a 2 1/2-valid visibility graph, it is 3-valid.*

Proof: Follows from the definitions. The triangulation condition is trivially satisfied. Condition (4.10) above follows directly from consistency condition (4.1): Since no segment intersects any face of F^α, any statement on the crossed faces is always true. Finally, since x lies in none of the visibility tetrahedra of $\mathcal{T}^\alpha$, condition (4.11) is trivially satisfied by condition (4.2). □

The major differences between this new definition and the previous validity definitions are that:

- G may be disconnected. An obvious example in which this is desirable is given by the case when tetrahedral bundles do not intersect. There is then no criterion by which the data points of different frames can be joined.
- Back-crossing of a face by a segment is not considered to be a violation of the first consistency condition.
- A segment $\overrightarrow{Px}$ may front-cross a face f of G, but only if x is inserted into a 3-cycle of the graph that s front-crosses before it does f. This seemingly-contrived statement embeds the notion that s may originally front-cross more than one face. In order to remain a triangulation, x can obviously be inserted into only one of those faces. The crucial point is that x be inserted into the first face f that s front-crosses. Once the insertion is made, f becomes a 3-cycle of the graph rather than a face, but the order of face crossings remains.
- For any graph G, at least one of the two consistency conditions is always verified for each segment $s \in S$. Which condition is necessarily verified depends on whether s intersects a face of G. Cases where the two consistency conditions are violated but where only one can be resolved are thus avoided.

- A datum point x may lie inside one or more tetrahedra t_f, but only if $f \notin F(G)$ for at least one of the tetrahedra. The second 3-consistency condition states that the points corresponding to convexities should be inserted into the face associated with one of the tetrahedra in which they lie. Since they may lie in more than one such tetrahedron, but can only be inserted into one face, this condition is deemed sufficient.

4.2.6 Multi-frame Merging and 3-Validity

In section 4.2.2.3, we saw that data points could be incrementally inserted into the graph, giving it a tree-like structure. This structure in turn allows us to efficiently insert new data into the graph. Unfortunately, this efficient procedure makes the algorithm fail to produce a 3-valid graph in some instances.

Figure 4.12 gives such an example. Suppose a datum point $x^3 \in V^3$ is found to lie in $t_f \in \mathcal{T}^1$. Algorithm `merge-frames` will insert x^3 into face f of F^1. In doing so, new faces will be created. Let F be that set of faces. Because the segments of the earlier frames other than $\mathcal{F}^1$ (for example those of $\mathcal{F}^2$) were not checked against the faces of F, there is in general no guarantee that those earlier segments do not in fact front-cross some of the faces in F. Thus, condition (4.10) may be violated in that case and the 3-validity cannot be guaranteed.

A possible cure for that problem would be to check if any of the earlier segments intersects any of the newly-created faces. Suppose such an intersection occurs between a segment $(s = \overrightarrow{P^2x}) \in S^2$ and a face $f \in G^{1,2,3}$. Then, x should be inserted into f in order not to violate condition (4.10). This step implies that x should be first removed from its current position in the graph $G^{1,2}$.

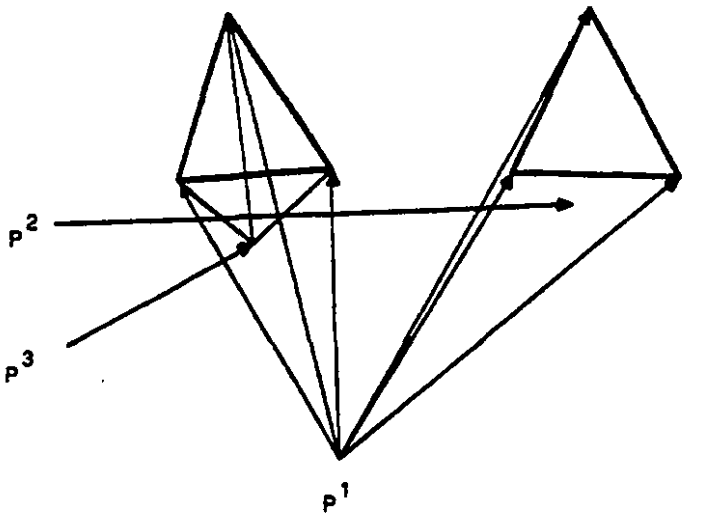

Figure 4.12 The `merge-frames` algorithm may not always produce a 3-consistent graph. Here, *merge-frames* would not detect that the first face crossed by s^2 is not the one in which s_2 is inserted.

As a result, the problem of multi-frame graph construction using a modification of `merge-frames` is at least polynomial in the order of the number of frames ν. Although this is feasible, this additional step greatly degrades the efficiency of the algorithm. Worse still, it necessitates constructing a complex update mechanism which is not compatible with incremental monotonic scene construction.

In the next sections, we outline the cases under which `merge-frames` is guaranteed to yield a 3-valid graph.

4.2.6.1 The Case of Two Frames

Theorem 6 *Algorithm* `merge-frames` *can be slightly modified to yield a 3-valid graph*

when $\nu = 2$.

Proof: See Appendices A.9 and A.10.

4.2.6.2 The Two-dimensional Case and Extensions

The reason for the failure of the algorithm in the example given above is that segments do not in general intersect in three-dimensional space. Even if a segment $s = \overrightarrow{Px}$ entirely **traverses** a given tetrahedron t_f, *i.e.* crosses two of its sheets, it may be that s front-crosses within t_f a *not-yet inserted* face f'. Hence , f' is the first face that s front-crosses, but `merge-frames` inserts x into f.

In two dimensions, the situation is quite different. If a segment s traverses a visibility triangle formed by acquisition segments, opacity guarantees that s does not intersect any graph edge inside that triangle. See figure 4.13.

Theorem 7 *If any set of three acquisition segments intersect into a triangle, there exists no 2-valid graph on the data set if the triangle contains any data point. Formally,*

$$\forall\{s_1, s_2, s_3\} \subset \bigcup_{\alpha=1}^{\nu} S^{\alpha},$$

$$(s_1, s_2, \text{ and } s_3 \text{ intersect in } p_1, p_2, \text{ and } p_3) \Longrightarrow \left(\left(t = \widehat{p_1, p_2, p_3} \cap V\right) = \emptyset\right). \quad (4.12)$$

Proof: Since the data points all lie at the termination of a segment, there exists at least three data points which lie outside t, namely the three data points x_1, x_2, and x_3 which terminate s_1, s_2, and s_3.

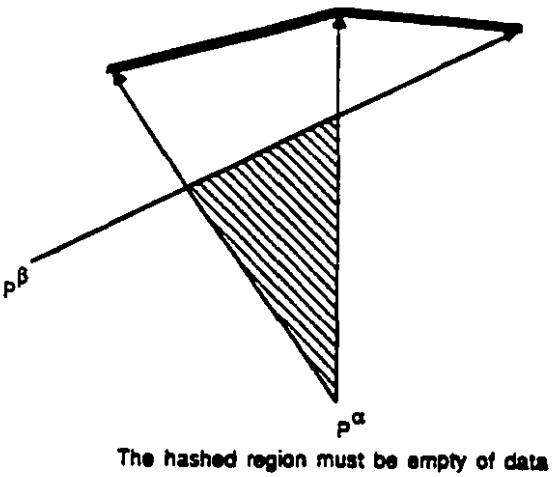

Figure 4.13 Illustration of theorem 7.

Now suppose t contains a datum point x. Then some elements of V (for example x_1) lie outside t while at least one, namely x, lies inside t. Since G is a chain, it is connected. By the Jordan Curve Theorem, any path joining x to x_1 must cross the boundary of t. Hence at least one of the edges of G must cross either s_1, s_2, or s_3. This violates the first 2-consistency condition. So G cannot be 2-consistent if (4.12) does not hold. □

If we now project the data of figure 4.12 onto a plane, we can see that segments $\overrightarrow{P^2x_2}, \overrightarrow{P^1x_a}, \overrightarrow{P^1x_b}$ intersect two-by-two, forming a triangle. By theorem 7, if datum point x_3 is in that triangle, there can be no 2-valid graph on the data. Hence, no two-dimensional object homeomorphic to a circle may have generated such a data set and the pathological situation of figure 4.12 cannot occur in two-dimensions.

A direct consequence of the validity of algorithm **merge-frames** in two-dimensions

is that it also is valid in cases where the object being modeled is a "swept" two-dimensional object or a surface of revolution.

4.2.7 Complexity Analysis

We assume in the following that there is a bounded and usually small number ν of frames. Each frame is composed of a maximum of n points and we are to merge a total of N data points. Hence, $n\nu > N$. We analyse the complexity of constructing a graph on the merged data set.

As mentioned in section 4.2.2.1, the graph is built in the form of a tree, and this structure reduces the number of computations needed to insert new points into the graph. However, no guarantee can be given in the general case that the data points will naturally equi-partition themselves among the tree branches. The following is then a worst-case analysis, except when otherwise indicated.

The initial triangulation of a given frame or of any connected subset can obviously be performed in a time that grows linearly with the number of data points involved: thanks to the matrix-like indexing of the data, no check need be performed. The analysis of complexity then centers on the amount of work needed to insert a new point $x_i \in V^\beta$ into $G^{1,\ldots,(\beta-1)}$.

4.2.7.1 Complexity of algorithm `merge-frames`

Since the graph G is planar, the number of its faces never exceeds $O(N)$ throughout its construction. For each datum point, the main loop of the algorithm checks whether

its segment intersects each face of G. Hence, an obvious worst-case complexity result is that the algorithm runs in $O(N^2)$ time.

An interesting question is whether the expected behavior of the algorithm can be made lower than the quadratic bound.

We first note that for a given new datum point x and its segment $s = \overrightarrow{Px}$, not all faces of the graph need in general be checked. Because the segment travels in a straight-line path, and because the frames form a matrix structure, the number of visibility tetrahedra of a given frame that s traverses never exceeds the larger linear size of each frame. Hence we can precompute the set of faces that the segment **may** cross.

Let now suppose that the frames are approximately square. Since no frame contains more than $O(N)$ data points, the linear size of each frame is at worst $O(\sqrt{N})$. In the worst-case, namely when s is checked with respect to each tetrahedral bundle, the complexity does not exceed

$$O\left(N \sum_{\alpha=1}^{\alpha=\nu} \sqrt{n}\right) = O(N\nu\sqrt{n}).$$

ν is usually bounded downwards by 2 (the cones of support of the convex hull of the object) but depends in practice on how we choose to position the sensor.

Table 4.1 illustrates how this complexity result varies as a function of ν. As one would expect, the greater the number of frames, the fewer the points in each frame, and the smaller the efficiency gain due to the frame data organisation.

$O(\nu)$	$O(n)$	Complexity
1	N	$N^{\frac{1}{2}}$
$\sqrt{(N)}$	$\sqrt{(N)}$	$N^{\frac{1}{4}}$
N	1	N^2

Table 4.1 Worst-case complexity of graph construction process when the frames are approximately square.

4.2.7.2 Complexity of the 3-valid algorithm

As noted in Section 4.2.6, the algorithm `merge-frames` can be modified to guarantee the 3-validity of the graph if, every time we insert a point corresponding to a convexity, we verify that the newly-created faces do not intersect any earlier segment. Since the number of newly-created faces is bounded by 3 for each new datum point (the graph being planar), the complexity of this additional test then is proportional to the number of earlier segments. Hence this algorithm is less efficient than algorithm `merge-frames` by a factor of $O(N)$. The overall worst-case complexity is then $O(N^3)$, and the expected complexity is $O(N^2\nu\sqrt{n})$, if one assumes that the frames are approximately square.

4.2.7.3 Complexity in the Case of a Surface of Revolution

As mentioned above, the algorithm is guaranteed to be valid for an arbitrary number of frames in the case of a surface of revolution. Here, we analyse the complexity in that case.

We first must partition the set V^β into its subsets. For each $x \in V^\beta$, we must test if $s = \overrightarrow{P^\beta x}$ intersects a face of G, and if so , which such face, or if x belongs to a tetrahedron of $\mathcal{T}^\alpha$, and if so, to which tetrahedron.

These tests are done by testing for intersections between s and the sheets of the frames. For a given frame G^α, the problem is equivalent to the point-location problem in a planar graph subdivision. It can be resolved in $O(log^2 n^\alpha)$, while thanks to the particular structure of G^α, no initial preprocessing is needed (Preparata & Shamos, 1985, p. 23).

The $U^\alpha(V^\alpha)$ sheets define a complete ordering around P^α. So we can logarithmically determine which sheets are crossed by s and which are not, until we find a pair of adjacent sheets, only one of which is crossed by s (see figure 4.8). Alternately, if s either crosses no sheet or crosses two bounding sheets, then x belongs to B' or to O respectively, and no further search is needed.

We make use of the following feature: if a segment s traverses a visibility tetrahedron t_f, we are guaranteed that there exists no face of the graph inside t_f that s may first front-cross. Since the object is a three-dimensional extension of a two-dimensional shape, by theorem 7 t_f cannot at the same time contain data any data point and be entirely traversed by segments.

The search is first performed in, say, the u direction. This allows us to locate x between two polygonal sheets U_i^α and U_{i+1}^α. Since the sheets are sorted, this requires $O(logn^\alpha) * Q$ where Q is the complexity of determining, for a given sheet, on which of its two sides x lies.

Because the number of vertices of each sheet may be at least a constant fraction of n^α, the number of such vertices is $O(n^\alpha)$. However, since the sheet has a trivial sorted triangulation made up of the $v_{j,i}^\alpha$ triangles, we can also perform this test logarithmically. Hence $Q = logn^\alpha$, and the complexity of partitioning a given point with respect to a given frame is $O(log^2 n^\alpha)$.

$O(\nu)$	$O(n)$	Complexity
1	N	$N\log^2(N)$
$\sqrt{(N)}$	$\sqrt{(N)}$	$N^{\frac{3}{2}}\log^2(N)$
N	1	N^2

Table 4.2 Worst-case complexity of graph construction process in the case of a surface of revolution.

We noted above that we may need to perform the partition with respect to each previously inserted frame. Hence partitioning a given point with respect to all previous frames takes $O(\nu * log^2 n)$ operations.

Finally, we need to estimate the work required to actually insert the point into the graph. Since these operations are strictly local however, they can be performed in constant time.

In summary, since we have a total of N points and ν frames, the worst-time complexity of the graph construction process in the case of a surface of revolution is

$$N * (2 * \nu * (logn * logn)) = \nu * N * (log^2 n).$$

Table 4.2 illustrates how the complexity result varies as a function of ν in this case.

4.3 Chapter Summary

In Chapter 3 we showed that surface connectivity of objects is a crucial consideration for building solid models from discrete sensor data, and that connectivity cannot be based on metric considerations alone. In this chapter, we gave formal definitions of validity for the surface connectivity of objects homeomorphic to spheres. These definitions allowed us to link the issue of object geometry and that of object visibility.

We then described a sub-quadratic on-line data acquisition algorithm that incrementally constructs a surface representation from a set of three-dimensional data points and sensor positions. We analysed the algorithmic's efficiency under different assumptions about the input data.

We analysed the conditions under which the algorithm maintains our formal validity criteria. We were able to guarantee such validity in the most general case when the number of sensor locations did not exceed two. We also obtained the same result when the object was a surface of revolution, or a planar figure swept in space. In all other cases, the validity of the resulting graph could only be guaranteed by relinquishing the algorithm's efficiency. For this reason, we conclude that algorithm `merge-frames` cannot efficiently solve the the three-dimensional multi-frame on-line connectivity reconstruction problem.

Chapter 5

A Global Algorithm

In the last chapter, we saw that organizing data along separate views does not in general guarantee an efficient merging algorithm for a large number of views. The reason is that `merge-frames` must verify whether the segments of the later frames intersect the protuberances created by the elements of B_f belonging to the earlier frames.

In this chapter, we describe another algorithm that assumes no organization for the data points. We do not require any longer that the data be partitioned into frames, namely sets with a common acquisition center. It follows that the units of data with which we deal are the data points themselves, and we gain in generality.

On the other hand, such a paradigm does not support incremental data acquisition and incremental data merging. If it did, every point would constitute its own "frame". But we saw in Chapter 4 that the initial frame requires a minimum of three points in order for a face to be constructed. Even if we were to guarantee the existence of

such a three-point-frame, the frame merging algorithm would not perform efficiently because the number of frames would still be of the same order as the number of points.

Hence, a price to be paid for the lack of data organization into frames is the loss of the on-line, or incremental property (See Section 1.2.3). As a result, all data must be available before the reconstruction algorithm begins, and we thus say that the algorithm is *global.*

5.1 Algorithm Overview

The global algorithm first constructs the convex hull[1] of the set V of all surface data points as an initial approximation G_0 to the desired graph G. Then successive graphs G_i are iteratively constructed by local modifications of G_{i-1}. The number of data points spanned by the successive graphs is guaranteed to grow monotonically. The process stops when the set of vertices spanned by the graph is equal to V.

The modification of G_i is done on a face-by-face basis, where each face f of G_i is locally modified into a new subgraph G_i^f by the adjunction of new vertices not yet spanned by G_i. The vertex set of G_i^f is the union of the vertices of f and of the set of vertices whose acquisition segment traverses f.

We can see that the algorithm borrows ideas from both (O'Rourke, 1981) and (Alevizos et al., 1987). While both of the above use the convex hull as an initial shape, O'Rourke carves the convex hull by locally modifying the faces of the graph with

[1]The convex hull of a set of points is the smallest convex set including these points. Equivalently, it is the intersection of all convex sets containing these points.

points that are "near" these faces. As we noted in Chapter 3, nearness is in this case defined using Euclidean distance criteria only. Alevizos *et al.*, on the other hand, use the information provided by the acquisition segments to obtain a total angular sort on the two-dimensional data points. The total sort represents the order of the vertices along the boundary of the desired polygon. The sort is obtained using segment-to-segment crossing arguments and is successful because lines in a plane intersect in general. Since lines do not in general intersect in three-dimensional space, their algorithm does not generalize in three dimensions. Note that the situation is similar to the one we encountered in Section 4.2.6.2, as their algorithm relies on the fact that the triangle formed by triplets of intersecting segments is provably empty of data points.

5.2 Algorithm Description

5.2.1 Notation:

We use the following notational conventions. Also refer to the notation introduced in Section 4.1.

- G is the resulting solution graph.
- V is the set of input data points (or vertices).
- S is the set of directed acquisition line segments associated to the elements of V. If $\hat{V} \subset V$, then $S(\hat{V})$ is the set of segments associated with the elements of $\hat{V}$.

- F is the set of faces of G.
- If $v \in V$, $s(v)$ is the element of S associated with v, and $p(v)$ is the other endpoint of $s(v)$, namely the point of acquisition of the datum.
- Graph subscripts indicate iteration levels. For example, G_i is the graph at iteration i and S_i is the set of acquisition segments of the vertices spanned by G_i.
- Graph superscripts refer to the face of the preceding iteration's graph that a given subgraph is *rooted at*. For example, G_i^f is the subgraph of G_i rooted at face f, where f is a face belonging to F_{i-1}.
- We elide graph attributes as follows: $V(G_i) = V_i$, $E(G_i) = E_i$ and $F(G_i) = F_i$.
- $\overline{V_i} = V \setminus V_i$.
- $\mathcal{M}_i^f$ is the polyhedron drawn by the graph G_i^f, and $\mathcal{M}$ is the polyhedron drawn by the final graph G.

5.2.2 Preliminary Assumptions

By definition, the elements of V_0 lie on the hull of V, whereas the elements of $\overline{V_0}$ lie in its interior. As mentioned above, we first construct the convex hull of V, thus obtaining the graph G_0, and we then determine which face of F_0 each element of $S(\overline{V_0})$ intersects.

Proposition 1 *Each element of V_0 is associated with a segment that strictly intersects no face in F_0. Each element of $\overline{V_0}$ is associated with a segment that strictly intersects exactly one face in F_0.*

Discussion:

For Proposition 1 to be true, we must make a few additional assumptions. In the following, intersections refer to strict intersections.

Because G_0 draws the graph of a convex polyhedron $\mathcal{M}_0$, any general segment has at most two strict intersections with $\mathcal{M}_0$, while a segment terminated in the closure of $\mathcal{M}_0$ has at most one intersection with $\mathcal{M}_0$.

We first discuss the first part of the proposition. The elements of V_0 belong to the boundary of $\mathcal{M}_0$, therefore their associated segment has at most one intersection with that boundary.

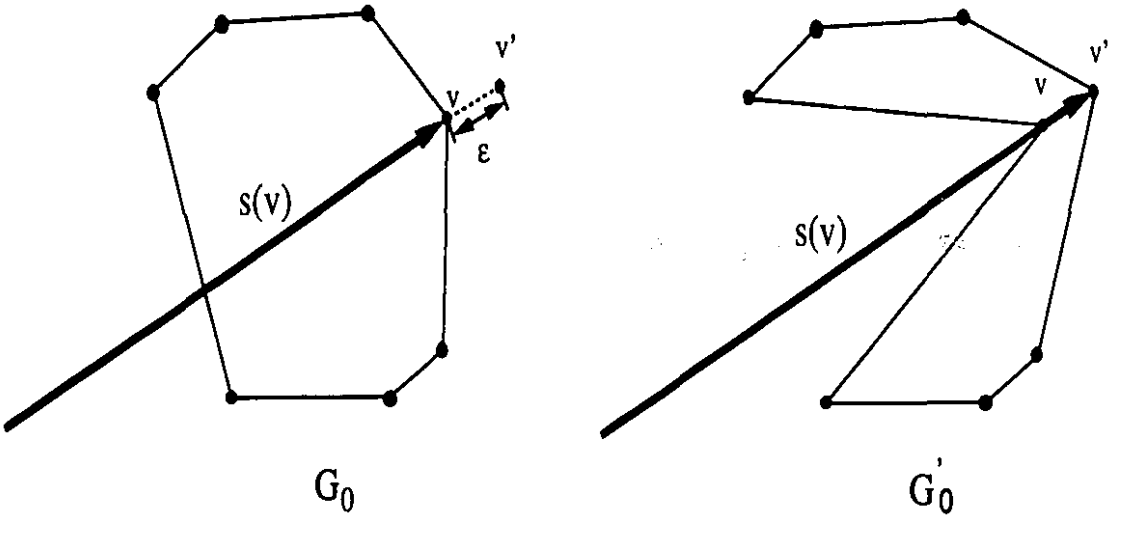

Figure 5.1 Illustration of the first part of Proposition 1: Creation of a phantom datum when not enough data is available. v' is added to the data in order to guarantee that the data points which belongs to the hull do not have an associated segment which intersects the hull. Note the ensuing recomputation of the initial shape.

Suppose there exists $v \in V_0$ with exactly one such intersection (See the two-dimensional analogue in Figure 5.1). In this case, $p(v)$ lies outside $\mathcal{M}_0$, hence it lies outside $\mathcal{M}$ ($\mathcal{M}$ lies inside its convex hull). Because of the opacity assumption, $s(v)$ must not intersect any face of $\mathcal{M}$. Hence $s(v)$ entirely lies outside $\mathcal{M}$, while $v - \epsilon\overrightarrow{s(v)}$ lies inside $\mathcal{M}_0$, for ϵ infinitesimally small, since $s(v)$ intersects $\mathcal{M}_0$ exactly once. On the other hand, $v' = v + \epsilon\overrightarrow{s(v)}$ lies outside of $\mathcal{M}_0$, since v is a boundary point and $\mathcal{M}_0$ is convex. Hence v is an isolated point and $\mathcal{M}$ has no thickness at v. Hence $\mathcal{M}$ cannot model a *simple* polyhedron.

Yet the figure shows this situation can occur when insufficient data is available, as when the solid angle spanned by the acquisition procedure is less than 2π steradians. The situation is easily detectable from its very definition: If a point v belonging to the convex hull $\mathcal{M}_0$ is acquired with a segment which intersects $\mathcal{M}_0$, we add a phantom data point at $v' = v + \epsilon\overrightarrow{s(v)}$. The existence of v' guarantees the first part of Proposition 1 is true.

We next deal with the second part of the proposition. The elements of $\overline{V_0}$ lie in the interior of $\mathcal{M}_0$. Therefore, they have at most one intersection with the boundary of $\mathcal{M}_0$. Suppose there does exist a data point $\overline{v}$ whose segment has no intersection with that boundary. Then $s(\overline{v})$ entirely lies inside $\mathcal{M}_0$. This means that $\overline{v}$ was obtained while the sensor was inside $\mathcal{M}_0$. Because $s(\overline{v})$ does not intersect the hull, the global algorithm is unable to initialize the construction of G_1 for that data point (remember that G_{i+1} is constructed by first verifying which face of G_i the elements of $\overline{V_i}$ intersect).

In Section 1.4 we introduced the concept of a visibility region w associated with each data point v. For simplicity, we later restricted w to be the segment of acquisition $s(v)$. Suppose instead that $w(v)$ is the union of $s(v)$ and of the path of travel of the

sensing apparatus up to the taking of data v (See Figure 5.2). We distinguish two cases depending on whether the sensor is known to have started its path from inside or from outside of the convex object $\mathcal{M}_0$.

If the sensor starts its path outside $\mathcal{M}_0$, then it must have intersected one of the faces of G_0 if there exists $\overline{v} \in \overline{V}$ such that $s(\overline{v})$ lies entirely inside $\mathcal{M}_0$. So $w(\overline{s})$ intersects a face $f \in F_0$. So the algorithm proceeds as if $s(\overline{v})$ had intersected f. In the event where the sensor intersects several faces of F_0, for example if the sensor moves in and out of $\mathcal{M}_0$, then f is chosen to be either the last such intersected face before acquisition, or the first intersected face after acquisition of that data. We assume that the acquisition procedure makes this information available.

Finally, suppose the sensor starts its path inside $\mathcal{M}_0$. In this case, we are unable to initiate the convex hull "carving" procedure. The global algorithm fails as it is unable to make use of the additional information provided by w if the sensor does not eventually leave the convex hull. However, if the sensor eventually leaves $\mathcal{M}_0$ by crossing face f, all data points acquired before then are assigned as if their segment had crossed f, and we then fall back into the previously-described situation for the remaining data points.

In summary, if all data is acquired while the sensor remains inside the convex hull of $\mathcal{M}_0$, the global algorithm cannot make use of the information provided by the acquisition segments since no face of the hull gets ever crossed, neither by the segments nor by the sensor itself. The algorithm then stops and fails. In general, such a situation does not occur when one acquires an object by taking several views of it. In such a case, it is clear that the sensor navigates outside of the hull $\mathcal{M}_0$. But the situation can nonetheless occur when acquiring data along the boundaries of say, the inside of a room (See Section 4.2.1.2). It may then be that the sensor never leaves

the inside of the the convex hull of all the acquired points. Such situations must thus be avoided by careful design of the acquisition sequence.

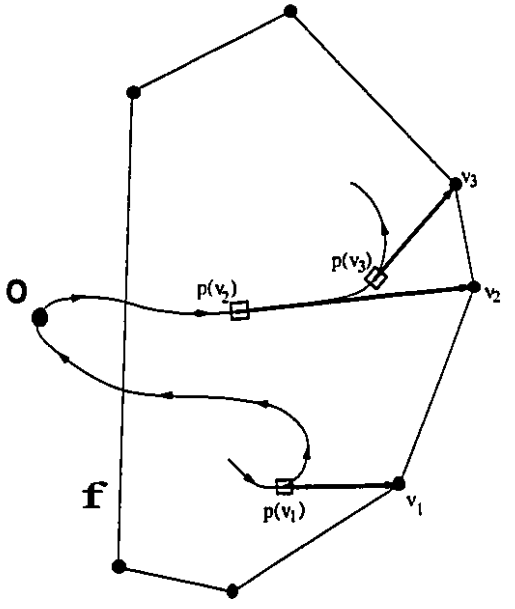

Figure 5.2 Illustration of the second part of Proposition 1: Generalization of the visibility region w for a two-dimensional analogue. The arrows on the trace of the sensor path indicate its sense. The small squares on the trace indicate the sensor position at the time of acquisition. O is an arbitrary point on the trace of the sensor path that lies outside the hull, and between hull boundary crossings. In this example, the trace intersects face f several times. One possible definition of the w_i regions for v_1, v_2 and v_3 is:
$w_1 = \text{Trace}(p(v_1), v_1) \bigcup \text{Trace}(p(v_1), O)$.
$w_2 = \text{Trace}(O, p(v_2)) \bigcup \text{Trace}(p(v_2), v_2)$
$w_3 = \text{Trace}(O, p(v_3)) \bigcup \text{Trace}(p(v_3), v_3)$

5.2.3 Graph Construction Iteration

We have described above how to initialize the algorithm. In the following sections we describe a full algorithm iteration i. We assume we have already constructed a graph

G_{i-1}.

We first construct a partition $\mathcal{P}$ of $\overline{V_{i-1}}$, whose cardinality is that of F_{i-1}. Each element $\overline{v}$ of a given equivalence class p^f of $\mathcal{P}$ is such that f is the first face intersected by $s(\overline{v})$.

Let $f = (v_0, v_1, v_2)$. If p^f is empty, no element of $\overline{V_{i-1}}$ crosses f. In this case, we define G_i^f as the graph drawn by f (namely the complete graph on $\{v_0, v_1, v_2\}$). If p^f is non-empty, we define G_i^f as a maximal three-connected planar graph whose vertex set is $V_i^f = q^f \bigcup \{v_0, v_1, v_2\}$, where q^f is a non-empty subset of p^f, and whose construction we explain in the next section.

Finally, we define G_i as the union[2] of G_{i-1} and of all the G_i^f subgraphs, thus completing the ith iteration:

$$G_i = G_{i-1} \bigcup \overset{f \in F_{i-1}}{\bigcup} G_i^f. \tag{5.1}$$

If G_i now spans the set V, (*i.e.* if $\overline{V_i} = \emptyset$), the algorithm stops. Figure 5.3 illustrates the process for one subgraph.

Claim 1 *At each iteration i, G_i is a three-connected maximal planar graph (3CMPG).*

Proof: See Appendix B.1.

[2]The union of two graphs has for vertex set the union of the graphs' vertex sets and for edge set the union of the graphs' edge sets(Harary, 1972, p 21).

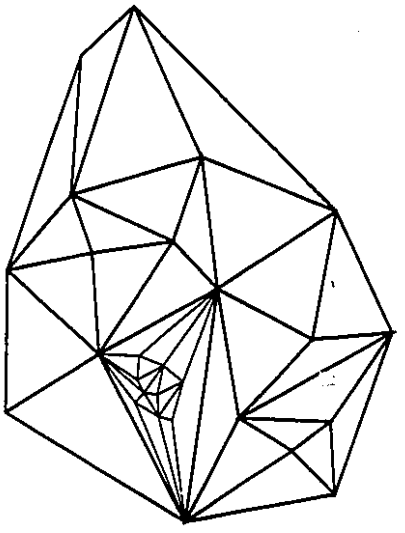

Figure 5.3 A hull G_0 with one subgraph G_1^f. The edges of E_0 are shown in bold face, those of E_1 are shown in lighter face. Non-visible (*i.e.* obscured) edges are not shown.

An obvious corollary of the claim is that the final graph is guaranteed to be a 3CMPG, which is a proper model for the polyhedron $\mathcal{M}$.

Claim 2 *The volume enclosed by the final model is that drawn by the convex hull, less that of the union of the polyhedra drawn by the individual subgraphs of all iterations. Formally,*

$$I(\mathcal{M}) = \mathcal{M}_0 \setminus \bigcup_i^f I(\mathcal{M}_i^f). \tag{5.2}$$

Proof: See Appendix B.2.

Claim 2 states that each successive model monotonically "carves" the volume enclosed by the preceding one.

Claim 3 *The segments associated with the vertices spanned by G_i^f do not intersect any of the faces of G_i^f.*

Proof: See Appendix B.3.

Claim 3 states that each subgraph draws a figure which is *locally* consistent with the data acquisition procedure and the opacity assumption.

Proposition 2 *At any stage i of the graph construction, none of the segments associated with the vertices spanned by the current graph G_i intersects any face of F_i.*

Proposition 2 extends Claim 3 to *all* graph faces and *all* acquisition segments. Since the graph eventually spans all the vertices of V, if Proposition 2 were true, then no segment would intersect any face of G, and the graph would be fully compatible with the opacity assumption. The next sections address the following questions:

1. How are the subgraphs constructed?
2. Can we ensure that Proposition 2 is satisfied?

5.2.4 Subgraph Construction

Recall that f is a graph face of the previous iteration, and that the segments associated with the elements of p^f intersect f. As in (Boissonnat, 1984), we prefer to link to the vertices of f the points of p^f which are "close" to f, or those whose "penetration" into

f is shallow. In this manner, we first construct subgraphs whose faces are near the original f face, and which are themselves intersected by the segments of the "deeper" data points. Again we prefer to use the acquisition segments rather than Euclidean distance measures in order to guide the carving process and to capture to notion of "closeness". This requirement can be met by a special use of *convex layers* (Preparata, 1985).

Let $V^f(0) = V_i^f = p^f \bigcup \{v_0, v_1, v_2\}$. We construct the zeroth layer by taking the convex hull of $V^f(0)$. The next layer consists of the convex hull of the points internal to that hull, again augmented with the vertices of f. The process is repeated until a hull is found which contains no internal point. The graph of that hull is the sought subgraph G^f. The process is more formally described in the next paragraph and is illustrated in Figure 5.4.

Let $G^f(i)$ be the graph drawn by the ith convex layer and let $V^f(i)$ be its vertex set. Let $p^f(i+1)$ be the subset of $V^f(i)$ with degree zero in $G^f(i)$. The elements of $p^f(i+1)$ are the data points which are internal to the ith convex layer. If $p^f(i+1)$ is empty, then $G^f = G^f(i)$. If $p^f(i+1)$ is non-empty, we then let $V^f(i+1) = p^f(i+1) \bigcup \{v_0, v_1, v_2\}$, and the $(i+1)$th convex layer is the convex hull of $V^f(i+1)$, whose graph is $G^f(i+1)$.

Claim 4 *The convex layer algorithm always terminates.*

Proof: See Appendix B.4.

5.2.5 Algorithm's Efficiency

We now analyze the theoretical worst-case asymptotic complexity of the algorithm.

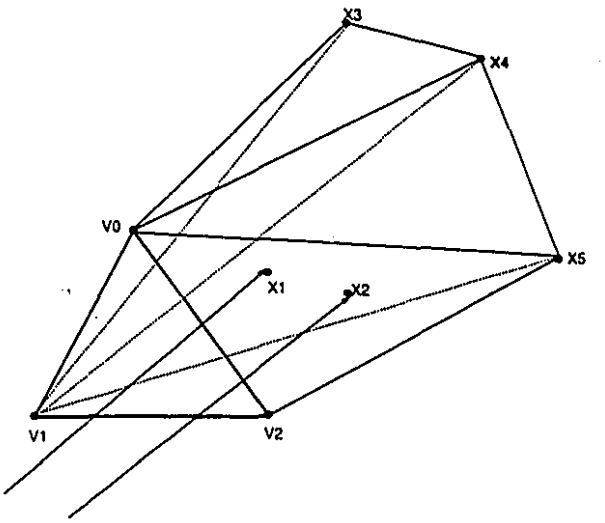

Figure 5.4 A convex layer at a given level i. x_1 and x_2 have degree zero in $G^f(i)$. Hence, we write $p^f(0) = p^f = \{x_1, x_2, x_3, x_4, x_5\}$, $V^f(0) = \{x_1, x_2, x_3, x_4, x_5, v_0, v_1, v_2\}$, $p^f(1) = \{x_1, x_2\}$, $V^f(1) = \{x_1, x_2, v_0, v_1, v_2\}$. For clarity, the only acquisition rays shown here are those of x_1 and x_2.

The initialization phase requires the computation of the three-dimensional convex hull of a set of N spatial points. Such a hull can be optimally computed in $O(N \log N)$ operations (Preparata, 1985). Each graph iteration then requires:

1. The determination of which face f in $\mathcal{M}_i$ each segment of $S(\overline{V_i})$ intersects.
2. For each face of F_i, the determination of the points of p_i^f which achieve maximum *depth*, where the depth of a point is the number of convex layers that have to be stripped from $V^f(0)$ before these points are removed.

In (Dobkin and Kirkpatrick, 1983), a data structure called the *drum representation* is given for polyhedra. It can be constructed in $O(n \log n)$ for an n-vertex polyhedron. Among other uses, the representation can detect the intersection of a polygon with

that polyhedron in $\log n$ time. By extension, it can also detect the intersection of the polyhedron with a segment, since a segment is a degenerate polygon.

So the intersection of all segments in $\overline{V_i}$ with the faces in F_i (Step 1 above), can be performed in

$$O(n_f \log n_f + n_s \log n_f) \tag{5.3}$$

using the drum structure, where n_s is the number of segments and n_f is the size of the polyhedron against which to check intersection. n_s is the cardinality of $\overline{V_i}$, which is bounded above by N. Because the graph of a polyhedron is planar, the cardinality of the face set of $\mathcal{M}_i$ has the same order as that of its vertex set. But V_i is the vertex set of $\mathcal{M}_i$, and its cardinality is also bounded above by N. Hence, $O(n_f) = N$ and Step 1 requires $O(N \log N)$ operations per iteration.

Step 2 requires the determination of the maximal depth convex layer for each V_i^f set. In 2 dimensions, this can be achieved in $O(m^f \log m^f)$ (Chazelle, 1983), which is provably optimal, where m^f is the cardinality of the input set. No such result is known in three dimensions however, so we resort to a repeated application of the simple gift-wrapping technique (Preparata, 1985, pp. 125 and 166). This convex hull construction technique consists in determining an initial facet of the hull, and to "march" around the partial hull by joining each sub-facet of the already-constructed facets with a given data point. This data point is selected from the set of input points using simple trigonometric relationships. Hence each input point is "inspected" a number of times proportional to the number of sub-facets of the resulting hull. In three-dimensions, the sub-facets are the edges of the hull. From the planarity of the graph, the number of edges is of the same order as the number of vertices on the hull. Hence if there are h_0 vertices on the hull, the gift-wrapping technique determines the zeroth layer hull in $O(h_0 m^f)$. Similarly, it determines the $i_t h$ layer in

$$O(h_i \sum_{j=i}^{d} h_j) < O(h_i \sum_{j=0}^{d} h_j) = O(h_i m^f), \tag{5.4}$$

where d is the depth of the set of points. Since all layers must be computed before arriving at the maximal-depth layer, Step 2 is performed in

$$\sum_i O(h_i m^f) = O\left(\left(m^f\right)^2\right), \tag{5.5}$$

for each graph face. Recall that m^f is the cardinality of p^f augmented by 3, and that the sum of the cardinalities of the p^f's is the cardinality of $\overline{V_i}$, which is bounded above by N. Hence Step 2 requires at most $O(N^2)$ operations per iteration and dominates the running time of the algorithm's iterations.

Hence, the overall algorithm's complexity is given by the running time of Step 2, multiplied by the number of iterations. Since we are aggregating at least one point at each iteration, the worst-case complexity of the algorithm is $O(N^3)$.

The bound is achieved if and only if

1. At each step of the algorithm, only $O(1)$ faces are intersected by segments of $\overline{V_i}$.

2. At each step of the algorithm, only $O(1)$ points are aggregated into the graph. This can only happen if the above condition is satisfied *and* the cardinality of the set of maximum depth is $O(1)$.

Because of the recursive nature of the algorithm, we can reasonably hope that the

partition of segments into faces is well-balanced, and that a large number of points gets aggregated at each algorithm iteration. For example, suppose that for every subgraph face $f \in G_i^f$, the number of points of p^f of maximal depth is $O(n)$, where n is the cardinality of p^f. Then the number of iterations is $O(1)$ and the algorithm requires $O(N^2)$ operations. Additionally, suppose that at every iteration, the number of faces of $\mathcal{M}_i$ that get crossed by segments is $O(N)$. Then each p^f set is at most $O(1)$, and Step 2 is performed in constant time a constant number of times. Step 1 however is incompressible, giving the algorithm total execution speed of $O(N \log N)$. The claim that the *expected* speed is smaller than $O(N^3)$ is warranted by the experimental execution speeds for the example runs we report in the next sections.

5.2.6 Example

We implemented the above algorithm on real noisy three-dimensional data. We gathered the data with a two-dimensional triangulation-based synchronized laser range finder developed at the NRC (Rioux, 1984). The subject was the pencil holder shown in Figure 1.6. The essential criterion for the choice of subject was that it be homeomorphic to a sphere, so as to be properly modeled by a polyhedron. A second criterion for the subject was that it not be convex, since a convex object is trivially modeled by the graph G_0, and therefore it does not test the algorithm. The pencil holder is a very simple shape, since it contains only one *deficiency* [3], but is still difficult to model from purely geometric tests because of that deficiency's large size. Also note that the deficiency is itself convex, making the pencil holder a *strongly visible* polyhedron. Such a polyhedron has the property that any point of its deficiencies is visible from any point on their *lid*. The concavity lid is the face of the hull separating

[3] We borrow the term from the Computational Geometry literature. In this chapter, we reserve the term *concavity* to refer to sets of *merged* faces (See Section 5.3).

the deficiency from the object's exterior. Strong visibility is a sufficient, although not necessary, condition for the object to be entirely visible from its exterior. A practical consequence is that we do not have to penetrate the hull's interior, thereby avoiding the complications depicted in Figure 5.2.

The subject measures approximately 20cm in each dimension. Several thousand three-dimensional surface data points were taken from three different viewpoints from a distance of about 60cm, and were smoothed with a Gaussian filter. All three views were designed to sample a substantial portion of the deficiency. The accuracy of the measurements using this setup is better than 1mm. The inter-view calibration was done with a least-squares minimization technique on a set of seven fiducial points, namely the top of the pyramids shown in Figure 1.6. We empirically verified that the absolute positional error after transforming the data to a common frame was as large as 3–4 millimeters. As a result, we used no more than a few hundred points to test the algorithm. Equal numbers of points were selected from the three views. Within each view, the sampling of the data points was made across approximately equal solid angles. Other than this data point selection mechanism, we made no further use of the fact that the data points came from a small number of views, rather than as a set of isolated acquisition segments.

The model output by the algorithm from an input set of 250 data points is shown in Figure 5.5.

The results are somewhat disappointing from a perceptual point of view. The main reason is that the edges from the higher level always remain in the graph. Where a large deficiency is present, as is the case in the example, each higher-level face gets carved as if *it* contained its own separate deficiency. As the carving proceeds, these "separate" deficiencies get larger and eventually criss-cross.

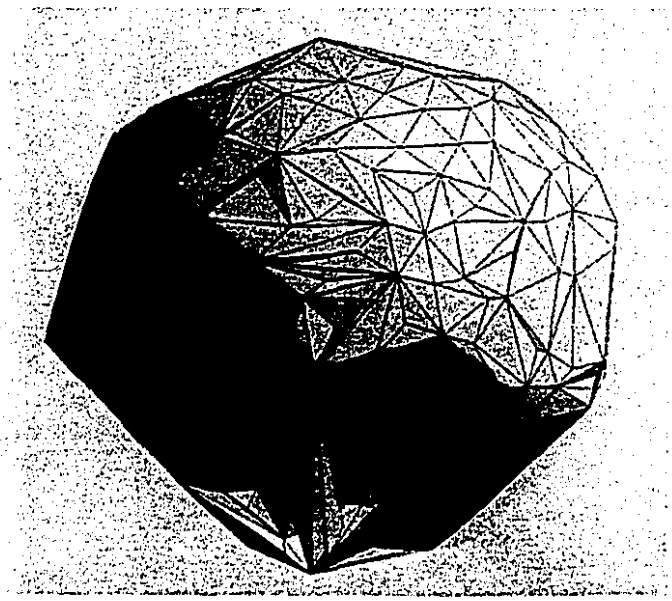

Figure 5.5 Model output by the algorithm from an input set of 250 surface points acquired off the pencil holder shown in Figure 5.12.

In order to alleviate the problem, we added a heuristic to the algorithm to allow the "merging" of several higher-level faces in cases where a deficiency encompassing several of these faces is likely to be present. We present the heuristic in the next section.

A second problem pertains to the planar-facet representation itself and also shows up in the form of self-intersections for the resulting model. Claim 3 only shows that Proposition 2 is true *locally.* It shows that the acquisition segments of the vertices spanned by a local graph G^f do not intersect any face of G^f. However, they still *may* intersect the faces of $G^{f'}$, where $f \neq f'$ (we then say loosely that the subgraphs G^f and $G^{f'}$ *intersect*).

Unfortunately, as shown by the tests presented above, such situations do frequently arise in practice. One reason of course is that the data is insufficiently sampled to

correctly and unambiguously reconstruct the object. As well, the sensing errors introduce local spurious artifacts. But a more fundamental reason is that the simplicial representation we adopted is an arbitrary representation for the underlying surface. In practice, it tends to "carve out" too much of the object's enclosed volume.

Consider the simple case illustrated in Figure 5.6. f and f' are two faces, each of which is intersected by one acquisition segment. In this case, no segment-to-face intersection is present.

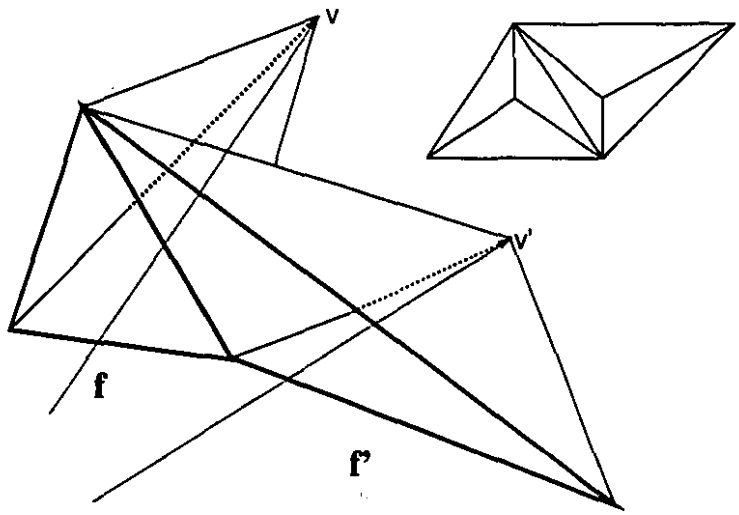

Figure 5.6 f and f' are two (neighbor) faces of a graph G_i. The cardinality of both p^f and $p^{f'}$ is 1. The acquisition segments are shown for the singletons $v \in p^f$ and $v' \in p^{f'}$. On the right of the figure, the corresponding graph for G^f_{i+1} and $G^{f'}_{i+1}$ is shown.

If we modify the position of the data points in Figure 5.6 to allow v' to translate towards the left of the figure, we reach a limit where v' penetrates through a face ϕ of F^f_{i+1}, as shown in Figure 5.7. As a result, $s(v')$ intersects ϕ, and the graph violates Proposition 2.

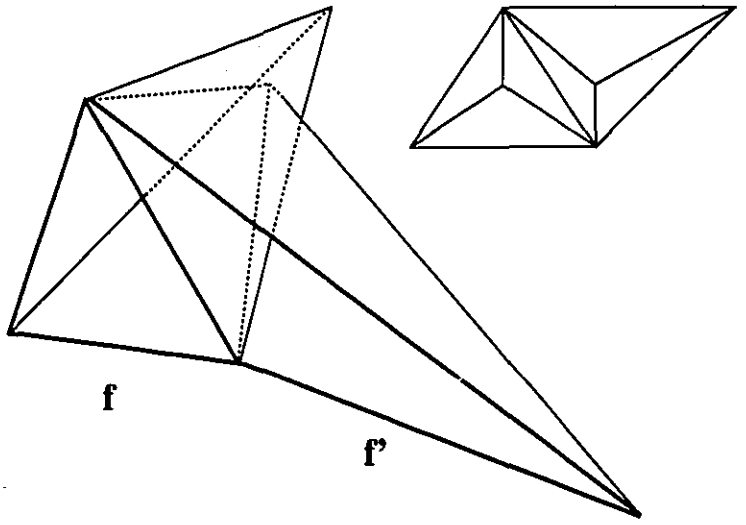

Figure 5.7 The faces of G^f and $G^{f'}$ intersect. The graph shown on the right now violates Proposition 2. For clarity, the acquisition segments are omitted from this and the following figures.

This happens because the volume of the deficiencies carved out by the convex $\mathcal{M}_i$ objects is too large. As mentioned at the end of Section 3.1.2.3, the implicit visibility information provided by the acquisition segments is only that there exists an ϵ-diameter cylinder of free-space around each segment, instead of a polyhedron of free-space rooted at face f. The polyhedral representation we adopt is one of convenience.

Hence, suppose that we fix face ϕ at its vertices, and that we "bend" it away from v', thus making it non-planar, so as to allow v' (and hence $s(v')$) to not intersect ϕ any longer. Because we know that both s and s' entirely lie in free-space, there exists a topological mapping $\mathcal{T}$ such that the image of ϕ through $\mathcal{T}$ is a sheet with the same boundaries as ϕ, but which intersects neither $s(v)$ nor $s(v')$.

Hence, although the *geometry* of the modeled object is incorrect, the *connectivity*

of its vertices is consistent (in the sense of Section 1.3.2) with the data acquisition procedure. This is similar to the situation we encountered in Section 4.2.5.

Still the large number of segment-to-face intersections and of model self-intersections is a very undesirable feature of the algorithm. The heuristic we present in the next section reduces that number.

5.3 The Face-Merging Algorithm

Figure 5.7 illustrated a situation where one of the segments s of G^f crosses a face ϕ of $G^{f'}$. Clearly this violates the opacity assumption. While we argued in the previous section that we *could* make the model consistent with the acquisition procedure by relaxing the planar facet constraint, we present in this section a heuristic that reduces the number of such occurrences and also retains the same output representation.

We first illustrate the *face-merging* concept. Refer back to Figure 5.6. Faces f and f' both spawn a sub-tree, with vertex v and v' respectively. Each sub-tree models a deficiency, which is deemed to have been "discovered" by the intersecting segments $s(v)$ and $s(v')$. Thanks to the particular *geometry* of the data, the model drawn by the resulting graph is free of self-intersections.

In Figure 5.8, the same data configuration is shown, except that the convex layer algorithm has been performed by *merging* faces f and f'. More formally, the convex layer algorithm has been applied to the merged input set

$$V^{f,f'} = \{x_1, x_2, x_3, x_4\} \cup p^f \cup p^{f'},$$

where $f = (x_1, x_2, x_3)$ and $f' = (x_1, x_2, x_4)$.

We can see that the resulting boundary and graph-theoretic interpretation of Figure 5.8 is as consistent with the data as that of Figure 5.6. Yet the face-merging algorithm we described chooses Figure 5.6's interpretation. This choice agrees with the least-commitment principle outlined on Page 15. The chosen interpretation minimizes the volume enclosed by the models drawn by the subgraphs, hence it maximizes the volume of the resulting model (See the proof of Claim 2).

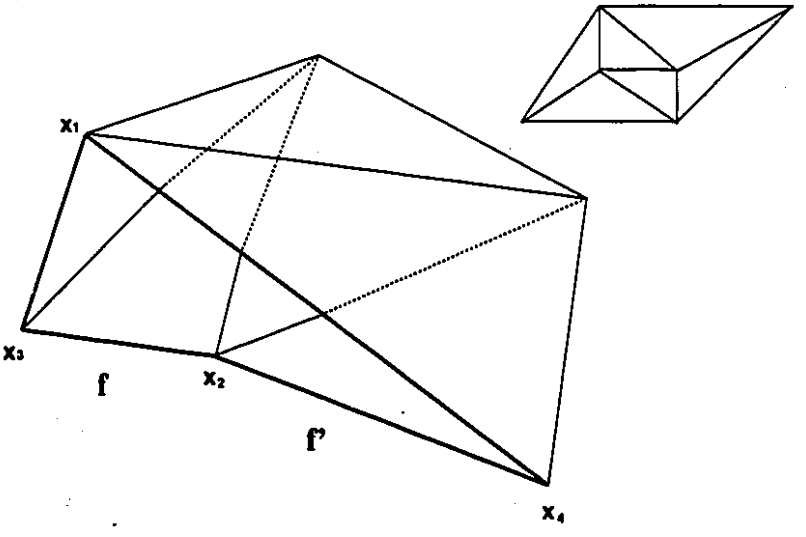

Figure 5.8 f and f' have been merged for the convex layer algorithm, resulting in a different graph.

Contrast the above situation with that which we illustrated in Figure 5.7, where the resulting *planar-facet* model violated the opacity condition. The same data geometry is shown in the companion Figure 5.9, but with the convex layer algorithm having been applied to the merged faces f and f'. As before, merging faces yields a different graph. Yet it also eliminates the self-intersection between G^f and $G^{f'}$.

This example illustrates the need for merging faces in the case where neighboring

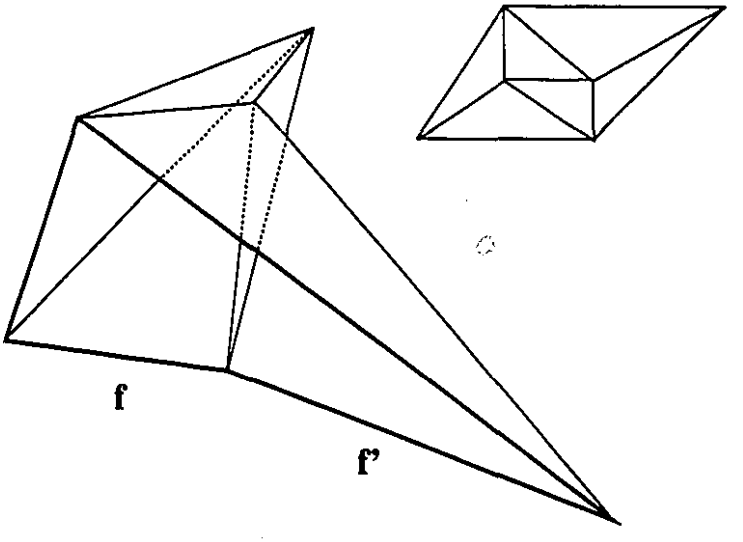

Figure 5.9 Modified graph for f and f'. The segments associated with the segments of vertices of $V^{f,f'}$ do not now intersect any face of $G^{f,f'}$.

deficiencies intersect. In the next subsection, we formalize the face-merging conditions for any set of graph faces.

5.3.1 The Face-Merging Conditions

Let $\overline{G_i}$ be the *dual graph* of G_i. If we assign a uniform weight (say of 1) to the edges of $\overline{G_i}$, we can calculate the shortest path between any two faces of G_i. Let $sp(f, f') =< f, f_1, \ldots, f_{D-2}, f' >$ be the shortest path between faces f and f'. $D(f, f')$ is the *distance* between f and f' along the shortest path.

Recall that model self-intersection occurs in a large part when a unique deficiency to be modeled by G_{i+1} is "covered" by more than one face of G_i. Figure 5.10 illustrates a geometry where this takes place. If the subgraphs rooted at the faces shown in

the figure do not intersect, we assume, in the absence of better knowledge, that the subgraphs each correspond to a separate deficiency and no merging takes place. If on the other hand, they do intersect, their parent faces are candidates for merging.

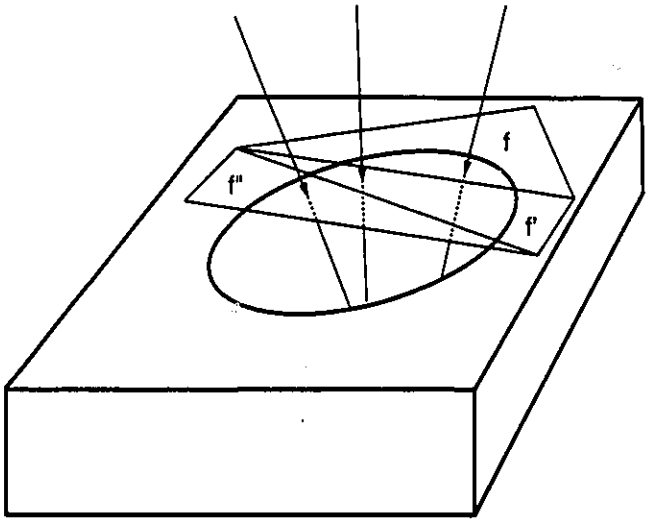

Figure 5.10 More than one graph face may make up a deficiency lid. In this figure, f, f' and f'' cover a hole. They are intersected by acquisition segments which travel down to the bottom of the hole. If their corresponding subgraphs intersect two-by-two, the faces are merged.

The additional merging conditions we now outline are designed to ensure that

1. The heuristic does not degrade the algorithm's average computational complexity.

2. The resulting graph remains a 3CMPG.

The first condition is met by setting an upper bound D on the faces' mutual distance. f and f' are not checked for intersection if $D(f, f') > D$. This ensures that the face-merging process remains *in general* local. This point is further explained below.

The second condition is met by ensuring that the merged faces do not contain a cycle. It they do, there must exist a data point v internal to that cycle (See Figure 5.11). Because the face-merging step breaks all the edges separating the merged faces, the existence of a cycle in the set implies that v becomes an isolated vertex. This situation cannot be allowed if the graph is to remain 3-connected. Consequently, the face-merging algorithm does not attempt merging face sets which form or contain a cycle. The dual of a triangulated polygon is a tree (O'Rourke, 1987, Chapter 1). Therefore, a set of faces is acceptable for merging if and only if the dual of these faces form a tree.

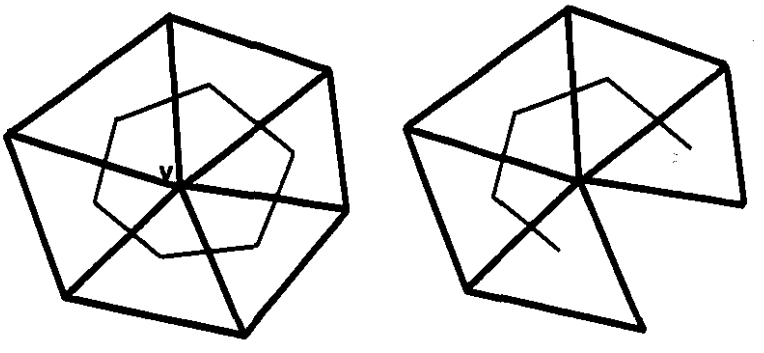

Figure 5.11 The faces at left (shown in bold) cannot be merged together, as indicated by their dual, which forms a cycle. If they belong to a unique equivalence class, the algorithm does not attempt to merge them. In contrast, the faces at right can be merged.

We now give a more detailed description of the face merging process for a given iteration i. The complete algorithm is further described in pseudo-code in Appendix B.4. We assume we have constructed the G_i^f subgraphs for each face of F_{i-1} as described in Section 5.2.4. Then we construct the dual graph $\overline{G_{i-1}}$. For each face f of F_i, we check if the subgraph rooted at f intersects any of its neighboring subgraphs, up to a graph distance of D. If we detect no intersection, we proceed to iteration $i+1$ as

before.

Suppose however we that we detect some subgraph intersections. Then for each pair of intersecting subgraphs, we merge the faces at which they are rooted, along with all the faces that lie on their shortest path. We call the merged faces *concavities*. Let C be such a concavity and let F be the set of faces that comprises it. The convex layer algorithm is then reapplied for C, by setting

$$p^C = \bigcup_{f_i=\{i_1,i_2,i_3\}\in F} p^{f_i},$$
$$V^C = \bigcup_{f_i} \{i_1, i_2, i_3\} \bigcup p^C.$$

Then the subgraph intersection detection is also repeated for the concavities. If intersections between the concavity subgraphs are found, these subgraphs are themselves merged, thus growing the concavities further.

Note that the bound D does not necessarily preclude f and f' from being merged if their mutual distance exceeds D. If the subgraphs of the faces lying on $sp(f, f')$ intersect two-by-two in such a way that none of the intersecting face pairs have a mutual distance greater than D, than all faces on the path, including f and f', may be merged. Hence D does not in principle limit how far the merging process eventually extends. A concavity can theoretically grow to comprise a number of faces in the order of the input size. For this reason, the face-merging heuristic theoretically degrades the worst-time algorithmic complexity by a linear factor. In practice, the process stops early, thanks to the existence of the distance bound D, or due to the detection of a cycle in the dual graph. In the latter case, the algorithm reports the existence of the cycle and does not attempt to grow the offending concavity any further.

We implemented the algorithm and tested it on experimental, noisy data. Figures 5.12

and 5.13 show a snapshot of the object first shown in Chapter 1 (a pencil holder) and its planar-facet representation through three iterations. The object is made up of 300 distinct data points. The maximum graph distance D for subgraph intersection testing was set at 3.

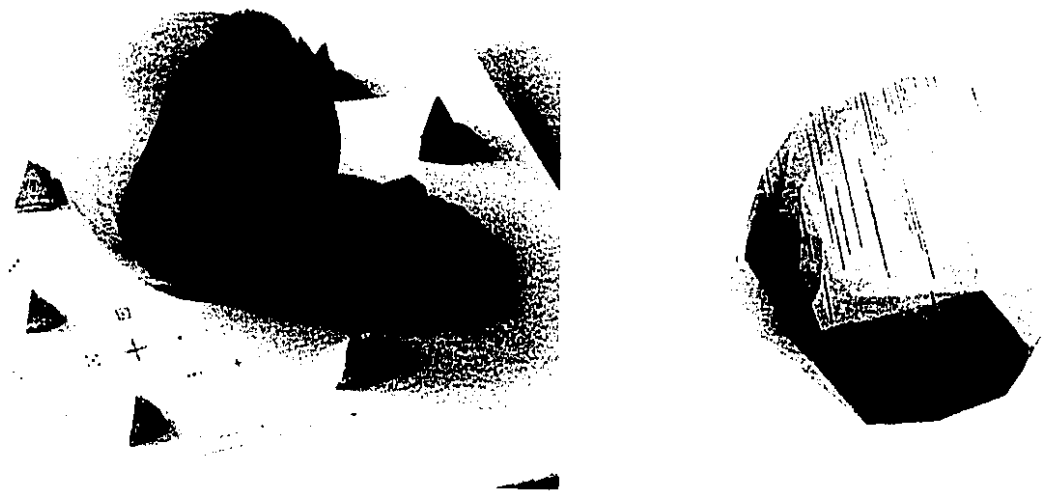

Figure 5.12 **Left**: A round object with a deep deficiency.
Right: The zeroth-order graph of the object (the convex hull). The black lines are the range finder's line-of-sight rays for those points which make up the next order's graph.

The average actual running speed of the two algorithm versions are shown in Table 5.1 for a series of three tests. For the regular algorithm, the complexity progression is approximately quadratic, as expected. While the face-merging algorithm takes much longer to terminate, its only exhibits a complexity progression slightly greater than quadratic.

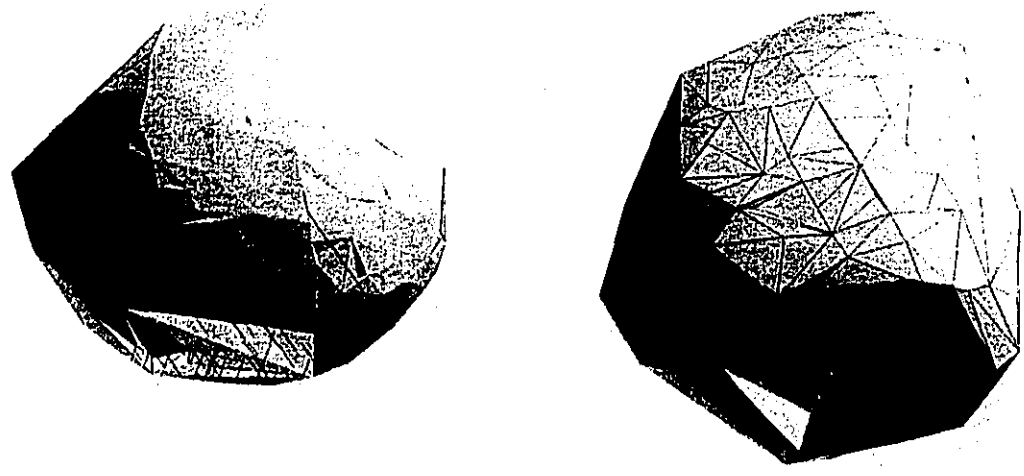

Figure 5.13 **Left**: The first-order graph of the object. Artifacts on the round part of the object are caused by calibration inaccuracies. The deficiency is now only partially carved.
Right: The second- and final-order graphs of the object.

5.4 Chapter Summary

We have presented a global algorithm for automatically constructing the connectivity graph of a set of points that lie on the surface of an object homeomorphic to a sphere. We assumed the points are acquired with a line-of-sight sensor and the details of the acquisition procedure are known. The algorithm uses the convex hull of the data points as its initial shape and builds successive iterations by carving the previous iterations' faces. A consequence is that the more detailed representations are strictly included inside the less detailed ones, making it well-suited for applications such as path planning that require hierarchical representations. The algorithm has a complexity at worst cubic in the input size, and approximately quadratic in practice. In order to obtain more perceptually-pleasing results, we implemented a heuristic to

Number of Input Points	Regular Algorithm	With face-merging
20	.1	.1
40	.3	.3
80	.8	3
160	4	18
320	17	80
640	68	445
1280	295	1987

Table 5.1 Actual running-time (in User Time seconds) of the two algorithm versions for different input sizes. The non-optimized code was run on a SPARC 1 workstation using the C++ programming language.

reduce the number of model self-intersections, when the graph assumes a planar-facet representation.

Chapter 6

Conclusions

The automatic integration of three-dimensional scenes from multiple viewpoints motivated the work we presented in this thesis. We reviewed the existing literature on the topic, and we saw that image integration is a large body of research in the computer vision field. As a result, many approaches perform data integration through image analysis and object reconstruction techniques, which are a form of data compression. Recent technological advances in range data acquisition have spurred interest in adapting these techniques to range images. We argued that the geometrical nature of range data, coupled with knowledge of the details of the data acquisition procedure, allows for a different form of processing. This processing obeys what we called the ASDT paradigm. The ASDT paradigm delays the data compression performed by image understanding and object reconstruction techniques. Rather, the output of the ASDT is a general-purpose least-commitment representation.

The least-commitment representation decouples the multiple viewpoint integration phase, which is low-level and purely geometric, from the object reconstruction phase,

which is high-level and context-dependent.

Thus we found that an essential task of the ASDT is to determine the *connectivity* of the surface data points. Connectivity for a set of discrete data points can be expressed as a graph, whose vertices form the data point set. This connectivity is often simply assumed, or computed solely based on the basis of the Euclidean distance between the points. When the surface resolution for the acquired scene is high, this may be a reasonable thing to do. For example, an implicit and correct connectivity is often assumed between neighbor points in pixel-based images. However, when the resolution is coarser than the object features, we showed that neither image neighboring relationships nor Euclidean proximity lead to accurate inferences of surface connectivity.

We observed that the topological class we model must be known ahead of time in order to properly determine connectivity. Hence, we looked at the case of the simplest two-dimensional topological class, namely the sphere, which is homeomorphic to the polyhedron. We then showed that the theoretical number of polyhedral graphs that can be drawn on a given data set is prohibitively large. We then introduced the assumptions of surface opacity and rigidity in order to prune the number of admissible graphs. These assumptions are useful only if additional information is known about the data acquisition procedure. Such information led us to model range data acquisition as a process whose output is a set of surface data points, each of which lies at the boundary of the object being acquired, and of a subset of free-space. The geometry of these subsets depends entirely on the data acquisition technique. For many of these techniques, including triangulation-based laser scanning, the subset can be approximated as a line segment that extends from the acquisition device's imaging

center to the data point. This knowledge yields additional information, which complements that of the coordinates of the data point. Then we demonstrated that this geometrical information helps in reconstructing surface connectivity.

The two-dimensional equivalent problem was already solved. A polygon in a plane can be unambiguously and efficiently reconstructed from a set of line segments terminating on its contour. At the outset, we determined that reconstruction uniqueness cannot be guaranteed for three-dimensional objects probed under similar conditions. Yet, the question remained as to whether "perceptually-consistent" polyhedral connectivity can be constructed from simple geometric considerations.

We developed `merge-frames`, an "on-line", *i.e.* incremental algorithm for doing so in the case where the input is grouped into a set of *frames*, where a frame is an abstraction for a matrix of data points with a common imaging center. The frame is a convenient data organization model for range "images". One of the main features of this algorithm is to use the partial connectivity provided by the individual frames, as initial subgraphs for the modeled object. Further, the structured form of the frames allows us to merge the subgraphs using a sequence of efficient binary searches. The object of the binary search is to determine geometrical relationships, such as the intersection of acquisition segments with planar "sheets" defined by an acquisition center and a subset of its data points.

We gave precise definitions for graph *validity* based on object opacity and visibility considerations. We proved several results for the algorithm, but we did not deem those results to be sufficiently powerful to warrant implementation and testing. We found counter-examples that made the graph produced by the binary searches invalid. The major failure mode of the algorithm occurs when the object's concavities are acquired before its protuberances. This has the effect of invalidating the intersection

tests performed *before* the acquisition of the protuberances.

`merge-frames` showed that tests based on partial information may be invalidated by subsequent data. Hence we developed a *global* algorithm, for which all data must be acquired before attempting to establish connectivity. An advantage of this algorithm is that it makes no assumption about the organization of the data points. In particular, it does not assume that range data is organized along frames, or images.

This algorithm is based on the iterative "carving" of the graph faces, starting from the convex hull of all points. Hence, the algorithm proceeds "from the outside in" by incorporating an ever larger set of data points into the current graph. The algorithm terminates when all points are inserted into the graph. At each iteration, the algorithm partitions the points internal to the current iteration's model, based on the intersection of faces with the segments associated with those points. New faces are created by joining the intersected face with the points of the maximum depth convex layers for that partition. The remaining points are themselves further subpartitioned at the start of the next iteration. This algorithm is therefore recursive as well, and displays an average computational complexity which is quadratic in the number of data points, although the worst-case complexity is cubic. The algorithm produces a hierarchical representation. Each node is a model, from the less detailed convex hull at the top, to the more detailed spanning graph at the bottom. The deeper nodes are guaranteed to lie inside the higher ones.

We tested the algorithm on an object containing a large concavity, on experimental, noisy data obtained from a triangulation-based laser range finder. Even though the connectivity information calculated by the algorithm was essentially correct, we found that it did not yield perceptually pleasing results. One reason was that we chose to *display* the results with a planar-facet model, and by doing so, we made assumptions

about the object's geometry that were not present in the original data, nor in the constructed graph. A second reason pertains to the algorithm itself and has to do with the fact that there is no one-to-one correspondence between the concavities present in the object, and the faces of the higher-level graphs. We introduced a heuristic to remedy this problem. The heuristic merges faces that appear to "cover" the same concavity. These faces are detected by testing for model self-intersections. We found that this heuristic reduced the number of planar-facet model self-intersections, for a computational cost at most linear, and in practice sub-linear, with respect to the size of the input.

6.1 Further Work

The approach we followed in this work applies mostly to scenes acquired with a low- to medium-resolution. One reason is that high resolution acquisition yields an implicit connectivity which is generally correct. The connectivity is simply inferred from Euclidean distance or from image plane nearest-neighbor relationships. A second reason is that the planar faces are a very crude underlying representation for planar graphs. When the cardinality of the input set increases, our algorithm is not robust enough to handle the greater probability for model self-intersections.

This second problem, however, can be circumvented by developing a better underlying geometrical representation for the resulting graph. For example, curving inward the faces of the subgraphs rooted at a given face, would reduce the volume enclosed by the model drawn by that subgraph. As a result, the number of subgraph intersections would be reduced, resulting in a perceptually more pleasing model.

Another approach to reduce the number of self-intersections would be to experiment with other heuristics for the global face-merging algorithm. For example, the graph distance beyond which faces are not merged, can be made to depend on the geometric resolution.

Yet, the cause at the root of the difficulties we encountered in showing the generality of the approach is due to the nature of the input data. Points and line segments are zero- and one-dimensional entities that are used to reconstruct a three-dimensional object. As a result, no three-dimensional subset of space can ever be unambiguously classified as free or empty. If the w free-space regions were three-dimensional, as with contact sensing or CMM, the data reconstruction would be much less ambiguous. However segment-based graph construction is justified by technological considerations.

Handling other topological classes such as single- or multi-hole tori would be a generalization of this work. We stressed that in order to maintain graph consistency, the topological class of the desired model must be known. Such knowledge could be an ASDT's input variable. Then, when a segment is found to "penetrate" through the object, it can serve as a basis for "growing" a topological hole.

In conclusion, we showed that range data acquisition allows the automatic construction of least-commitment connectivity models with simple geometrical tools. Further work remains to be done to generalize this approach, to increase its robustness, to improve the perceptual appearance of the resulting model, and to study how it can be used in conjunction with the higher-level processes of three-dimensional analysis.

Appendix A

The Number of Labeled 3-connected Maximal Planar Maps

A.1 Labeled and Unlabeled Enumeration

In the first paragraph of their seminal book on graphical enumeration (Harary and Palmer, 1973), Harary & Palmer state: "Labeled enumeration problems always appear to be much easier to solve than the corresponding unlabeled problems." This is true because the solution of an unlabeled enumeration problem require the computation (explicit or implicit) of the number of symmetries that various graphs have. Unfortunately, no general method exists to detect symmetries, and solutions to unlabeled enumeration problems often involve ingenious but ad hoc methods.

We shall see that Harary & Palmer's statement applies to the problem of enumerating 3-connected maximal planar graphs (3CMPG's). Even though no exact formula is known for enumerating unlabeled 3CMPG's, we derive such a formula for the labeled case. We could not find such a formula in the existing literature.

A.2 The Connect-the-dots Problem

Let V be a set of n points embedded in the 3-dimensional Euclidean space. Suppose we wish to connect the points of V to form a fully-triangulated polyhedron P of n vertices. It is well-known that the set of vertices and edges of P form a 3CMPG G. G is maximal because since every face of P is a triangle, no edge can be added to G without losing the planarity property. It is also well-known that every 3CMPG can be realised as a fully-triangulated polyhedron (Grűnbaum, 1967, page 235).

We wish to calculate the number of different such polyhedra realisable from V. By the above observation, this number is the number of different 3CMPG's. Because the points of V are assumed to have a specific embedding however, different though isomorphic graphs yield different space-occupancy functions, and hence correspond to different polyhedra. As a result, we will concern ourselves with enumerating *labeled* 3CMPG's.

A.3 A review

The best account on the research status of enumeration of 3-connected planar graphs was given by Federico (Federico, 1975). In this review, Federico concerned himself

with the enumeration of unlabeled graphs, because of the one-to-one correspondence between polyhedra *types* and unlabeled 3-connected planar graphs. He indicated that no exact solution was known, and that rote enumeration had not even been carried beyond 12 vertices.

A.3.1 Edge-rooted planar triangulations

Tutte (Tutte, 1962a; Tutte, 1962b; Tutte, 1963) was the first to make significant contributions to the theory of enumerating planar graphs.

He tackled the problem by *rooting* planar graphs. A rooted planar graph is a planar graph embedded on the sphere, where a particular edge and the embedding sphere have both been assigned a positive orientation. Hence, a planar graph with e edges gives rise to $4e$ rooted graphs.

In (Tutte, 1962a), Tutte showed that the effect of rooting a graph is to destroy any symmetry present in the unrooted graph. He went on to give a recursive formula for the number of rooted 3-connected planar graphs, and an explicit formula $\phi(n)$ for the number of rooted 3CMPG's on n vertices [1].

$$\phi(n) = \frac{2(4n-11)!}{(n-2)!(3n-7)!} \quad , \tag{A.1}$$

where we slightly changed Tutte's notation to suit the definitions given in this note.

From the above observation on the number of rooted graphs, we can conclude that

[1] Brown (Brown, 1964) later generalized Tutte's explicit formula to all rooted 3-connected planar graphs.

the number of unlabeled 3CMPG's is bounded above by $\phi(n)$ and bounded below by $\frac{\phi(n)}{4e}$.

The upper bound is true since we know that $\phi(n)$ is the number of non-isomorphic rooted 3CMPG's, and that there at least as many rooted 3CMPG's as there are unrooted 3CMPG's. The lower bound is true by the above observation on the number of edge-rooted graphs obtainable from a given unrooted graph.

Tutte also showed that to each unrooted 3CMPG G corresponds $\frac{4e}{\Gamma(G)}$ rooted 3CMPG, where $\Gamma(G)$ is the number of symmetries of G (technically, the order of the automorphism group of G).

It follows that the number $U(n)$ of unlabeled, unrooted 3CMPG's of n vertices is

$$U(n) = \sum_i \frac{\phi_i(n)}{\frac{4e}{i}}, \tag{A.2}$$

where $\phi_i(n)$ is the number of 3CMPG's having i symmetries. Further, the index i is known to vary over the set of divisors of $4e$ (Harary and Tutte, 1966).

A.3.2 Unlabeled Planar Triangulations

Tutte (Tutte, 1962a) conjectured that the number of unlabeled 3-connected planar graphs tends to $\frac{\phi(n)}{4e}$ as n tends to infinity. In other words, almost all 3CMPG's are unsymmetric for large n, and the ratio of symmetric to unsymmetric 3CMPG's tends to zero as the number of vertices tends to infinity.

Bender and Wormald (Bender and Wormald, 1985) later proved that the fraction of e-edged 3-connected planar graphs which are symmetric is at most $O(c^e)$, with $c < 1$.

Since for maximal graphs, e and n are linearly related, it proves Tutte's conjecture for 3CMPG's. Therefore,

$$\lim_{n\to\infty} U(n) = \frac{\phi(n)}{4e}. \tag{A.3}$$

A.4 Labeled Planar Triangulations

The number of ways of labeling a given graph G of order n is (Harary and Palmer, 1973, page 4)

$$l(G) = \frac{n!}{\Gamma(G)}. \tag{A.4}$$

Let $L(n)$ be the number of labeled 3CMPG's of n vertices. By (A.4),

$$\begin{aligned} L(n) &= \frac{U(n)n!}{i} \\ &= \sum_i \frac{\phi_i(n)}{\frac{4e}{i}} \frac{n!}{i} \\ &= \sum_i \frac{\phi_i(n)n!}{4e} \\ &= \frac{\phi(n)n!}{4e} \\ &= \frac{\phi(n)n!}{12(n-2)}, \end{aligned} \tag{A.5}$$

where we use the relation $e = 3n - 6$, which holds for all 3CMPG's.

Substituting (A.1) into (A.5), we get

$$\begin{aligned} L(n) &= \frac{2(4n-11)!}{(n-2)!(3n-7)!}\frac{n!}{12(n-2)} \\ &= \frac{(4n-11)!}{(3n-7)!}\frac{n(n-1)}{6(n-2)} \\ &= \frac{(4n-11)!}{(3n-6)!}\binom{n}{2}. \end{aligned} \tag{A.6}$$

A.5 An Asymptotic Formula

We now compute an asymptotic formula for $L(n)$ in order to estimate its rate of growth. Setting $m = n-3$ and using Stirling's factorial formula, (A.6) becomes

$$\begin{aligned} L(n) &\sim \frac{(4m+1)^{4m+1}}{(3m+3)^{3m+3}}\frac{\sqrt{2\pi(4m+1)}}{\sqrt{2\pi(3m+3)}}\frac{e^{-4m-1}}{e^{-3m-3}}\frac{(m+3)(m+2)}{2} \\ &= \frac{(4m(1+\frac{1}{4m}))^{4m+1}}{(3m(1+\frac{3}{3m}))^{3m+3}}\sqrt{\frac{4m(1+\frac{1}{4m})}{3m(1+\frac{3}{3m})}}e^{-4m-1+3m+3}\frac{m^2(1+\frac{3}{m})(1+\frac{2}{m})}{2} \\ &\sim \frac{(4m)^{4m+1}e}{(3m)^{3m+3}e^3}\sqrt{\frac{4}{3}}e^{-m+2}\frac{m^2}{2} \\ &= \left(\frac{256}{27e}m\right)^m \frac{4\sqrt{3}}{81} \\ &= \frac{4\sqrt{3}}{81}\left(\frac{256}{27e}(n-3)\right)^{n-3}. \end{aligned} \tag{A.7}$$

A.6 Conclusion

No explicit formula exists for the number of unlabeled 3CMPG's. By using Tutte's formula for the number of rooted 3CMPG's and the general relation between the numbers of labeled and unlabeled graphs, we easily derived an explicit formula for the number of labeled 3CMPG's. This number is the number of different fully-triangulated polygons that can be drawn on a set of vertices embedded in E^3.

Table A.1 displays the numbers for particular values of n, for both the exact number and its asymptotic approximation. We checked the results by rote enumeration up to $n = 6$.

n	$L(n)$ (exact formula)	$L(n)$ (asymptotic formula)
4	1	.2983
5	10	4.163
6	195	98.00
7	5712	3241
8	223,440	138,005
9	1.093e+07	.7187e+07
10	6.413e+08	4.424e+08
11	4.386e+10	3.144e+10
12	3.424e+12	2.532e+12
13	3.004e+14	2.280e+14
14	2.926e+16	2.269e+16
⋮	⋮	⋮
120	2.534e+304	2.466e+304

Table A.1 Exact and asymptotic number of labeled 3-connected maximal planar graphs expressed as a function of the number of vertices. The greatest value we could compute using standard 8-byte floating point arithmetic was for $n = 120$.

A.7 Proofs of Theorems

A.7.1 Proof of Theorem 1

We first prove the two following lemmas:

Lemma 1 *$\mathcal{P}^\alpha : V^\alpha \longrightarrow V^{\alpha\prime}$ is bijective.*

Proof: $\mathcal{P}$ is injective by definition, and so is its restriction. Since $V^{\alpha\prime}$ is the image set of V^α, every element of $V^{\alpha\prime}$ has a pre-image. Now suppose:

$$(\exists x' \in V^{\alpha\prime})(\exists x, y \in V^\alpha), \mathcal{P}^\alpha(x) = \mathcal{P}^\alpha(y) = x'.$$

Then $\theta_x = \theta_y$ and $\phi_x = \phi_y$. But then $\exists \lambda > 0, \overrightarrow{P^\alpha x} = \lambda \overrightarrow{P^\alpha y}$, which violates our assumption. Therefore, every element of $V^{\alpha\prime}$ has only one pre-image through $\mathcal{P}^\alpha$. So $\mathcal{P}^\alpha$ is also surjective, which proves the Lemma. □

Lemma 2 *G^α and $G^{\alpha\prime}$ are isomorphic.*

Proof: By Lemma 1, $\mathcal{P}^\alpha$ is a bijection between V^α and $V^{\alpha\prime}$. Further, (4.3) implies there exists a bijection between $E(G^\alpha)$ and $E(G^{\alpha\prime})$. Therefore, G^α and $G^{\alpha\prime}$ are isomorphic (Bondy and Murty, 1976). □

We now prove the theorem.

Notation:

- $\forall C$ s.t. C is a closed, compact set, $I(C)$ is the interior of C.
- $\forall C$ s.t. C is a closed, compact set, $B(C)$ is the boundary of C.

We wish to prove that if $G^{\alpha\prime}$ is a planar triangulation, then G^{α} is a triangulation and is 2 1/2-consistent. Since G^{α} and $G^{\alpha\prime}$ are isomorphic, $G^{\alpha\prime}$ is a planar triangulation if and only if there exists an embedding for which G^{α} is a planar triangulation. This is true by assumption. Hence the first validity condition is satisfied by Lemma 2.

We must now prove that G^{α} is 2 1/2-consistent. We will do so by showing that if either consistency condition on G^{α} is not verified, then $G^{\alpha\prime}$ cannot be a triangulation.

Let A be the set of closed triangles defined by the triplets of $(V^{\alpha})^3$. Since $\mathcal{P}$ is a central collineation, it is incidence-preserving, so for every closed, planar curve $C \subset E^3$, the following four statements hold:

$$\mathcal{P}(C) \text{ is a closed curve,}$$

$$\Big(x \in B(C)\Big) \Rightarrow \Big(\mathcal{P}(x) \in B\Big(\mathcal{P}(C)\Big)\Big),$$

$$\Big(x \in I(C)\Big) \Rightarrow \Big(\mathcal{P}(x) \in I\Big(\mathcal{P}(C)\Big)\Big).$$

$$\mathcal{P}(\text{closure of } C) = \text{closure of } \Big(\mathcal{P}(C)\Big).$$

In particular, $\forall t \in A$,

$$\mathcal{P}(t) \text{ is a triangle on } S^2,$$

$$\Big(x \in B(t)\Big) \Rightarrow \Big(\mathcal{P}(x) \in B\Big(\mathcal{P}(t)\Big)\Big), \tag{A.8}$$

$$\Big(x \in I(t)\Big) \Rightarrow \Big(\mathcal{P}(x) \in I\Big(\mathcal{P}(t)\Big)\Big). \tag{A.9}$$

$$\mathcal{P}(\text{closure of } t) = \text{closure of } \mathcal{P}(t).$$

Suppose there exists a segment s_i terminated at point x_i such that condition (4.1) is not verified. Then:

$$\exists y \in E^3, \exists t \in F^\alpha, \Big((y \neq x_i) \wedge (y \in t) \wedge (y \in s_i)\Big).$$

Since $(x_i \in s_i) \wedge (y \in s_i), \exists \lambda > 0, \overrightarrow{P^\alpha x_i} = \lambda \overrightarrow{P^\alpha y}$. Hence, $\mathcal{P}(x_i) = \mathcal{P}(y)$ since P^α is the center of the collineation $\mathcal{P}$.

Suppose now that $y \in B(t)$. From (A.8), $\mathcal{P}(y)$ must belong to $B\Big(\mathcal{P}(t)\Big)$, that is to an edge of $\mathcal{P}(t)$. Let x_1, x_2,and x_3 be the vertices of t and x_1', x_2', and x_3' be their respective images through $\mathcal{P}^\alpha$. Since $\{x_1, x_2, x_3\} \subset V^\alpha$, $x_1 \neq x_2 \neq x_3$. Then, $x_1', x_2', x_3' \in V^{\alpha\prime}$ and $x_1' \neq x_2' \neq x_3'$ by the bijectivity of $\mathcal{P}^\alpha$.

So, $\mathcal{P}^\alpha(t)$ has four vertices, namely x_1', x_2', x_3' and $\mathcal{P}^\alpha(x_i)$. Therefore, $G^{\alpha\prime}$ cannot be a triangulation.

Suppose now that $y \in I(t)$. From (A.9), $\mathcal{P}(y)$ must belong to $I\Big(\mathcal{P}(t)\Big)$. So, $\mathcal{P}(y)$ belongs to the vertex set of $G^{\alpha\prime}$ and belongs to the interior of a face of $G^{\alpha\prime}$. Therefore, $G^{\alpha\prime}$ cannot be a triangulation.

Hence in both cases $G^{\alpha\prime}$ cannot be a triangulation. So condition (4.1) must be true.

Similarly, suppose condition (4.2) is not satisfied. Then, $\exists x_i, (x_i \in s_i) \wedge (x_i \in T)$, where x_i is a datum point, s_i is its segment, and T is the visibility tetrahedron with one vertex at P^α and the others at the vertices of a triangle $t \in A$.

Let R be the semi-infinite segment terminated at P^α and whose support is $\overrightarrow{P^\alpha x_i}$. We first prove that $(R \cap t) \neq \emptyset$.

Since T is a convex figure, R intersects it in at most two points. By construction, one of those points is P^α. Further, R contains at least one point internal to T, namely, x_i. Hence the intersection of R with T is non-degenerate and the second intersection point must belong to a face other than those to which P^α belongs. But P^α belongs to all faces of T, except for t. Hence the second point of $R \cap T$ is in t. Let y be that point.

By construction, all points of R verify $\mathcal{P}(l) = \mathcal{P}^\alpha(x_i) = y'$. Hence $\mathcal{P}(y) = y'$.

We can now show as above that $y' \in V^{\alpha\prime}$ (since $\mathcal{P}(x_i) = y'$), and that either $\mathcal{P}(y) \in I(\mathcal{P}(t))$, (if $y \in I(t)$), or $\mathcal{P}(y) \in \mathcal{P}(t)$, (if $y \in B(t)$). Either case violates the assumption that $G^{\alpha\prime}$ is a triangulation, so condition (4.2) must also be true.

So G^α is 2 1/2-consistent, which proves the theorem. □

A.7.2 Proof of Theorem 2

If $\beta < 2\pi$, the polar angle transformation $\mathcal{P}$ maps V^α on an open hemisphere H around P^α. But for any open hemisphere there exists a bijective mapping $\mathcal{B}$ between a Euclidean plane and the open hemisphere. One such mapping is the projection with center P^α onto a plane Π tangent to H and parallel to its equator (Preparata, 1985, page 23). So Π always exists.

Since $\mathcal{B}$ is a central collineation around P^α, the polar coordinates of any point $x \in E^3$

are invariant under $\mathcal{B}$. Similarly, since $\mathcal{Q}$ is a central collineation around P^α, the polar coordinates of any point $x, x \in E^3$ are invariant under $\mathcal{Q}$. So, $\forall x \in V^\alpha, \mathcal{P}(x) = \mathcal{P}\Big(\mathcal{B}\Big(\mathcal{Q}(x)\Big)\Big)$ But since $\mathcal{B}\Big(\mathcal{Q}(x)\Big) \in H$, $\mathcal{B}\Big(\mathcal{Q}(x)\Big)$ is invariant by $\mathcal{P}$. Hence,

$$\forall x \in V^\alpha, \mathcal{P}(x) = \mathcal{B}\Big(\mathcal{Q}(x)\Big).$$

Suppose now that the graph $G^{\alpha\prime\prime}$ is a triangulation $V^{\alpha\prime\prime}$ in Π.

Let $G^{\alpha\prime}$ be the graph with $V(G^{\alpha\prime}) = \{x' \in H | \forall x'' \in V^{\alpha\prime\prime}, x' = \mathcal{B}(x'')\}$, and

$$\forall x \forall y \in V^{\alpha\prime\prime}, \Big(\overline{x,y} \in E\Big(G^{\alpha\prime\prime}\Big)\Big) \Leftrightarrow \Big(\overline{\mathcal{B}(x), \mathcal{B}(y)} \in E\Big(G^{\alpha\prime}\Big)\Big)$$

In the same manner as in Lemma 2, we can show that $G^{\alpha\prime}$ and $G^{\alpha\prime\prime}$ are isomorphic, thanks to the bijectivity of $\mathcal{B}$. Further, since $\mathcal{B}$ has the incidence property, it maps every triangle of $G^{\alpha\prime\prime}$ into a unique triangle on H. Therefore, $\mathcal{B}$ maps $G^{\alpha\prime\prime}$ into an isomorphic triangulation $G^{\alpha\prime}$ on H. Then, by Theorem 1, G^α is valid. □

A.7.3 Proof of Theorem 3

The proof of this theorem follows the lines of that given for Theorem 1. Replacing $\mathcal{P}^\alpha$ by $\mathcal{R}^\alpha$, we can prove as in Lemma 2 that G^α and $G^{\alpha\prime}$ are isomorphic and hence that G^α is a chain if $G^{\alpha\prime}$ is a chain.

Suppose now that condition (a) in Definition 5 is not verified. So there exists a segment s_i terminated at x_i such that

$$\exists y \in E^2, \exists e \in E(G^\alpha), \Big((y \neq x_i) \wedge (y \in e) \wedge (y \in s_i)\Big).$$

Since $(x_i \in s_i) \wedge (y \in s_i)$, we have $\overrightarrow{P^\alpha x_i} = \lambda \overrightarrow{P^\alpha y}$, with $\lambda > 0$.

So, $\mathcal{R}^\alpha(x_i) = \mathcal{R}(y)$. Let $x_l \in V^\alpha$ and $x_{l+1} \in V^\alpha$ be the vertices that edge e joins. By the bijectivity of $\mathcal{R}^\alpha$,

$$\mathcal{R}^\alpha(x_l) \neq \mathcal{R}^\alpha(x_{l+1}) \neq \mathcal{R}^\alpha(x_i).$$

So $\mathcal{R}^\alpha(x_i)$ belongs to e while it is not either of its endpoints. Hence $G^{\alpha'''}$ is not a chain if condition (a) is not verified.

Suppose now that condition (b) in definition 5 is not verified. Then

$$\exists x_i \in V^\alpha, \exists (e = \overline{x_l, x_{l+1}}) \in G^\alpha, \text{s.t. } x_i \in t,$$

where t is the triangle formed by x_l, x_{l+1}, and P^α.

Let R be the semi-infinite segment whose support is $\overrightarrow{P^\alpha x_i}$. As in the proof of theorem 1, we can easily show that R intersects edge e in a point y, such that $\mathcal{R}(y) = \mathcal{R}^\alpha(x_i)$. Because $\mathcal{R}$ has the incidence property, $\mathcal{R}(y) \in \overline{\mathcal{R}(x_l), \mathcal{R}(x_{l+1})}$. Hence, $\mathcal{R}^\alpha(x_i) \in \overline{\mathcal{R}^\alpha(x_l), \mathcal{R}^\alpha(x_{l+1})}$. As above, this implies that $G^{\alpha'''}$ cannot be a chain. □

A.7.4 Proof of Theorem 4

Let $P = \{P_1, \ldots, P_N\}$ be the set of successive positions O assumes on $\mathcal{C}$. $\forall x \in E^2$, let φ_x be the polar angle of x around P_1, subject to $\varphi_1 = 0$ and to $\varphi_i > 0$ for $i \neq 1$.

Let

$$\mathcal{R} : \{1, \ldots, N\} \longrightarrow [0, 2\pi],$$

$$i \mapsto \varphi_{P_i}.$$

Let C_1 be a circle centered at P_1. Let $g = \langle P, E(g) \rangle$ be a graph such that

$$\forall P_i, \forall P_j \in P, \left(\overline{P_i, P_j} \in E\,(g)\right) \Leftrightarrow (j = i + 1),$$

and let g' be its projection on C_1 as defined in Theorem 3.

Since O follows a closed, convex, planar curve, $\mathcal{R}$ is a monotonically increasing function of i. So g is a (convex) chain and so is g'.

Hence, the "frame"

$$\left\langle P_1, P, \{\overrightarrow{P_1 P_1}, \ldots, \overrightarrow{P_1 P_N}\} \right\rangle$$

is 2-valid by Theorem 3.

Now the acquisition segments lie on the directed tangents to $\mathcal{C}$. By the convexity of $\mathcal{C}$, these tangents never cross (See Figure 4.4). Hence, the projection of the data points onto C_1 yields the same monotonic ordering as did $\mathcal{R}$. But both the elements of P and the data points lie on the segments. Hence the graph $G = \langle V, E(G) \rangle$ defined by

$$\forall x_i \forall x_j \in V, \left(\overline{x_i, x_j} \in E\,(G)\right) \Leftrightarrow (j = i + 1)$$

is 2-valid. □

A.8 The merge-frames Incremental Merging Algorithm

The main procedure. A'' is the set of back-crossing segments and A''' is the set of back-crossing segments that eventually do not get inserted at all into G.

procedure merge-frames()
  $G = G^1$;
  $A'' = \emptyset$;
  for $\beta \in \{2, \ldots, \nu\}$
    $A'' = A'' \cup$ merge-one-frame (V^β, G);
  $A''' =$ merge-one-frame(A'', G);

This procedure merges one frame into the current graph G.

procedure merge-one-frame(V, G)
  $(A, A', B, O) =$ partition-one-frame(V, G);
  insert-one-frame(A, B, O, G);
  return (A');

This procedure partitions one frame with respect to the current graph G. It returns the partition to the calling procedure. i is the current frame index.

procedure partition-one-frame(V, G)
  foreach $f \in F(G)$
    $A_f = A_f' = \emptyset$;
  foreach $f \in (F(G) \cap (\bigcup_{\alpha=1}^{\beta} F^\alpha))$
    $B_{t_f} = \emptyset$;
  foreach triangle of the bounding sheets of all frames

$B_u' = \emptyset$;

$O = \emptyset$;

foreach $x^\beta \in V$

"S" =partition-one-point(x, G);

$S = S \cup \{x\}$;

return (A, A', B, O);

This procedure returns a string which is the name of the global set that the data point should be partitioned into. It iterates through the existing frames until a visibility face is found, with respect to whom the point corresponds to either a convexity or a concavity. Then it descends the subgraph of G rooted at that face until a face of G is reached.

procedure partition-one-point(x, G)
  $\alpha = 1$;
  do
    "S" =partition-one-point-with-a-graph(x, F^{α});
    if ("S" = "A_f'")
      return "A_f'";
    $\alpha = \alpha + 1$;
  until(("S" = "A_f") **or** ("S" = "B_{t_f}") **or** $(\alpha = \beta)$)
  if$(\alpha = \beta)$
    return ("O");
  while$\Big(($"S" = "A_f"$)$ **or** $\Big($("S" = "B_{t_f}") **and** $f \notin F(G)\Big)\Big)$
    "S" =partition-one-point-with-a-graph(x, G_f);
  return ("S");

This procedure partitions a point with respect to a visibility graph, or to a subgraph of G. i is the current frame index.

procedure partition-one-point-with-a-graph(x, g)
  $f = f^{F(g)}(s = \overrightarrow{Px})$;
  if s front-crosses f
    return ("A_f");
  elseif s back-crosses f
    return ("A_f'");

elseif $\exists t_f \in \bigcup_{\alpha=1}^{\beta} \mathcal{T}^{\alpha}$, s.t. $(f \in F(G) \cap F(g)) \wedge (x \in t_f)$

return ("B_{t_f}");

else

return ("B_u'") or ("O");

This procedure inserts a given frame into G.

procedure insert-one-frame(A, B, O, G)
  insertB(B, G);
  insertA(A, G);
  insertO(O, G);

This procedure inserts the "convexity" points, namely those that lie in a visibility tetrahedron. Function triangulate(B, f) builds a 2 1/2 valid triangulation of the set B augmented with the vertices of face f.

procedure insertB(B, G)
  foreach $B_{t_f} \in B$
    triangulate(Bt_f, f);

This procedure inserts the "concavity" points, namely those whose segments front-cross the graph.

procedure insertA(A, G)
  foreach $A_f \in A$
    triangulate(A_f, f);

This procedure builds disconnected subgraphs for the data points neither lie in a tetrahedral bundle nor cross any face of the graph.

procedure insertO(O, G)
  S =connected sets of O;
  foreach $s \in S$
    g_s = triangulate$(s, \emptyset)$;
  $G = G \cup_{s \in S} g_s$;

A.9 Modifying Algorithm merge-frames to Guarantee 3-Validity in the Two-Frame Case

A.9.1 Inserting the convexity points

As indicated in Section 4.2.4, the elements of B are inserted first into the graph. Each subset B_t is inserted in turn. The order in which the insertion occurs, however, is important. In this subsection, we define that order.

Definition 16 *The* **(i,j)-cell** *of $\mathcal{F}^\alpha$ is the region bounded by $U_i^\alpha, U_{i+1}^\alpha, V_j^\alpha, V_{j+1}^\alpha (0 < i < n, 0 < j < m)$. We call these regions* **proper cells**. *The proper cells are inside $\mathcal{T}^\alpha$. Each proper cell contains two tetrahedra of $\mathcal{T}^\alpha$. Additionally, the bounding sheets of $\mathcal{F}^\alpha$ define $n + m + 4$ infinite* **improper cells**. *An improper cell is (i,j)-cell such that $i = 0 \vee i = 0 \vee i = n \vee j = m$.*

Definition 17 *The* **d-neighbours** *of the (i,j)-cell are the (k,l)-cells such that $|k - i| + |l - j| = d$. Further, a point lying in a given (i,j)-cell c is said to be a d-neighbour of a point lying in a d-neighbour cell of c.*

Notation:

- For a given $B_t \in B_\alpha^\beta$, $d(B_t) = d$ if and only if the elements of B_t are d-neighbours of P^β.

The elements of B_α^β are inserted in such a way that $d = d(B_t)$ forms an increasing sequence. That is, we first insert the 0-neighbours of P^β (if $P^\beta \in \mathcal{T}^\alpha$), then we insert

the 1-neighbours of P^{β}, etc...

We call the modified algorithm `merge-frames-2()`. In the following subsection, we show the subroutines that are different from those used by `merge-frames()`.

A.9.2 The merge-frames-2 Incremental Merging Algorithm

This procedure implements the frame-merging for the second frame $\mathcal{F}^2$. In this algorithm, the elements of B are inserted into the graph in a certain order. This order guarantees that if one of the new segments s intersects one of the new faces f, f is *created* before we check for front-crossing of s with the graph faces.

procedure merge-one-frame(V, G)
  $A' =$ partitionA'(V, G);
  $V = V \setminus A'$;
  $B =$ partitionB(V, G);
  $A =$ insertB(B, G);
  $V = V \setminus B$;
  $A =$ partitionA(A, V, G);
  $O = V \setminus A$;
  insertA(A, G);
  insertO(O, G);
  return (A');

This procedure finds the backcrossing segments and returns them as a set.

procedure partitionA'(V, G)
  foreach $x \in V$
    foreach $f \in F(G)$
      if ($\overrightarrow{Px}$ back-crosses f)
        $A'_f = A'_f \cup \{x\}$
  $A' = \bigcup_{f \in F(G)}(A'_f)$;
  return (A');

This procedure finds the points that lie in the tetrahedral bundles.

procedure partitionB(V, G)
  foreach $x \in V$
    foreach $f \in F(G)$
      if $(\exists \alpha \in \{1, \ldots, (\beta - 1)\}), f \in F^{\alpha} \wedge x \in t_f$
        $B_{t_f} = B_{t_f} \cup \{x\}$;
  $B = \bigcup_f (B_{t_f})$;
  return (B);

This procedure inserts the points that lie in the tetrahedral bundles.

procedure insertB(B, G)
  foreach $B_{t_f} \in B$
    $d = d(B_{t_f})$;
  for$(d = 0; d <= d(B_{t_f})_{max}; d = d + 1)$
    foreach B_{t_f}, s.t. $d(B_{t_f}) = d$
      $I = \emptyset$;
      foreach $x \in B_f$
        foreach $\phi \in F(G)$
          if ($\overrightarrow{Px}$ crosses ϕ)
            $A_{\phi} = A_{\phi} \cup x$;
          else
            $I = I \cup x$;
      triangulate(I, f);
  $A = \bigcup_{f \in F(G)} (A_f)$;
  return (A);

This procedure finds the front-crossing segments. Function `firstcrossed`(s, C) returns the first face in the set C that s crosses.

procedure `partitionA`(A, V, G)
  foreach $x \in V$
    $C = \emptyset$;
    foreach $f \in F(G)$
      if ($s = \overrightarrow{Px}$ front-crosses f)
        $C = C \cup \{f\}$;
    if $C \neq \emptyset$
      $F =$ `firstcrossed`(s, C);
      $A_F = A_F \cup \{x\}$;
  $A = \bigcup_{f \in F(G)}(A_f)$;
  return (A);

A.10 Proof of Theorem 6

We wish to prove that, when only two frames are present, say $\mathcal{F}^1$ and $\mathcal{F}^2$, then algorithm `merge-frames-2` yields a graph which is:

- Either a triangulation or a set of triangulations.
- 3-consistent.

It is easy to see that the first statement is true. All data points but those in $A_1^{2\prime}$ are inserted. The points of both A_1^2 and of B_1^2 are triangulated within G^1. Finally, those in $B_1^{2\prime}$ or O_1^2 are inserted in the form of separate triangulations.

We now prove that the resulting graph is 3-consistent.

In the remainder of this paper, we shall say that a face f of the graph is I-consistent with respect to a given acquisition segment s if and only if s does not front-cross f in such a way that (4.10) is violated. By definition, 3-consistency with respect to a segment implies I-consistency with respect to that segment.

Similarly, we shall say that a face f of the graph is II-consistent with respect to a given acquisition segment s if and only if s does not terminate within a visibility tetrahedron t_f in such a way that (4.11) is violated. By definition, 3-consistency with respect to a segment implies II-consistency with respect to that segment.

We shall omit in the remainder the subscript and superscript from the names of the partition subsets. For example, we shall write A in place of A_1^2.

Lemma 3

$$\forall x \in V^2, \forall f \in F^1, (x \in B) \Longrightarrow (f \textit{ is I-consistent with } s = \overrightarrow{Px}).$$

Proof:

If $x \in B$, then s does not front-cross G^1. Therefore the premisse of (4.10) is false for all faces of G^1. Hence all faces of F^1 are I-consistent with respect to s. □

Lemma 4

$$\forall f \in F(G), \forall x \in V^1, f \textit{ is II-consistent with } s = \overrightarrow{P^1x}.$$

Proof:

If $x \in S^1$, the set of prior frames is empty for x. Hence any face of G is II-consistent with respect to the segments of S^1. □

The following proof follows the steps of the insertion procedure.

Lemma 5 *After insertion of B, the graph $G = G^{1,B}$ is 3-valid.*

Proof:

After insertion of B,

$$F(G) = F^1 \cup (\cup_{(B_f \neq \emptyset)} T_f) \setminus (\cup_{(B_f \neq \emptyset)} f), \tag{A.10}$$

where T_f is the 2 1/2-valid triangulation of B_f within f.

We shall prove that all the faces of $F(G)$ are 3-consistent with all the segments of S.

The faces of F^1 are obviously 3-consistent with the segments of S^1 since G^1 is valid. Further, since we are only considering the data points of B and their associated segments, the faces of F^1 are I-consistent with the segments of S^2 by Lemma 3.

In order for each such face f to also be be II-consistent with the segments of S^2, it is sufficient that each tetrahedron $t_f \in T^1$ contain no data point. If it is true, then we are done. If it is not, then B_f is not empty, and by (A.10), $f \notin F(G)$. So the faces of $F(G) \cap F^1$ are II-consistent with respect to all segments and therefore are 3-consistent with respect to all segments.

We now prove that the faces of T_f are 3-consistent.

The vertex set of each such triangulation T_f is $B_f \cup \{x_1, x_2, x_3\}$, where $\{x_1, x_2, x_3\}$ is the vertex set of f. Hence, T_f has the property that all its vertices lie inside the closure of the visibility tetrahedron t_f, and t_f is a convex figure. Therefore, the faces of T_f entirely lie inside t_f.

Now since F^1 is 2 1/2 valid, t_f is intersected by no segment of S^1. Hence, no face of T_f is intersected by a segment of S^1. Hence the faces of T_f are I-consistent with respect to the segments of S^1.

But by Lemma 4, the faces of T_f are II-consistent with respect to the segments of S^1.So the faces of T_f are 3-consistent with respect to the segments of S^1.

It remains to prove that the faces of T_f are 3-consistent with the segments of S^2. Some of the faces of T_f are made up of vertices belonging to V^2 only. By the 2 1/2-validity of G^2, those faces are 3-consistent.

The other faces of T_f are *mixed.* They are made of up of vertices of V^1 and of V^2. These faces are not faces of any visibility graph, and there exists no visibility tetrahedron associated with them. So they are necessarily II-consistent.

Lastly, we need to show that the mixed faces are I-consistent with respect to the segments of S^2. We need to prove that a given segment segment $s \in S^2$ front-crosses no mixed face.

We stated above that all faces of T_f, and hence all mixed faces, were entirely contained in the visibility tetrahedron t_f. So a given segment $s \in S^2$ can only intersect a mixed face inside t_f.

Let S be the sequence of visibility tetrahedra traversed by s, starting from P^2, and let S_n be the last such tetrahedron. The d-neighbour number with respect to P^2 increases monotonically along S. Hence, if s front-crosses a mixed face f inside a tetrahedron t other than S_n, the d-neighbour number of t is lower than that of S_n. This means that algorithm `merge-frames-2` has inserted f prior to verifying the faces front-crossed by s. Therefore, the datum point associated with s cannot be an element of B.

Finally, s cannot front-cross a mixed face within S_n. Since S_n is the last tetrahedron traversed by s, s terminates in S_n. Hence the data point associated with s is an element of B_{S_n}. But the triangulation built over the elements of B_{S_n} is by definition valid with respect to the segments of S^2.

□

Lemma 6 *After insertion of A, the graph* $G = G^{1,B,A}$ *is 3-valid.*

Proof: This proof follows the same lines as the proof of Lemma 5.

The set of faces of G becomes:

$$F(G) = F(G^{1,B}) \cup (\cup_{(A_f \neq \emptyset)} T_f) \setminus (\cup_{(A_f \neq \emptyset)} f), \tag{A.11}$$

where T_f is the 2 1/2-valid triangulation of A_f within f, and f is a face of $G^{1,B}$.

We first prove that the old faces, namely the faces in $F(G^{1,B}) \cap F(G)$, are 3-consistent with the new segments. In the last lemma, we saw these faces were 3-consistent with the old segments.

Let f be such a face. f cannot be the first face of the graph that s front-crosses, since if it were then x would be inserted into f, and by (A.11), f would not be a face of the graph.

I-consistency of f is then clearly true: x is inserted into the first face f' of the graph that s crosses. So if s also front-crosses f, then f' is the 3-cycle in G such that (4.10) is verified.

Likewise, II-consistency is also true for f. If $f \notin F^1$, then (4.11) is true by definition. So suppose $f \in (F^1 \cap F)$. Suppose further that $\exists x \in A, \text{s.t. } x \in t_f$. Since $x \in A$, s front-crosses at least one face f' of $G^{1,B}$. That face cannot be f, for if it was x would be outside of t_f. So the only way for x to be inside t_f is for s to back-cross f. Therefore $x \in A'$, which is false by assumption.

We now prove that the new faces are 3-consistent with respect to all segments.

Let f be such a new face, and let f' be the face of the current graph that s front-crosses. Then either f is made up of vertices of A only or f is made up of vertices of A and of f'. In both cases, f lies entirely to the interior of f', whereas P^2 lies to the exterior of f'.

Hence there exists a 2 1/2-consistent visibility graph for $A \cup_{f'} V(f')$ with respect to P^2, where $V(f')$ is the set of vertices of f'. Hence every new face is guaranteed to be 3-consistent with respect to the segments of S^2.

Finally, we need to prove that the new faces are 3-consistent with the segments of S^1.

We first prove that no segment $s \in S^1$ can front-cross f.

Suppose that there exists such a segment s. Then s must also front-cross f' since P^1 is to the exterior of f' and f' is to the exterior of f. But this is impossible since the current graph is 3-consistent. So f is I-consistent with respect to the segments of S^1.

Lastly, by Lemma (4), f is II-consistent with the segments of S^1. □

Lemma 7 *After insertion of O, the graph $G = G^{1,B,A,O}$ is 3-valid.*

Proof:

The elements of O are such that their segments cross none of the faces of the graph. Further, they lie in none of the tetrahedral bundles of the previous visibility graphs. So the faces of the current graph are 3-consistent with the triangulations induced on the connected components of O.

These triangulations in turn are constructed so that they are 2 1/2-consistent, hence 3-consistent with respect to the segments of S^2.

By the previous lemmas, therefore, G is 3-consistent □

Theorem 6 trivially follows, since $G = G^{1,A,B,O} = G^{1,2}$. □

Appendix B

Proofs of Chapter 5 Claims

B.1 Proof of Claim 1

We prove the claim that G_i is a three-connected maximal planar graph (3CMPG) by induction on the faces of G_{i-1}. We first note that by construction, G_0 is a 3CMPG. Suppose now that we have aggregated l elements of the partition $\mathcal{P}_{i-1}$ into a graph $G_i(l)$ such that

$$G_i(l) = G_{i-1} \bigcup \cup_{f=0}^{f=l} G_i^f \tag{B.1}$$

where for convenience the graph superscript indicates the face index rather than the face itself. We will show that if $G_i(l)$ is a 3CMPG, then the graph defined as

$$G_i(l+1) = G_i(l) \cup G_i^{l+1} \tag{B.2}$$

is also a 3CMPG, thus proving the claim. We stated that, by construction, the current iteration's subgraphs verify

$$\forall f = (x_1, x_2, x_3) \in F_{i-1}, G_i^f \text{ is a 3CMPG.}$$

Hence G_i^{l+1} is a 3CMPG. Also by construction,

$$V_i(l) \cap V_i^{l+1} = \{x_1, x_2, x_3\}, \tag{B.3}$$

$$E_i(l) \cap E_i^{l+1} = \Big\{(x_1, x_2), (x_2, x_3), (x_1, x_3)\Big\}, \tag{B.4}$$

$$F_i(l) \cap F_i^{l+1} = \Big\{(x_1, x_2, x_3)\Big\}. \tag{B.5}$$

It is well-known (Hartsfield and Ringel, 1990, Chapter 8) that a 3CMPG verifies the relationships

$$n - e + f = 2 \tag{B.6}$$

$$e = 3n - 6, \tag{B.7}$$

where n, e, and f are the number of vertices, edges, and faces of the 3CMPG respectively. The first expression expresses the Euler number relationship, while the second holds in the case of maximal planar graphs.

It immediately follows from (B.6,B.7) that

$$3f = 2e. \tag{B.8}$$

Now let n_1, e_1 and f_1 be the number of vertices, edges, and faces of $G_i(l)$, n_2, e_2 and f_2 be the number of vertices, edges, and faces of the subgraph G_i^{l+1}, and n_3, e_3 and f_3 be the number of vertices, edges, and faces of the union graph $G_i(l+1)$.

From (B.4), it follows that

$$e_3 = e_1 + e_2 - 3 \tag{B.9}$$

since the cardinality of the intersection graph edge set is 3, and from (B.5), that

$$f_3 = f_1 + f_2 - 2 \tag{B.10}$$

since the one common face of the component graphs does not belong to the union graph.

Hence

$$3f_3 = 3(f_1 + f_2) - 6 = 2(e_1 + e_2) - 6 = 2(e_1 + e_2 - 3) = 2e_3. \tag{B.11}$$

Hence $G_i(l+1)$ obeys the maximal planar graph face-edge relationship. It remains to show that $G_i(l+1)$ is 3-connected, namely that each of its vertices has at least 3 neighbors. But the union of any number of n-connected graphs is necessarily n-connected since the connectedness of each vertex of the union is at least as large as the connectedness of that vertex in each of its components.

Since, for each l, $G_i(l)$ is a 3CMPG, it follows that $G_i = G_i(f_1)$ is a 3CMPG. □

B.2 Proof of Claim 2

By definition, all the vertices of a given polyhedron $\mathcal{M}_i^f$ belong to $\overline{V_{i-1}}$ and hence are contained in $\mathcal{M}_{i-1}$, except for the vertices of f which are shared by both polyhedra. Hence $\mathcal{M}_i^f$ is contained in $\mathcal{M}_{i-1}$.

Furthermore, the acquisition segments associated with the vertices of $\mathcal{M}_i^f$ end their course inside $\mathcal{M}_i^f$, but by construction, these segments entirely lie to the outside of $\mathcal{M}_{i-1}$. So the inside of $\mathcal{M}_i^f$ lies outside of $\mathcal{M}_{i-1}$. Since this is true for all $\mathcal{M}_i^f$ polyhedra, we have:

$$\mathcal{M}_i = \mathcal{M}_{i-1} \setminus \bigcup^{f \in F_{i-1}} \mathcal{M}_i^f. \tag{B.12}$$

Equation (5.2) directly falls out if we write (B.12) for each level i, and then eliminate the $\mathcal{M}_i$'s for all values of i, save 0 and i_{max} (where $\mathcal{M} = \mathcal{M}_{i_{max}}$).

□

B.3 Proof of Claim 3

We are to show that the segments associated with the vertices of each subgraph do not intersect any face of that subgraph.

Suppose the claim is not true. Let s be the offending segment, v its associated vertex and ϕ the intersected face. Let $\mathcal{M}_i^f$ be the convex figure drawn by G_i^f.

We distinguish two cases, depending on whether v is one of the vertices of f (*i.e.* whether $v \in \{v_1, v_2, v_3\}$). Suppose first that it is false. In this case, because v belongs to p_i^f, s intersects face f, and v lies on the boundary of $\mathcal{M}_i^f$. Since v terminates s, and since f is a face of $\mathcal{M}_i^f$ (but not of G_i^f), it follows that s has two distinct intersections with $\mathcal{M}_i^f$, one of which is non-strict. But if s has a third intersection with $\mathcal{M}_i^f$ at face ϕ, $\mathcal{M}_i^f$ is not convex, which violates the assumption.

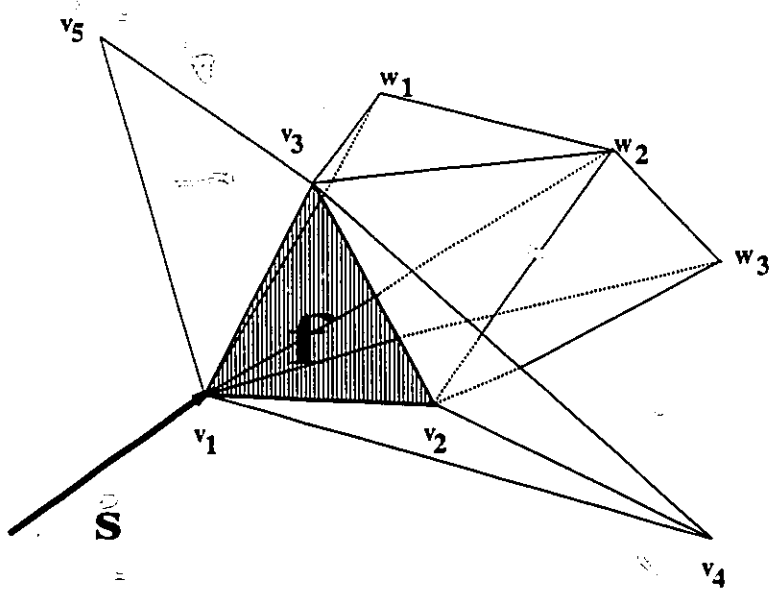

Figure B.1 $\mathcal{M}_i^f$ and s lie on opposite sides of face f.

Suppose now $v \in \{v_1, v_2, v_3\}$. In this case, s cannot intersect any face $\mathcal{M}_i^f$ because s and $\mathcal{M}_i^f$ entirely lie on opposite sides of face f (See Figure B.1). To prove the statement, note that f belongs to the convex figure $\mathcal{M}_i^f$. Hence $\mathcal{M}_i^f$ entirely lies to one side of f, say the "beyond" side. For each element p of p^f, however, $s(p)$ intersects first f', where f' is the face that f is rooted at, then it intersects f and terminates at p. Since p belongs to $\mathcal{M}_i^f$, it lies on the beyond side of f, and hence f' lies on near side of f. But s also intersects f' (remember that s must be an element of $p^{f'}$), and

lies on one side of f only since it terminates at a vertex of f. Therefore s lies on the near side of f and the proof is complete in the case of a regular iteration i. If $i = 0$ though, f' does not exist. But in this case $\mathcal{M}_0^f = f$, and the claim is true by virtue of Proposition 1. □

B.4 Proof of Claim 4

We are to show that the convex layer algorithm always terminates.

The termination condition is

$$\exists n, p^f(n) = \emptyset. \tag{B.13}$$

Hence it suffices to show that

$$\forall i, p^f(i+1) \text{ is a proper subset of } p^f(i).$$

The associated segment of each element of p^f, and hence of each element of p^f, intersects f. Hence these elements all lie on the same side of f. Therefore, $\forall i$, the vertices of f are extremes of $V^f(i)$ and f belongs to the set of faces of $G^f(i)$. Hence the vertices of f never have degree zero in $G^f(i)$. We then have

$$\forall i, p^f(i+1) \bigcap \{v_0, v_1, v_2\} = \emptyset.$$

Since we also have by construction

$$p^f(i) \bigcup \{v_0, v_1, v_2\} = V^f(i),$$
$$p^f(i+1) \subseteq V^f(i),$$

a simple Venn diagram shows that

$$p^f(i+1) \subseteq p^f(i).$$

It remains to show that $p^f(i+1) \neq p^f(i)$. Suppose both sets are equal. Then all points of $p^f(i)$ are internal to $G^f(i+1)$. Therefore, the only points with non-zero degree in $G^f(i+1)$ are v_0, v_1, and v_2. But since the elements of $p^f(i)$ are not coplanar with f (they strictly intersect f) this implies that $p^f(i)$ is empty. But if $p^f(i)$ is empty, $p^f(i+1)$ is not constructed by virtue of the stopping criterion. So the situation does not occur. □

B.5 The Global Face-Merging Algorithm

The main procedure. It first constructs the hull of all data points. Then it builds new graph iterations as long as the graph does not span all data points,

procedure `main` (V, S)
  $G =$ `convex-hull` (V);
  while $(V \neq \emptyset)$
    `iterate-graph` (G);
    $V =$ internal points of (G);
  return G;

This procedure partitions the internal points according to which face intersected by their segment. Each equivalence class forms a concavity. C is the set of concavities.

procedure `iterate-graph` (G)
  $C = \emptyset$;
  foreach $f \in F(G)$
    $p(f) = \left\{v | (v \in V(G)) \wedge (s(v) \cap f \neq \emptyset)\right\}$;
    $V(f) = p(f) \cup$ vertices of f;
    $C = C \cup \{V(f)\}$;
  `merge-faces` (C);

This procedure merges concavities when certain criteria are met. It first builds the maximal-depth convex hull CL for each concavity. Then it selects a set P of concavity pairs which are candidates for merging. If the respective hulls of the candidates intersect, all the concavities that lie on the shortest path between the candidates are merged into a single concavity. The process continues until no intersection is found.

procedure `merge-faces`(C)
  do
    $Intersection = False$;
    foreach $c \in C$
      $CL(c)$ =`convex-layer`(c);
    P =`build-concavity-pair-test-set` (C);
    foreach $p = (c_1, c_2) \in P$
      if $\left(CL(c_1) \cap CL(c_2)\right) \neq \emptyset$
        c_1 =`shortest-path`(c_1, c_2);
        $c_2 = \emptyset$;
        $C = C \setminus c_2$;
        $Intersection = True$;
  while $(Intersection == True)$;

This procedure computes the constrained maximal-depth convex layer of the concavity. The constraint is that the concavity faces belong to each convex layer.

procedure `convex-layer`(c)
  do
    CH =`convex-hull` (c);
    V = (internal points of CH) $\bigcup$ (face vertices);
  while $(V \neq \emptyset)$;
  return CH;

This procedure constructs the set of candidate pairs for the concavity merging test. *sp* is the shortest-path between a pair, measured by the number of separating concavities in the current graph. If *sp* is short, *and* only contains concavities which are not intersected by segments (except for the pair elements), *and* contains no cycle, then the pair is a candidate.

procedure `build-concavity-pair-test-set` (C)

$P = \emptyset$;

foreach $(c_1, c_2) \in (C \times C)$

sp =`shortest-path`(c_1, c_2);

if $\Big($`dist`$(sp) < D\Big) \wedge \Big($`not-cycle`$(sp)\Big) \wedge \Big(\nexists c \in sp \mid (p(f))(c) \neq \emptyset\Big)$

$P = P \cup (c_1, c_2)$;

return P;

Bibliography

Acampora, A. and Winters, J. (1989). Three-dimensional ultrasonic vision for robotic applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(3):291–303.

Aggarwal, J. and Magee, M. (1986). Determining motion parameters using intensity-guided range sensing. *Pattern Recognition*, 19(2):169–180.

Agin, G. and Binford, T. (1976). Computer description of curved objects. *IEEE Transactions on Computers*, 25:439–449.

Aleksandrov, P. (1956). *Combinatorial Topology*, volume 1. Graylock Press, Rochester, NY.

Alevizos, P., Boissonnat, J.-D., and Yvinec, M. (1987). An optimal $O(n \log n)$ algorithm for contour reconstruction from rays. In *ACM Annual Symposium on Computational Geometry.*

Allen, P. and Michelman, P. (1990). Acquisition and interpretation of 3-d sensor data from touch. *IEEE Trans. on Robotics and Automation*, 6(4):397–404.

Altschuler, M., Altschuler, B., and Taboada, J. (1981). Laser electro-optic system for rapid three-dimensional (3-d) topographic mapping of surfaces. *Optical Engineering*, pages 953–961.

Audenaert, K., Peremans, H., Kawahara, Y., and Campenhout, J. V. (1992). Accurate ranging of multiple objects using ultrasonic sensors. In *IEEE Int. Conf. on Robotics & Automation*, pages 1733–1738, Nice, France.

Baker, H. and Binford, T. (1981). Depth from edge and intensity based stereo. In *Int. Joint Conf. on Artificial Intelligence*, pages 631–636.

Ballard, D. and Brown, C. (1982). *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ.

Beckwith, T., Buck, N., and Marangoni, R. (1981). *mechanical measurements*. Addison-Wesley, Reading, MA.

Bender, E. and Wormald, N. (1985). Almost all convex polyhedra are asymmetric. *Canadian Journal of Mathematics*, 27(5):854–871.

Besl, P. (1988). Active, optical range imaging sensors. *Machine Vision and Applications*, 1:127–152.

Besl, P. and Jain, R. (1985). Range image understanding. In *Int. Conf. on Computer Vision & Pattern Recognition*, pages 430–449.

Besl, P. J. and Jain, R. C. (1988). Segmentation through variable-order surface fitting. *IEEE Transactions on Pattern Anal. and Machine Intell.*, 10(2):167–192.

Bhanu, B. (1984). Representation and shape matching of 3-D objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6(3):340–351.

Bickel, G., Häusler, G., and Maul, M. (1985). Triangulation with expanded range of depth. *Optical Engineering*, 24:975–977.

Boissonnat, J. (1982). Representation of objects by triangulating points in 3-D space. In *Int. Conf. on Pattern Recognition*, pages 830–832, Munich, Germany.

Boissonnat, J. (1984). Representing 2D and 3D shapes with the delaunay triangulation. In *Int. Conf. on Computer Vision & Pattern Recognition*, pages 745–748.

Boissonnat, J. (1988). Shape reconstruction from planar cross-sections. *Computer Graphics & Image Processing.*

Boissonnat, J., Faugeras, O., and Bras-Mehlman, E. L. (1988). Representing stereo data with the delaunay triangulation. In *IEEE Int. Conf. on Robotics & Automation*, pages 1798–1803, Philadelphia, PA.

Bondy, J. A. and Murty, U. (1976). *Graph theory with applications.* North-Holland, Amsterdam, Netherlands.

Borenstein, J. and Koren, Y. (1988). Obstacle avoidance with ultrasonic sensors. *IEEE Trans. on Robotics and Automation*, 4:213–218.

Boyer, K. and Kak, A. (1987). Color-encoded structured light for rapid active ranging. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(1):14–28.

Breton, P., Iverson, L., Langer, M., and Zucker, S. (1992). Shading flows and scenel bundles: A new approach to shape from shading. In *Computer Vision - ECCV 92*, pages 135–150, Santa Margherita Ligure. Italy. Springer Verlag.

Brown, W. (1964). Enumerations of triangulations of the disk. *Proceedings of the London Mathematical Society*, 14(3):746–768.

Carrihill, B. and Hummel, R. (1985). Experiments with the intensity ratio depth sensor. *Computer Vision, Graphics & Image Processing*, 32:337–358.

Chazelle, B. (1983). Optimal algorithms for computing depths and layers. *Proc. 21st Allerton Conference on Comm., Control, and Computers*, pages 427–436.

Chen, Y. and Medioni, G. (1991). Object modeling by registration of multiple range images. In *IEEE Int. Conf. on Robotics & Automation*, pages 2724–2729.

Chien, C. and Aggarwal, J. (1986). Volume/surface octrees for the representation of three-dimensional objects. *Computer Vision, Graphics & Image Processing*, 36:100–113.

Choi, Y., Weiss, L., Gursoz, E. L., and Prinz, F. (1990). Rapid prototyping from 3D scanned data through automatic surface and solid generation. In Sharon, A., editor, *DE-Vol. 29, Issues in Design/Manufacture Integration*, pages 17–23. ASME. Book No. G00542.

Connolly, C. (1984). Cumulative generation of octree models from range data. In *IEEE Int. Conf. on Robotics & Automation*, pages 25–32, Atlanta, GA.

Dane, C. (1982). *An object-centered three-dimensional model builder*. PhD thesis, University of Pennsylvania, Philadelphia, PA. CIS Dept., Moore School of Electrical Engineering.

De Schutter, J. and Brussel, H. V. (1988). A three-dimensional assembly task quantification with apllication to machine dexterity. *Int. Journal of Robotics Research*, 7(4):18–33.

Dobkin, D. and Kirkpatrick, D. (1983). Fast detection of polyhedral intersection. *Theoretical Computer Science*, 27:241–253.

Durrant-Whyte, H. (1987). Consistent integration and propagation of disparate sensor observations. *Int. Journal of Robotics Research*, 6(3):3–24.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, 29:551–559.

Fan, T.-J., Medioni, G., and Neviata, R. (1989). Recognizing 3-D objects using surface descriptions. *IEEE Trans. on Pattern Analysis and Machine Intelligence.*

Faugeras, O., Hebert, M., Mussi, P., and Boissonnat, J. (1984). Polyhedral approximation of 3-d objects without holes. *Computer Graphics & Image Processing*, 25:169–183.

Faugeras, O. and Pauchon, E. (1983). Measuring the shape of 3D objects. In *Int. Conf. on Pattern Recognition & Image Processing*, pages 2–7.

Faugeras, O. and Ponce, J. (1983). Prism trees: A hierarchical representation for 3-D objects. In *Int. Joint Conf. on Artificial Intelligence*, pages 982–988.

Faux, I. and Pratt, M. (1979). *Computational Geometry for Design and Manufacture.* Ellis Horwood, Chichester, U.K.

Fedèrico, P. (1975). The number of polyhedra. *Philips Research Report*, 30:220*–231*. Issue in honour of C.J. Bouwkamp.

Ferrie, F. (1986). *Reconstructing and interpreting the 3D shape of moving objects.* PhD thesis, McGill University, Montréal, Canada.

Ferrie, F., Lagarde, J., and Whaite, P. (1990). Recovery of volumetric object description from laser rangefinder images. In *First European Conference on Computer Vision*, pages 387–396, Antibes, France. Springer-Verlag. Lecture Notes in Computer Science, 427.

Fuchs, H., Kedem, Z., and Uselton, S. (1977). Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20:693–702.

Gil, B., Mitiche, A., and Aggarwal, J. (1983). Experiments in combining intensity and range edge maps. *Computer Vision, Graphics & Image Processing*, 21:395–411.

Goldberg, N. (1982). Inside autofocus: How the magic works. *Popular Photography*, pages 77–83.

Gonzalez, R. and Wintz, P. (1977). *Digital Image Processing.* Addison-Wesley, Reading, MA.

Grűnbaum, B. (1967). *Convex Polytopes*, volume XVI of *Pure and Applied Mathematics.* Interscience Publishers, London.

Harary, F. (1972). *Graphical Theory.* Addison-Wesley, Reading, MA.

Harary, F. and Palmer, E. (1973). *Graphical Enumeration.* Academic Press, New York.

Harary, F. and Tutte, W. (1966). On the order of the group of a planar map. *Journal of Combinatorial Theory*, 1:394–395.

Harding, K. and Goodson, K. (1986). Hybrid, high-accuracy structured light profiler. In *SPIE Conf. on Optics, Illumination, and Image Sensing for Machine Vision*, pages 132–145. Vol. 728.

Hart, P., Nilsson, N., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems, Man, and Cybernetics*, pages 100–107.

Hartsfield, N. and Ringel, G. (1990). *Pearls in Graph Theory.* Academic Press, San Diego, CA.

Hasegawa, T. (1982). An interactive system for modeling and monitoring a manipulation environment. *IEEE Trans. on Systems, Man, and Cybernetics.*

Hayward, V. (1986). Fast collision detection scheme by recursive decomposition of a manipulator workspace. In *IEEE Int. Conf. on Robotics & Automation*, pages 1044–1049, San Francisco.

Hayward, V. and Aubry, S. (1987). On describing a robotic scene. In *SPIE Conference on Intelligent Robots and Computer Vision: Sixth in a Series*, Cambridge, MA.

Herman, M. (1985). Matching three-dimensional symbolic descriptions obtained from multiple views of a scene. In *Int. Conf. on Computer Vision & Pattern Recognition*, pages 585–590.

Hong, T. and Shneier, M. O. (1985). Describing a robot's workspace using a sequence of views from a moving camera. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 7(6):721–727.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1991). Surface reconstruction from unorganized points. Technical Report 91–12–03, University of Washington, Seattle, WA. Department of Computer Science and Engineering.

Horn, B. (1968). Focussing. AI Memo 160, MIT, Cambridge, MA. Project MAC.

Horn, B. (1975). Obtaining shape from shading information. In Winston, P., editor, *The psychology of computer vision*. McGraw-Hill.

Hummel, R. and Zucker, S. (1983). On the foundations of relaxation labelling processes. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:267–287.

Idesawa, M. and Yatagai, T. (1980). 3-d shape input and processing by moiré technique. In *Int. Conf. on Pattern Recognition*, pages 1085–1090.

Ikeuchi, K. and Horn, B. (1981). Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184.

Ishii, M. and Nagata, N. (1976). Feature extraction of 3-d objects and visual processing in a hand-eye system using laser tracker. *Pattern Recognition*, 8:229–237.

Jain, R. and Jain, A. (1989). Report on range image understanding workshop, east lansing, michigan, march 21–23, 1988. *Machine Vision and Applications*, 2(1):45–60.

Jarvis, R. (1976). Focus optimisation criteria for computer image processing. *Microscope*, 24:163–180.

Jarvis, R. (1983). A perspective on range finding techniques for computer vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5(2):122–139.

Kender, J. (1978). Shape from texture: a brief overview and a new aggregation transform. In *DARPA IU Workshop*, pages 79–84.

Keppel, E. (1975). Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research & Development*, 19:2–11.

Kim, Y. and Aggarwal, J. (1986). Rectangular parellelepiped coding: A volumetric representation of three-dimensional objects. *IEEE Trans. on Robotics and Automation*, 2(3):127–134.

Krotkov, E. and Kories, R. (1988). Adaptive control of cooperating sensors: Focus and stereo ranging with an agile camera system. In *IEEE Int. Conf. on Robotics & Automation*, pages 548–553, Philadelphia.

Krotkov, E. and Martin, J. (1986). Range from focus. In *IEEE Int. Conf. on Robotics & Automation*, pages 1093–1098, San Francisco.

Latombe, J. (1991). *Robot Motion Planning.* Kluwer, Boston, MA.

Leclerc, Y. (1989). *The local structure of image intensity discontinuities.* PhD thesis, McGill University, Montréal, Québec, Canada.

Levine, M. (1985). *Vision in man and machine.* McGraw-Hill, New York, NY.

Levine, M., O'Handley, D., and Yagi, G. (1973). Computer determination of depth maps. *Computer Graphics & Image Processing*, 2:134–150.

Lewis, R. and Johnston, A. (1977). A scanning laser range finder for a robotic vehicle. In *Int. Joint Conf. on Artificial Intelligence*, pages 762–768.

Lowe, D. (1987). Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395.

Lozano-Perez, T., Mason, M., and Taylor, R. (1984). Automatic synthesis of fine-motion strategies for robots. *Int. Journal of Robotics Research*, 3(1):3–24.

Lozano-Pérez, T. and Wesley, M. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22:560–570.

Markowski, G. and Wesley, M. (1980). Fleshing out wire frames. *IBM Journal of Research & Development*, 24:582–597.

Marr, D. (1982). *Vision.* W.H. Freeman, New York, NY.

Marr, D. and Poggio, T. (1979). A computational theory of human stereo vision. *Proceedings of the Royal Society of London B*, 204:301–328.

Martin, M. and Aggarwal, J. (1983). Volumetric description of objects from multiple views. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:150–159.

Mason, M. (1984). Automatic planning of fine-motions: Correctness and completeness. In *IEEE Int. Conf. on Robotics & Automation*, pages 492–503, Atlanta, GA.

Meagher, D. (1982a). Efficient synthetic image generation of arbitrary 3-D objects. In *prip*, pages 473–478.

Meagher, D. (1982b). Geometric modelling using octree encoding. *Computer Graphics & Image Processing*, 19:129–147.

Menq, C., Yau, H., and Lai, G. (1992). Automated precision measurement of surface profile in cad-directed inspection. *IEEE Trans. on Robotics and Automation*, 8(2):268–278.

Moravec, H. (1979). Visual mapping by a robot rover. In *Int. Joint Conf. on Artificial Intelligence*, pages 598–620.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE Int. Conf. on Robotics & Automation*, pages 116–121.

Moring, I., Ailisto, H., Heikkinen, T., A.Kilpelä, Myllylä, R., and Pietikäinen, M. (1987). Acquisition and processing of range data using a laser scanner-based 3-d vision system. In *SPIE– Optics, Illumination, and Image Sensing for Machine Vision*, pages 174–183. Vol. 850.

Moutarlier, P. and Chatila, R. (1989). Incremental environment modelling by a mobile robot from noisy data. In Hayward, V. and Khatib, O., editors, *Experimental Robotics I. The First International Symposium*, pages 327–346. Springer-Verlag, Montréal, PQ.

Mulgaonkar, P., Shapiro, L., and Haralick, R. (1982). Recognising three-dimensional objects in single perspective views using geometric and relational reasoning. In *Int. Conf. on Pattern Recognition & Image Processing*, pages 479–484.

Nevatia, R. (1982). *Machine Perception.* Prentice-Hall, Englewood Cliffs, NJ.

Nitzan, D., Brain, A., and R.Duda (1981). The measurement and use of registered reflectance and range data in scene analysis. *Proceedings of the IEEE*, 69(5):572–595.

Noborio, H., Fukuda, S., and Arimoto, S. (1988). Construction of the octree approximating three-dimensional objects by using multiple views. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(6):769–781.

Okada, T. (1982). Development of an optical distance sensor for robots. *Int. Journal of Robotics Research*, 1(4):3–14.

O'Rourke, J. (1981). Polyhedra of minimal area as 3D object models. In *Int. Joint Conf. on Artificial Intelligence*, pages 664–666.

O'Rourke, J. (1987). *Art Gallery Theorems and Algorithms.* Oxford University Press, New York.

O'Rourke, J. and Badler, N. (1979). Decomposition of three-dimensional objects into spheres. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 295–305.

O'Rourke, J., Booth, H., and Washington, R. (1987). Connect-the-dots: A new heuristic. *Computer Graphics & Image Processing*, 39:258–266.

Oshima, M. and Shirai, Y. (1983). Object recognition using three-dimensional information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 5:353–361.

Ozeki, O., Nakano, T., and Yamamoto, S. (1986). Real-time range measurement device for three-dimensional object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 550–554.

Parvin, B. and Medioni, G. (1991). A layered network for the correspondence of 3D objects. In *IEEE Int. Conf. on Robotics & Automation*, pages 1808–1813.

Pentland, A. (1986). Local shading analysis. In Pentland, A., editor, *From pixels to predicates*, pages 40–77. Ablex Publishing, Norwood, NJ.

Pogorelov, A. (1965). *Differential Geometry.* Noordhof, Groningen, Netherlands.

Potmesil, M. (1979). Generation of 3D surface descriptions from images of pattern-illuminated objects. In *Int. Conf. on Pattern Recognition & Image Processing*, pages 553–559.

Potmesil, M. (1983). Generating models of solid objects by matching 3D surface segments. In *Int. Joint Conf. on Artificial Intelligence*, pages 1089–1093.

Potmesil, M. (1987). Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics & Image Processing*, pages 1–29.

Pratt, W. (1978). *Digital Image Processing.* Wiley-Interscience, New York, NY.

Preparata, F. (1985). *Computational Geometry: An Introduction.* Springer-Verlag, New York, NY.

Raviv, D., Pao, Y.-H., and Loparo, K. (1989). Reconstruction of three-dimensional surfaces from two-dimensional binary images. *IEEE Trans. on Robotics and Automation*, 5(5):701–710.

Reid, G. (1986). Automatic fringe pattern analysis: A review. *Optics and Lasers in Engineering*, 7:37–68.

Requicha, A. (1980). Representations of rigid solids: theory, methods and systems. *ACM Computing Surveys*, 12(4):437–464.

Requicha, A. (1983). Solid modelling: Current status and research directions. *IEEE Computer Graphics and Applications*, pages 25–37.

Rioux, M. (1984). Laser range finder based on synchronized scanners. *Applied Optics*, 23:3837–3844.

Rioux, M. and Blais, F. (1986). Compact three-dimensional camera for robotic applications. *Journal of the Optical Society of America A*, 3:1518–1521.

Rosenberg, D., Levine, M., and Zucker, S. (1978). Computing relative depth relationships from occlusion cues. In *Int. Joint Conf. on Artificial Intelligence*, pages 765–769, Kyoto, Japan.

Samet, H. (1984). The quadtree and related hierarchical data structures,. *ACM Computing Surveys*, 16(2).

Sato, K. and Inokuchi, S. (1985). Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems*, 2(1):27–39.

Sato, Y., Kitagawa, H., and Fujita, H. (1982). Shape measurement of curved objects using multiple slit-ray projections. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 4(6):641–646.

Skifstad, K. and Jain, R. (1989). Range estimation from intensity gradient analysis. *Machine Vision and Applications*, 2:81–102.

Smith, G. (1983). Shape from shading: an assessment. Technical Note 287, SRI, Menlo Park, CA.

Soneira, R. (1988). Three-dimensional imaging and scanning range finders using the parallax principle. In *IEEE Int. Conf. on Robotics & Automation*, pages 1715–1720, Philadelphia, PA.

Soucy, M. (1993). *Modélisation de la surface d'objects 3-D à l'aide de plusieurs images de profondeur.* PhD thesis, Université Laval.

Srinivasan, P., Liang, P., and Hackwood, S. (1989). Computational geometric methods in volumetric intersection for 3D reconstruction. In *IEEE Int. Conf. on Robotics & Automation*, pages 190–195.

Srivastava, S. and Ahuja, N. (1990). Octree generation from object silhouettes in perspective views. *Computer Vision, Graphics & Image Processing*, 49:68–84.

Stenstrom, J. and Connolly, C. (1986). Building wire frames from multiple range views. In *IEEE Int. Conf. on Robotics & Automation*, pages 615–620.

Stevens, K. (1979). Representing and analyzing surface orientation. In Winston, P. and Brown, R., editors, *Artificial Intelligence: An MIT Perspective*. MIT Press.

Svetkoff, D., Leonard, P., Sampson, R., and Jain, R. (1984). Techniques for real-time feature extraction using range information. In *SPIE–Intelligent Robotics and Computer Vision*, pages 302–309.

Tajima, J. (1987). Rainbow range finder principle for range data acquisition. In *IEEE Workshop Industrial Applications of Machine Vision and Machine Intelligence*, pages 381–386.

Tanimoto, S. and Pavlidis, T. (1975). A hierarchical data structure for picture processing. *Computer Graphics & Image Processing*, 4(2):104–119.

Terzopoulos, D. (1988). The computation of visible-surface representations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):417–438.

Tozer, B., Glanville, R., Gordon, A., Little, M., Webster, J., and Wright, D. (1985). Holography applied to inspection and measurement in an industrial environment. *Optical Engineering*, 24(5):746–753.

Tutte, W. (1962a). A census of planar triangulations. *Canadian Journal of Mathematics*, 14:21–38.

Tutte, W. (1962b). A census of slicings. *Canadian Journal of Mathematics*, 14:708–722.

Tutte, W. (1963). A census of planar maps. *Canadian Journal of Mathematics*, 15:249–271.

Udupa, K. and Murthy, I. (1977). New concepts for three-dimensional analysis. *IEEE Transactions on Computers*, 26(10):1043–1049.

Ullman, S. (1979). *The interpretation of visual motion.* MIT Press, Cambridge, MA.

Veenstra, J. and Ahuja, N. (1986). Efficient octree generation from silhouettes. In *Int. Conf. on Computer Vision & Pattern Recognition*, pages 537–542, Miami, FL.

Vuylsteke, P. and Oosterlinck, A. (1990). Range image acquisition with a single binary-encoded light pattern. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(2):148–164.

Waltz, D. (1975). Understanding line drawings of scenes with shadows. In Winston, P., editor, *The psychology of computer vision*, pages 19–91. McGraw-Hill, New York, NY.

Wang, Y. and Aggarwal, J. (1986). Surface reconstruction and representation of 3-d scenes. *Pattern Recognition*, 19(3):197–206.

Wang, Y. and Aggarwal, J. (1989). Integration of active and passive sensing techniques for representing three-dimensional objects. *IEEE Trans. on Robotics and Automation*, 5(4):460–471.

Whitesides, S. (1985). Computational geometry and motion planning. In Toussaint, G., editor, *Computational geometry*, pages 377–427. North-Holland, Amsterdam, Holland.

Will, P. and Pennington, K. (1972). Grid coding: a novel technique for image processing. *Proceedings of the IEEE*, 60(6):669–680.

Witkin, A. (1980). *Shape from contour*. PhD thesis, MIT, Cambridge, MA.

Yakimovski, Y. and Cunningham, R. (1978). A system for extracting three-dimensional measurements from a stereo pair of tv cameras. *Computer Graphics & Image Processing*, 7:195–210.