This is to certify that the

dissertation entitled

CAD-BASED ROBOT MOTION PLANNING
FOR INSPECTION IN MANUFACTURING

presented by

WEIHUA SHENG

has been accepted towards fulfillment
of the requirements for

PH.D degree in ELECTRICAL & COMPUTER
ENGINEERING

Major professor

Date 05/08/02

CAD-BASED ROBOT MOTION PLANNING FOR INSPECTION IN MANUFACTURING

BY

WEIHUA SHENG

A DISSERTATION

Submitted to

Michigan State University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

2002

ABSTRACT

CAD-based Robot Motion Planning for Inspection in Manufacturing

By

Weihua Sheng

The last several decades have witnessed the rapid development of CAD/CAM technology and its extensive applications in the manufacturing industry. Although CAD and CAM technology have been fully developed individually, the connection between them, or the manufacturing process planning, has not achieved equal success. In recent years, planning manufacturing processes based on the CAD information has been receiving more and more attention.

In this dissertation, a general, CAD-based framework is developed for constraint-satisfying robot motion planning in automotive manufacturing. Particularly, motivated by achieving fully-automated, fast dimensional inspection for automotive parts, this dissertation focuses on the CAD-based inspection planning, which, essentially, is the automatic planning of the inspection sensor, or camera.

First, a CAD-based vision sensor planning approach is developed, which utilizes the geometric information of the part and the knowledge about the sensor model to generate constraint-satisfying sensor configurations. By using a new concept called bounding box, the proposed approach combines the advantages of two existing vision sensor planning approaches. Second, to improve the efficiency and the kinematics performance of the inspection system, two optimization problems are formulated: the minimum viewpoint problem and the optimal kinematics performance problem.

Based on a discretization scheme, both are converted into integer optimization problems. Third, the robot path planning problem is studied. It is formulated as a Traveling Salesman Problem (TSP) and a hierarchical algorithm is developed to obtain fast, approximate solutions. Finally, the implementation of an eye-in-hand inspection system is discussed and a new, simple hand/eye calibration method is developed. We believe that the general framework and the concepts proposed in this dissertation can benefit many similar robotic applications in manufacturing.

For my mother, father, and sister,

their love makes dream true.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Ning Xi, for his insightful guidance, continuous support and encouragement through my Ph.D study. Without his help, this dissertation would not have been finished. I deeply appreciate those weekend discussions with him. Also I would like to thank Dr. Mumin Song from Ford Motor Company, for his great efforts in initiating and promoting this project, and Dr.Yifan Chen, also from Ford Motor Company, for his many valuable suggestions, which shaped some fundamental approaches developed in this dissertation. Thanks should also go to other Ford researchers, especially Perry MacNeille and James Rankin III, for their support and suggestions. Dr.Wei Kang from Naval Graduate School also gave me help on references and literature. The discussion with him was very helpful to solving the viewpoint optimization problem. Many colleagues in the Robotics and Automation Laboratory gave me great help. The discussions with them were beneficial and full of joy as well. Among them, Jindong Tan helped me to implement some of the experiments. Heping Chen gave me good suggestions on extending my work to other applications. Other people who I should give thanks to are: Imad Elhajj, Amit Goradia, Jizhong Xiao, Yu Sun, Guangyong Li and Qi Zhu.

This research is supported by the Scientific Research Laboratories of Ford Motor Company and the National Science Foundation.

Finally, I would like to thank my dear parents for their enduring encouragement and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

In the last few decades, a large number of new technologies have been developed for the manufacturing industry. Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) technology are two of them. Both technologies are products of the computer era. Since their inceptions, CAD systems have evolved from the early graphic editors with some built-in design symbols to the current multi-functional, systematic software packages. Basically, in CAD systems, geometric modeling is used to mathematically represent an object. An object model usually consists of geometry, topology and auxiliary information [4]. In CAM systems, computers are utilized to automate a set of operations and activities which include material selection, planning, production and inspection, etc. [4].

Although CAD and CAM technology have been fully developed individually, a big gap still exists between them, which hinders the integration of CAD and CAM, or the full automation of CAD/CAM system. As a fact, human intervention is still indispensable in the planning of many manufacturing processes [5]. In this sense, people describe CAD and CAM as "islands of automation" [6]. To fully realize the benefits of CAD and CAM, computer support for process planning, or namely, Computer Aided Process Planning (CAPP) should be available to more conventional CAD and CAM processes [7]. However, as Requicha *et al.* found out, CAD/CAM applications that involve synthesis or planning are especially difficult [8]. Essentially, process planning, or the activity of devising means to achieve desired goals under given constraints and with limited resources, is an instinctive part of intelligent behavior, often performed subconsciously by human beings [5].

One big obstacle lies in the fact that CAD systems and CAM systems use dif-

ferent part representations. CAD systems provide a description of the part in low level, simple geometric entities (faces, edges and vertices), while in the manufacturing environment, high level descriptions of entities are desired to plan manufacturing processes [9]. Hence, Computer Aided Process Planning (CAPP) systems usually consist of two critical steps: the automatic understanding of the part shape and the automatic generation of manufacturing plans based on this understanding [9].

Inspection planning is an important type of manufacturing process planning. Quality and process control activities in a mechanical product's life cycle require that parts be measured, or dimensionally inspected [10]. Inspection generally involves a time-consuming operation that takes up a large portion of manufacturing lead time and therefore, has been creating serious bottlenecks in production lines [11]. For example, Coordinate Measuring Machine (CMM), an extensively used equipment for dimensional measurements in the manufacturing industry [10], uses touch probes to measure the coordinates of specified points on a workpiece's surfaces. CMMs can achieve very high precision compared to other measuring equipment. However, CMMs use a point-by-point scheme, thus the measuring processes are very time-consuming. For example, to inspect a typical automotive part, it usually takes several days.

In recent years, many active optical inspection techniques have been developed which greatly reduces the time in dimensional inspection. In fact, most of these noncontact-type methods are usually 50-100 times faster than the CMM method [11]. Among the current active optical sensing methods, structured light, which obtains $3D$ coordinates by projecting specific light patterns on the surfaces of the measured object, is one of the active methods that has been successfully employed in various applications [12].

However, to achieve full automation, as well as to improve the efficiency of the inspection system, sensor planning, or finding the suitable configurations of sensors so that the inspection task can be carried out satisfactorily, is a very important link that

should be paid attention to. In the structured light method, camera configurations, such as position, orientation and optical settings are critical parameters that affect measuring accuracy and efficiency directly. In most of the current structured light applications, the camera configuration (or viewpoint) planning was based on human experience, thus the measurements suffer considerable human errors and low efficiency. It is, therefore, highly desirable to develop a camera positioning system that is able to plan and realize the camera configurations in a fully-automated, accurate and efficient way. The above goal motivates the work in this dissertation.

To begin with, in the next section, related research work is reviewed.

1.2 Literature Review

First, we briefly survey the current status of the research in Computer Aided Process Planning (CAPP). Then the previous work on inspection planning, particularly vision sensor planning, is reviewed.

1.2.1 Computer Aided Process Planning

In the last few decades, there has been much research work in Computer Aided Process Planning (CAPP) [5]. Most of the research has shown that to effectively bridge CAD and CAM, *it is essential for the system to be able to obtain the higher level information and representations necessary for the task at hand.* This can be achieved using two approaches: feature-based design and feature extraction.

In the feature-based design approach, the designer uses primitives from a high-level representation scheme to design the part. In the feature extraction approach, the higher level information is derived from the low-level model representation of the part [9]. Based on the high level representations, there exist two different approaches for process planning: variant approach and generative approach [9].

In the variant approach, a Group Technology [13] is used to select a generic process plan from a database of standard process plans developed for each part family. The selected generic plan is then customized interactively for the particular part. If the plan does not exist for the family, the database is enhanced by an experienced process engineer. This approach, in its nature, is semiautomatic since it still needs human involvement. On the other hand, generative process planning systems are relatively new developments which construct plans from basic input data based on general rules for manufacturing. However, as Marefat *et al.* pointed out [9], the generative approach is still far from maturity.

Obviously, the generative approach is much more promising to achieve full automation in process planning. In this dissertation we prefer the generative approach to the variant approach.

In summary, the general trend in Computer Aided Process Planning (CAPP) has shown that feature, or high level model information, is playing a more and more important role in automated process planning. However, as a matter of fact, a global solution to the process planning problem is still far away [14].

1.2.2 Vision Sensor Planning

In recent years, sensor planning in computer vision is an emerging research area and many systems have been implemented. Here, we briefly review these systems. Based on whether a CAD model exists beforehand we can divide the vision sensor planning work into two categories:

- model-based sensor planning
- non-model-based sensor planning

In model-based sensor planning systems, the task is usually stated like this:

a finished part needs to be inspected to check if the dimensions of certain entities on the part are within the tolerance of the desired values in the design model.

In [15], Park *et al.* proposed a CAD-based planning and execution of inspection system. Aiming at compensating the gap between the representation of vision process and CAD database, they used rules to select important vision entities from the given CAD database and generated inspection procedures for each entity. Marefat and Kashyap [9] developed a knowledge-based inspection planning system for solid objects. Prieto *et al.* [16] proposed a CAD-based range sensor system for optimum $3D$ data acquisition. Other systems that used CAD models as input for sensor planning are described in [11, 17, 18, 19]. As we notice, many systems used rule-based approaches and, inevitably, they heavily depended on the human experience in rule generating.

In non-model-based sensor planning systems, there is no preexisting CAD model of the actual part that is to be inspected. In fact, to obtain a CAD model is the goal of these systems. Usually, the process to acquire CAD models from actual parts is called *reverse engineering* [20]. Since there is no *a priori* CAD model, the strategies to find the sensor configurations are quite different from those with models. Many sensor planning systems of this type rendered the problem as a "next-best-view"(NBV) problem. Pito [21] solved the NBV problem to determine which areas of a scanner's viewing volume need to be scanned to sample all of the visible surfaces of an *a priori* unknown object and where to position/control the scanner to sample them. Banta and Abidi [22] developed a system that reconstructs a model consisting of surfaces which have been viewed and volumes occluded from the camera's view. Using a similar idea, Reed and Allen [23] proposed an automated scene modeling system that consists of an incremental modeler and a sensor planner that analyzes the resulting partial model and computes the next sensor position.

As we have discussed, in our application, the sensor planning is model-based. The next section focuses on model-based vision sensor planning approaches.

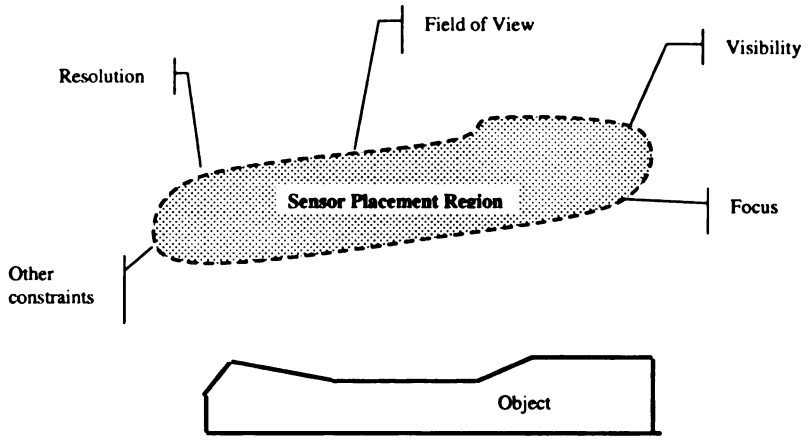

Figure 1.1: The vision sensor planning problem.

1.2.3 Model-based Vision Sensor Planning Approaches

Model-based vision sensor planning tries to understand and quantify the relationship between objects to be viewed and the sensors observing them in a model-based, task-directed way [24]. Figure 1.1 illustrates the vision sensor planning problem. Generally, to inspect a specific entity on an object, certain task constraints should be satisfied, which reflect the sensor's capabilities. A comprehensive survey on the model-based vision sensor planning can be found in [24].

Since the late 1980s, several vision sensor planning approaches have been proposed. Basically, they fall into two categories: generate-and-test and synthesis.

Generate-and-Test Approach

The generate-and-test approach was first introduced in the HEAVEN system developed by Sakane *et al.* [1]. It was then extended to other systems such as the Vision Illumination Object (VIO) system [25] and the Illumination Control Expert (ICE) system [26]. The generate-and-test approach simplifies the vision sensor planning problem into a search problem in a restricted solution space. Usually a tessellated sphere is adopted to model the potential sensor configurations which are evaluated with respect to the task constraints (see Figure 1.2). The determination of sensor

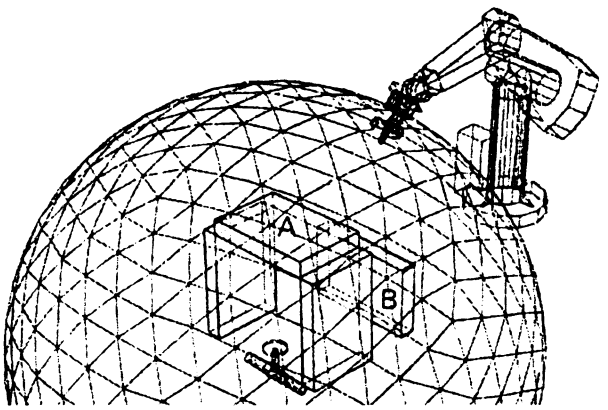

Figure 1.2: The generate-and-test approach (from [1]).

configurations is formulated as a search over the tessellated sphere guided by task related heuristics. In general, the generate-and-test approach is straightforward and easy to implement. However, the computational burden is relatively high due to the large number of facets on the tessellated sphere. Additionally, the focus and resolution constraints are not accounted for in this approach.

Synthesis Approach

Contrary to the generate-and-test approach, the synthesis approach forces an understanding of the causal relationships between the parameters to be planned and the goals to be achieved [24]. Task constraints are modeled as analytic functions and the task is formulated as a constraint-satisfaction problem.

Cowan and Kovesi's Work

The early work in this area was carried out by Cowan and Kovesi [2]. Their work was later extended by Cowan and Bergman [27] to include the planning of illuminator placement as well. This approach was to generate for each task constraint an equivalent geometric constraint, which in turn was satisfied in a domain of admissible locations in $3D$ space. However, the problem with their work is that some orientational degrees of freedom are omitted, which greatly reduces the solution space.

Tarabanis *et al.*'s Work

Cowan *et al.'s* work was later developed by Tarabanis *et al.* in their Machine Vision Planner (MVP) system [28].

In the MVP system, a generalized viewpoint is proposed, which consists of three positional degrees of freedom of the sensors and two orientational degrees of freedom (pan and tilt angle), as well as three optical parameters, namely, the back nodal point to image plane distance d, the focal length f, and the entrance pupil diameter a of the lens. In the MVP system, field of view, resolution, and focus constraints are characterized by closed-form relationships, which result in accurate solution regions and thus, eliminate the conservative nature of Cowan and Kovesi's approach. Then the solution regions obtained for all task constraints are combined in order to find parameter values that satisfy all constraints simultaneously. The problem is posed in an optimization setting in which globally admissible generalized viewpoint is sought that is central to the admissible region. Such a viewpoint will satisfy all the constraints and favor points far from the admissible region boundaries. For the visibility, they also provide an improved algorithm to find the visibility region in a polyhedral environment [29].

However, the problems with the MVP system are: 1) exact, closed-form constraint equations are not easy to solve, especially in high dimensional space, and 2) it is very difficult to mathematically express the exact solution regions to these equations.

1.2.4 New Trends in Sensor Planning

In recent years, new trends in the research of sensor planning have emerged, some of them are:

1) robust sensor planning.

In most of the previous sensor planning systems, the uncertainty of the sensor parameters was not considered. However, in real world implementation, sensor con-

figurations are usually required to be robust. That is, the sensor configuration can tolerate certain errors without affecting the inspection quality much. Several research work has been reported in robust sensor planning in recent years. Gu *et al.* proposed a robust approach for sensor placement in automated vision dimensional inspection [30]; Yao and Allen solved the multi-constraint robust viewpoint problem using tree annealing [31].

2) dynamic sensor planning.

Dynamic sensor planning deals with the planning of sensor configurations in an environment containing objects moving in known ways. Abrams *et al.* are the first researchers working in this area [32]. In their work, time becomes a factor and sensor configurations are generated to satisfy certain constraints over a time interval, which entails the calculation of the sweep volume of the moving objects.

1.2.5 Challenges

Generally, in most of the existing model-based vision sensor planning applications, only simple entities, such as line segments or selected entities on inspected parts, are considered [24]. However, in part dimensional inspection systems, continuous surfaces are the inspection subject, hence the full coverage of part surfaces should be guaranteed. Furthermore, the inspected surfaces are usually large enough to require that the camera planning problem be solved in a global environment, which, inevitably, brings new problems on efficiency and optimization that need to be considered. Within our knowledge, there has been no existing research that addresses vision sensor planning of complex surfaces in a global way. It is also observed that there lacks a general approach that can guide CAD-based automated process planning. In this dissertation we attempt to develop a framework for the automated sensor planning, in hoping that this framework will benefit many other similar CAD-based manufacturing process planning problems as well.

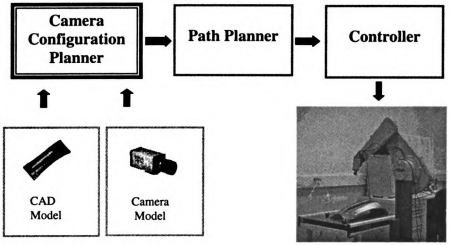

Figure 1.3: The automated CAD-based vision sensor planning system.

1.3 Our Vision Sensor Planning System

A vision sensor planning system is built in our lab to facilitate the planning and evaluation of sensor configurations. The framework of this system is depicted in Figure 1.3. Mainly, the system consists of three components:

- the sensor configuration planner
- the robot path generator
- the inspection implementation unit.

The sensor configuration planner is the key component in this framework. The planner takes the CAD model and the sensor model as two inputs, a number of sensor configurations are generated to inspect all surfaces of the part. For each sensor configuration and the corresponding inspected area, given task constraints are satisfied.

A robot path generator is then employed to calculate a time-efficient path to tour all these sensor configurations. If there are a great number of sensor configurations, it is not a trivial problem to find a time-efficient path.

Once a robot path is obtained, the robot controller drives the eye-in-hand robot to carry out the inspection task along the specified path. Currently our demonstration system does not include the structured light component. Instead, we choose an alternative method to verify the generated sensor configurations, which will be discussed later.

In the next two sections, the CAD model and the sensor model are introduced.

1.3.1 CAD Model

A CAD model contains the geometric information of a part. In $3D$ modeling, there are different representation schemes for an object. Typical schemes are [4]:

- Constructive solid geometry (CSG)
- Boundary representation (B-rep)
- Cell decomposition
- Sweeping

Among them, CSG and B-rep are the most prevalent. CSG uses primitive shapes as building blocks and boolean set operators to construct an object. In B-rep, objects are represented by their bounding faces. An alternative way categorizes the representation schemes into parametric representation and tessellation representation.

Parametric representation

In many applications, like die and mold design for automobile, ship and aircraft manufacturing, free-form surfaces (also called sculptured surfaces) are usually used [4]. Some common free-form surface models are bicubic surfaces, such as Bezier's surfaces, B-spline surfaces, and non-uniform rational B-spline surfaces (NURBS). All of them are parametric surfaces. Parametric surfaces generally satisfy certain continuity and smoothness constraints. Most of the parametric surfaces used in CAD

modeling have low curvatures. By piercing together these well-behaved parametric surfaces complicated geometry can be formed. Although parametric representation is mathematically accurate, its local nature brings difficulties for process planning [33]. In fact, surfaces used in CAD modeling are not designed for manufacturing purposes. Each parametric surface has its own variables, different from those of other surfaces, so it is difficult to have a global knowledge on the part geometry if we limit ourselves to parametric surfaces. While global knowledge of the part geometry is very important to automated process planning.

Tessellation representation

Tessellation representation is an important and useful scheme to approximate free-form surfaces. Triangle is most frequently used in tessellation. Usually, the more triangles, the less approximation error. To tessellate compound surfaces, or surfaces formed by multiple connected parametric surfaces, there are two approaches:

1) tessellate each parametric surface individually, and

2) tessellate all the compound surfaces as a whole.

The first approach is usually adopted in $3D$ visualization or rendering. Tessellating each individual parametric surface instead of all the compound surfaces as a whole can speed up the rendering of the object. However, there exist small gaps between surfaces, which needs to be dealt with carefully.

The second approach is usually seen in Finite Element Analysis (FEA). In this approach, the compound surfaces are tessellated as a whole and there are no gaps between surfaces.

The advantages of tessellation representation are obvious:

1) tessellation representation is a consistent, open data structure while parametric representation usually depends on the specific CAD system. Although CAD data exchange formats (like IGES) are being developed, there still exist heterogeneity and inconsistency among different CAD systems as far as the flavor of parametric surface

is concerned.

2) tessellation representation is far more attractive for calculation than parametric representation.

3) each tessellation unit, for example, a triangle, is a simple geometric entity, which is suitable for algorithm implementation.

As a matter of fact, there exist deviation errors, which are dependent on the number of triangles, between the tessellated surface and the parametric surface. However, according to Tarabanis *et al.*, the sensor configuration is not very sensitive to surface deviation errors [24], so these errors can be ignored in vision sensor planning.

1.3.2 Sensor Model

Pinhole model

To a first order, the imaging process can be approximated by a perspective transformation of object space onto image space with respect to a pinhole which serves as the center of projection. This model is widely used in computer vision. However, the effects associated with a lens, such as focusing and image brightness, are ignored. In this model, everything is perfectly focused and only a single ray of light reaches the image plane from any point of object space. To use this model, the lens is assumed to be thin and have an infinitesimally small aperture, which is not usually the case for most camera lenses in practical use.

Thick lens model

Instead of a simple pinhole model, Tarabanis *et al.* adopted a thick lens model in their MVP system [24], which is more appropriate to model compound lens systems in practical use.

The thick lens model is shown in Figure 1.4. For a thick lens, the optical center

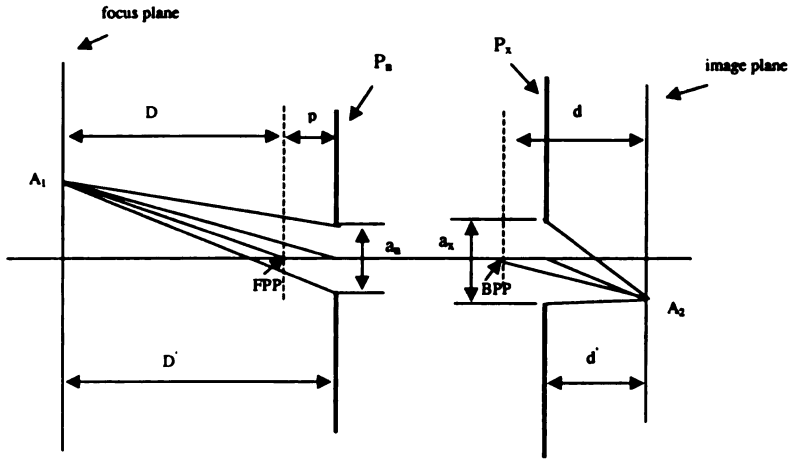

Figure 1.4: The thick lens model used in the MVP system.

is replaced by two points, the front nodal point and the back nodal point. The nodal points of an optical system are two conjugate points on the optical axis, such that any ray passing through the front nodal point emerges from the back nodal point in the direction parallel to that of the original ray. The principle points of an optical system are two conjugate points on the optical axis, such that planes drawn perpendicular to the optical axis through these points, are planes of unit magnification. The Gaussian lens formula applies to a thick lens when distances are measured from the principle points of the lens. When the media in the object and image spaces have the same refractive index (e.g. both in air), the front and back focal lengths are equal, and the principle points coincide with their respective nodal points. We adopt this thick lens model in our system.

1.4 Problems Addressed in This Dissertation

The main problems addressed in this dissertation are:

1) how to understand the geometric structure of complex surfaces and thereafter, how to extract the high-level geometric information based on low-level data representation to facilitate the vision sensor planning.

2) how to efficiently find the viewpoint(s) that can inspect a certain area of surface with all the task constraints, as well as the coverage requirement, satisfied.

3) how to improve the efficiency of the inspection system, or how to reduce the number of viewpoints and how to determine a time-efficient path to visit all the viewpoints.

4) how to combine the kinematics constraint into vision sensor planning such that all the viewpoints are well-behaved in kinematics sense.

1.5 Outline of This Dissertation

The outline of the dissertation is as follows: in Chapter 2, we propose the decomposition-based vision sensor planning approach for compound surfaces. This chapter mainly focuses on two subproblems: the decomposition of compound surfaces and the generation of sensor configurations (viewpoints) for well-behaved patches. Two optimal problems are discussed in Chapter 3: the minimum sensor configurations problem and the optimal kinematics performance problem. Chapter 4 focuses on the robot path planning, which is rendered as a Traveling Salesman Problem (TSP). A new approach is provided to solve it quickly, especially for large size problems. Simulation and experimental implementation issues are discussed in Chapter 5. Conclusions are provided in Chapter 6.

CHAPTER 2

SENSOR PLANNING FOR COMPOUND SURFACES

The main topic of this chapter is to solve the vision sensor planning problem for compound surfaces. Due to the complexity of the geometry of inspected surfaces, a divide-and-conquer strategy is adopted. Firstly, compound surfaces are decomposed into several patches with each satisfying certain geometric constraint. Secondly, based on a bounding box concept, sensor configurations that satisfy given task constraints are generated for each patch.

The first section gives out a mathematical formulation for the vision sensor planning problem and the second section reviews, in more detail, the synthesis approach on the solutions to individual task constraint, followed by the general camera planning strategy. The compound surfaces decomposition algorithm and the sensor configuration generation algorithm are discussed in detail thereafter. Finally, results from the tests on different real-world parts are reported.

2.1 Problem Formulation

Based on the discussions on the CAD model and the sensor model, the vision sensor planning problem can be formulated as follows:

Given a CAD model of the part P, which consists of triangles, i.e. $P = \{T_i\ (i = 1...n)\}$, and a camera model, design a set of sensor configurations $\mathbf{C} = \{C_i\ (i = 1...k)\}$ such that for any point p on the part surface, $p \in T_i \in P$, there exists at least one sensor configuration C_j ($C_j \in \mathbf{C}$), such that p is visible, in the field of view, resolvable and in focus.

Here a sensor configuration C is the necessary parameters that characterize a unique state of the sensor. For a camera, its configurations usually include optical settings, such as the back principal point to image plane distance d, the aperture of

the lens a, and the focal length f. However, due to the difficulties in automatically controlling these settings, we treat them as constants. Hence a sensor configuration, or viewpoint, of a camera consists of six parameters: the lens front nodal position: x, y, z; the camera orientation: pan angle θ, tilt angle ψ and the rotation angle around optical axis, ϕ. That is, $C = [x, y, z, \theta, \psi, \phi]^T$.

2.2 Background

Before we discuss the vision sensor planning approach, a more detailed look into the synthesis approach is provided and the viewpoint regions that satisfy each of the three constraints (resolution, field of view and focus) are presented.

2.2.1 Task Constraints and Their Solution Regions

To inspect a certain entity on a part, usually four task constraints should be satisfied: visibility, field of view, resolution and focus. The visibility constraint demands that there be no occlusion between the viewpoint and the entity to be inspected along the line of sight. There exist several methods that calculate the visibility region when an entity is viewed in a polyhedral environment [34]. In our vision sensor planning problem, the visibility problem is not critical due to the decomposition-based strategy we use. In fact, the visibility is implied in the other three constraints, which will be discussed later. The field of view constraint requires that the interested entity be imaged to the active area of the image plane. The resolution constraint ensures that a small entity be imaged to at least one pixel on the image plane. There are different definitions on the resolution constraint. According to Cowan and Kovesi, the resolution constraint was defined to be the minimum angle subtended by a given incremental surface length at the perspective center of the lens, which is illustrated in Figure 2.1. While in Tarabanis *et al.*'s work, the resolution is defined as a conventional

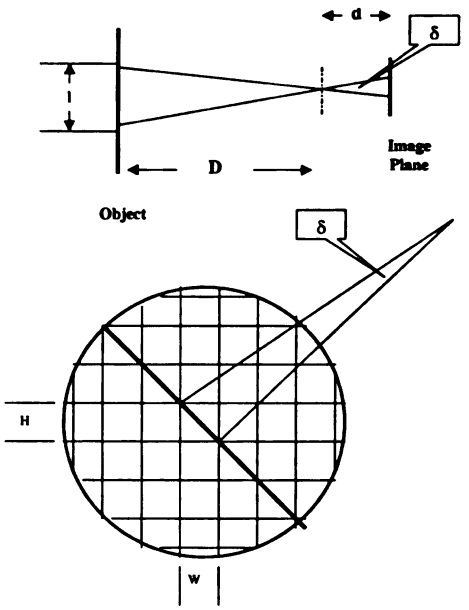

Figure 2.1: The resolution constraint used by Cowan and Kovesi.

pixel resolution. The focus constraint guarantees that the entity of interest is sharply imaged.

Cowan and Kovesi's Work

In Cowan and Kovesi's work, the solution to the field of view constraint is relatively straightforward. For a $3D$ object, Cowan and Kovesi constructed a circumscribing sphere with radius r and calculated the locus of viewpoints at a distance F from the sphere center with

$$F = \frac{r}{sin(\alpha/2)} \tag{2.1}$$

$$\alpha = 2arctan(\frac{P_{min}}{2d}) \tag{2.2}$$

Here P_{min} is the minimum dimension of the image plane. d is the distance from the back nodal point of the lens to the image plane.

With their resolution definition, Cowan and Kovesi determined the locus of the perspective center of the lens, when the given surface length is barely resolvable, to be the circular arc of points at which this surface length subtends a constant angle equal to the resolution angle [2]. Cowan and Kovesi further showed that the $3D$ region of viewpoints such that an incremental circle having diameter l subtends at least angle δ is a ball. The parameters of the ball are calculated as follows,

$$R = \frac{l}{2sin\delta}, \tag{2.3}$$

$$H = R\cos\delta, \tag{2.4}$$

$$\delta = 2\arctan\frac{\sqrt{h^2 + w^2}}{2d}. \tag{2.5}$$

Here H is the height of the ball center to the circle. R is the radius of the ball. h, w are the height and width of one pixel on the image plane. d is the distance from the back nodal point of the lens to the image plane.

To view a polygonal surface with a minimum spatial resolution, the viewpoint must lie in the intersection of the resolution balls of all the vertices on the surface. To view a polygonal surface, it is sufficient for the camera to lie within the intersection of the resolution balls of the vertices of the polygon's convex hull. Figure 2.2 displays the viewpoint region satisfying the resolution constraint when viewing a triangle $\triangle ABC$.

For the focus constraint, Cowan and Kovesi made use of the depth-of-field formula developed by Krotkov [35] that gives the range of distances of entity points that have

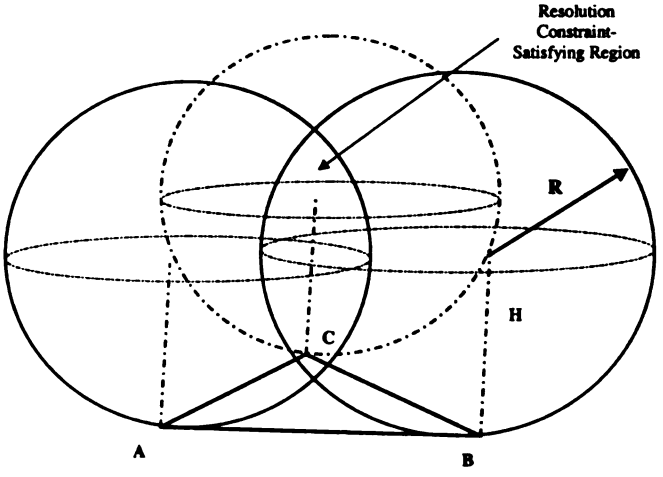

Figure 2.2: The resolution constraint-satisfying region for a triangle.

a blur circle diameter which is less than the minimum pixel dimension, and therefore in focus. The set of viewpoints, for which an inspected surface has its farthest point at a distance equal to the upper limit of this range and similarly its nearest point at a distance equal to the lower limit, defines the boundary of the locus of admissible viewpoints for this constraint. Instead of measuring these distances along the optical axis as required by the Gaussian lens law, Cowan and Kovesi measured it radially. In three dimensions, the camera optical axis was assumed to pass through the center of a circumscribing sphere of the entities, thus the orientational degrees of freedom were omitted, which greatly reduces the number of potential viewpoints.

Figure 2.3 shows the regions corresponding to the three constraints (field of view, resolution, focus) when a square is inspected. By intersecting all the admissible regions, Cowan and Kovesi were able to find the solution region in which a camera viewpoint satisfies all the task constraints.

Tarabanis *et al.'s* Work

In Tarabanis *et al.'s* work, the relationship describing the field of view-satisfying locus of generalized viewpoints is given by the implicit formula as follows,

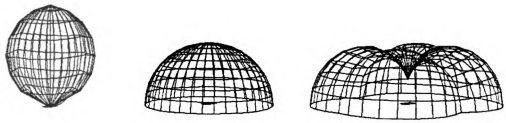

Figure 2.3: The viewpoint region for 3 task constraints: resolution, field of view and focus (from [2]).

$$(\vec{r_K} - \vec{r_V}) \cdot \vec{v} \geq cos(\frac{\alpha}{2}) \|\vec{r_K} - \vec{r_V}\| \tag{2.6}$$

where r_V is the position vector of the front nodal point V, v is the unit vector along the optical axis in the viewing direction, α is the field-of-view angle. r_K is the position vector given by the relationship $r_K = r_C - R_0 v$, where $R_0 = R_f / sin(\alpha/2)$ and r_C is the position vector of the center of a sphere circumscribing the object and R_f is the radius of the sphere.

For the resolution constraint, pixel resolution is used to indicate the approximate size of the smallest scene entity which can be seen by the vision system. The entities considered are line segments or general entities that can be approximated by a finite set of line segments. The locus of resolution-satisfying general viewpoints is described in vector form by the following formula:

$$\frac{\|[(r_A - r_V) \times u_l] \times v\|}{[(r_A - r_V) \cdot v][(r_B - r_V) \cdot v]} - \frac{w}{dl} \geq 0 \tag{2.7}$$

where r_A, r_B are the vector of the vertices, A and B of the linear entity AB, with respect to the object world coordinate system, u_l is the unit vector along the linear entity AB, l is the length of the linear entity AB, f is the focal length of the lens

and w is the required length of the image of AB.

For the focus constraint, using a different method from Cowan and Kovesi's, Tarabanis *et al.* found that the region satisfying the focus constraint for a general polyhedral entity is the band between the planes perpendicular to the viewing direction, and at distance D_1 and D_2 from the farthest and closest entity vertices, respectively.

$$(r_c - r_V) \cdot v - D_2 \geq 0 \tag{2.8}$$

$$D_1 - (r_f - r_V) \cdot v \geq 0 \tag{2.9}$$

where r_c (r_f) is the position vector of the nearest (farthest) entity vertex from the front nodal point of the lens along the viewing direction. D_1 and D_2 are given by:

$$D_1 = \frac{afd + cfp}{a(d-f) - cf}, \tag{2.10}$$

$$D_2 = \frac{afd - cfp}{a(d-f) + cf}. \tag{2.11}$$

Here p is the signed offset between the front nodal point and the entrance pupil, c is the limiting blur circle diameter.

2.2.2 Task Constraints Considered in Our System

Specific to our inspection problem, the following task constraints are adopted.

1) Visibility.

Denote the $3D$ position of the viewpoint as O, suppose a point p on the part

surface has a normal $\vec{n}$. The visibility constraint requires that $\vec{po} \cdot \vec{n} \geq 0$ and the line segment $\bar{po}$ is not occluded by any other surface. Later we will see that, in a patch, visibility can be guaranteed if the maximum angle between the average normal of the patch and normal of each triangle is less than a certain value.

2) Field of view

Different from the circular field of view constraint usually used in previous work, a rectangular field of view is defined. This field of view is more practical and it is characterized by a field of view angle α, which is extended by the two side planes corresponding to the two shorter edges of the field of view rectangle (see Figure 2.4).

3) Resolution

Tarabanis *et al.*'s calculation of resolution constraint is more accurate, however, it is not easy to calculate the solution region. To simplify the implementation, Cowan and Kovesi's resolution definition and its method is adopted to obtain the resolution region. To view a tessellated surface, the resolution region is the intersection of the resolution balls of all the vertices.

4) Focus

Tarabanis *et al.*'s solution is adopted to calculate the in-focus region. For a given patch the solution region is the band between the planes perpendicular to the viewing direction and at distances D_1 and D_2 from the farthest and nearest patch vertex, respectively.

Based on the above discussions, the view cone of the camera can be modeled as in Figure 2.4, where the two parallel planes denote the two depth-of-field planes corresponding to D_1 and D_2. α is the field of view angle.

2.3 Camera Planning for Compound Surfaces

In this section, the CAD model data representation is discussed, then the camera planning strategy is developed.

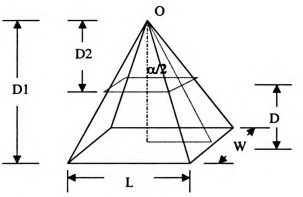

Figure 2.4: The view cone of the camera.

Figure 2.5: The $3D$ rendering of the part–bpilar.

2.3.1 CAD Model Data Representation

A tessellation representation, the Virtual Reality Modeling Language (VRML), is chosen as the input CAD model data representation. VRML is a standard for web-based rendering of objects [36] and it belongs to the type of tessellation by each parametric surface we mentioned in Chapter 1. This data representation is favorite to us since it combines the advantages of both the parametric surface and the tessellation in that we can retrieve the well-behaved parametric surfaces easily and meanwhile simplify the calculation of viewpoint by using triangles. Figure 2.5 and Figure 2.6 display an original part called *bpilar* and its tessellated counterpart, respectively.

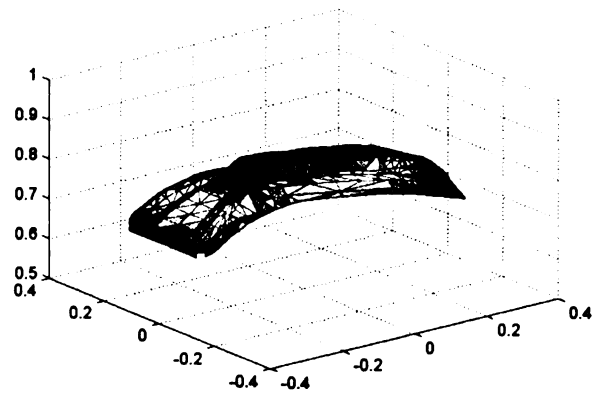

Figure 2.6: The bpilar in triangular representation.

2.3.2 Camera Planning Strategy

It is observed that most parts in automotive manufacturing, especially vehicle bodies, share a common characteristic: they have several continuous, flat patches joined by edges, or smooth transitional surfaces like fillets. Therefore, it is possible to partition the surfaces of the part into flat patches such that each flat patch has attributes desirable for the camera planning: there are no complex geometries on the flat patch; the curvatures of the flat patch do not change abruptly; the flat patch extends more in length and width direction than in depth direction. In this sense, the flat patches represent the high level information extracted from the low level CAD data.

However, due to the large number of parametric surfaces, it is not efficient to plan viewpoints for each parametric surface separately since it is possible for one viewpoint to inspect several surfaces that have similar normals. Therefore the challenge is, how can we merge those surfaces with similar normals so that we can obtain less patches and eventually, less viewpoints? This problem bears certain similarity with a segmentation problem in image analysis [37]. Inspired by this observation, a graph-based algorithm is developed to merge well-behaved surfaces into big flat patches.

The overall camera planning strategy is shown in Figure 2.7. To decompose the surfaces of the part into flat patches, we first construct an adjacency graph with each node representing a surface and each edge representing an adjacency relationship

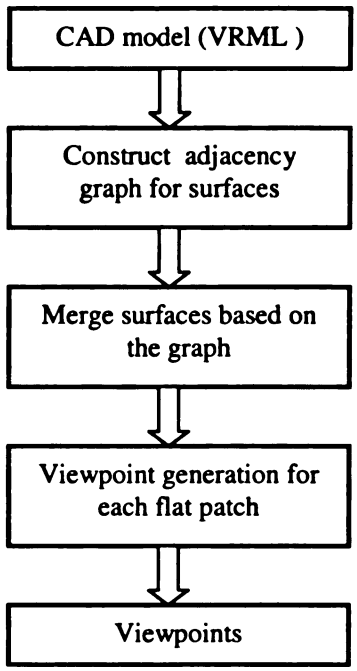

Figure 2.7: The camera planning algorithm.

between the corresponding two surfaces. Then, based on this graph a merging algorithm is proposed to combine the surfaces with similar normals. Finally, viewpoints are generated for each flat patch.

2.4 Compound Surfaces Decomposition

In this section, the flatness constraint is introduced first, followed by the concept of the Flat Patch Adjacency Graph (FPAG) and its construction method. The rest of this section describes the merging algorithm on the FPAG.

2.4.1 Flatness Constraint

To begin with, some definitions are introduced.

Definition 2.4.1 (surface) *A* **surface** *is a collection of all the connected triangles corresponding to a parametric surface.*

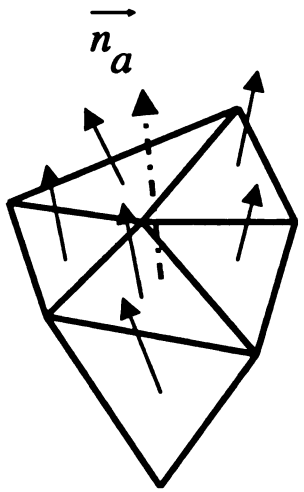

Figure 2.8: The average normal of a surface.

Definition 2.4.2 (average normal of a surface) *The* **average normal of a surface** *is the area-weighted average of all the unit normals of the triangles in the surface. i.e.,*

$$\vec{n}_a = \frac{\sum_{i=1}^{k} A_i \vec{n}_i}{\sum_{i=1}^{k} A_i}, \tag{2.12}$$

here, $\vec{n}_i$ and A_i are the unit normal and the area of triangle T_i, respectively; k is the number of triangles in the surface.

A surface and its average normal are shown in Figure 2.8.

Definition 2.4.3 (patch) *A* **patch** *is a collection of n ($n \geq 1$) connected surfaces.*

Definition 2.4.4 (average normal of a patch) *The* **average normal of a patch,** *$\vec{n}_p$, is the area-weighted average of all the unit normals of the triangles in the patch, or*

$$\vec{n}_p = \frac{\sum_{i=1}^{m} A_i \vec{n}_i}{\sum_{i=1}^{m} A_i}, \tag{2.13}$$

m is the number of triangles in the patch.

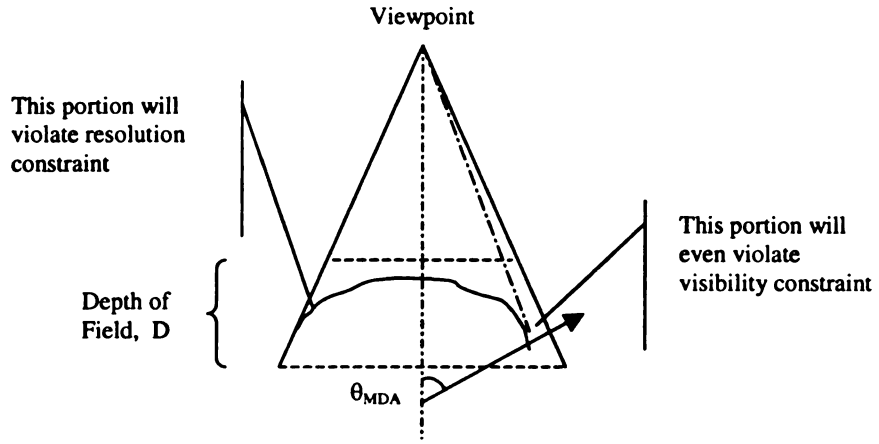

Figure 2.9: The flatness constraint.

To facilitate the camera planning, the flatness is chosen as the geometric constraint on the patch. The reason lies in the following fact: the bigger curvature a patch has, the less likely to find a viewpoint which satisfies the resolution and the focus constraint; if the curvature is too big, even the visibility constraint will be violated. Figure 2.9 illustrates the cases where the resolution and the visibility constraint are likely to be violated. Furthermore, if the patch has both big area and big curvature, the patch can not be hold in the depth of field range D, thus the focus constraint will not be satisfied.

The flatness of a patch can be characterized by the maximum deviation angle defined below.

Definition 2.4.5 (maximum deviation angle) *The* **maximum deviation angle** **(θ_{MDA})** *of a patch is the maximum of all the angles between the average normal of the patch, $\vec{n}_p$, and the normal of each triangle, $\vec{n}_i$. i.e.,*

$$\theta_{MDA} = \max_i(cos^{-1}(\frac{\vec{n}_p \cdot \vec{n}_i}{\| \vec{n}_p \cdot \vec{n}_i \|})) \tag{2.14}$$

Here $\| \cdot \|$ denotes the length of a vector in Euclidean space.

Definition 2.4.6 (flat patch) *A* **flat patch** *is a patch that satifies* $\theta_{MDA} \leq \theta_{th}$. *Here* θ_{th} *is a predetermined flatness threshold.*

As has been discussed in Chapter 1, parametric surfaces usually have low curvatures and are well-behaved, so it is reasonable to assume that each surface satisfies the flatness constraint if a suitable θ_{th} is chosen. Hence, according to the above definition, a surface is a flat patch.

For a flat patch, another concept, the maximum-area-direction, can be introduced, which designates the direction that when the flat patch is orthographically projected (with scaling factor $m = 1$) along this direction, the image of the patch has the maximum area. What is interesting is that the maximum-area-direction of a flat patch is the same direction of its area-weighted average normal. It can be shown as follows.

When the scaling factor equals 1, the image area of a patch in orthographic projection is:

$$A = \sum_{i=1}^{m} A_i |\vec{n}_i \cdot \vec{v}|. \tag{2.15}$$

Here $\vec{v}$ is the direction of orthographic projection. To find $\vec{v}$ such that A is maximum, we have

$$\frac{dA}{d\vec{v}} = 0, \tag{2.16}$$

which results in

$$\vec{v} = \frac{\sum_{i=1}^{m} A_i \vec{n}_i}{\sum_{i=1}^{m} A_i} \Big/ \| \frac{\sum_{i=1}^{m} A_i \vec{n}_i}{\sum_{i=1}^{m} A_i} \|, \tag{2.17}$$

that means the maximum-area-direction of a patch is the same direction as the area-weighted average normal of that patch.

For a flat patch, this equivalent relationship still holds under the perspective projection since, when the scene depth is small relative to the average distance from the camera, perspective projection can be approximated by orthographic projection [38]. This small-scene-depth condition can be guaranteed by the flatness of the patch.

2.4.2 Flat Patch Adjacency Graph

Construct Adjacency Relationships among Surfaces

In VRML, since each parametric surface is tessellated separately, no two triangles from two different surfaces are connected (or share a common edge). Therefore, all the triangles belonging to one surface can be identified based on the connectedness relationships among triangles.

A "growth" algorithm is proposed to identify the surfaces. First, a seed triangle is arbitrarily selected, then its edge-neighboring triangles, or the triangles sharing one common edge with the seed triangle, are pulled in to form a sub-surface, or a portion of a surface. Then, for each triangle in the sub-surface, its edge-neighboring triangles are pulled in. This process continues until no more triangles can be added. The final collection of triangles form a surface. Again, among the remaining triangles, another triangle is arbitrarily selected as the seed and the above procedures are repeated until each triangle is in one of the resulting surfaces. Since each parametric surface is tessellated separately, whichever seed triangle to choose does not affect the identifying results.

To build the adjacency relationships among surfaces, it is necessary to calculate the distance between two surfaces. Following are two definitions related to the adjacency relationship.

```
Adjacent=FALSE;

for each triangle Ti in Surface A do
 for edge Ej=(Vj1,Vj2) in Ti do
  if(Ej is a boundary-edge of Surface A)
   if(Vertex2Surface_Distance(Vj1,Surface B) < THRESHOLD
    && Vertex2Surface_Distance(Vj2,Surface B) < THRESHOLD)
    Adjacent=TRUE;

for each triangle Ti in Surface B do
 for edge Ej=(Vj1,Vj2) in Ti do
  if(Ej is a boundary-edge of Surface B)
   if(Vertex2Surface_Distance(Vj1,Surface A) < THRESHOLD
    && Vertex2Surface_Distance(Vj2,Surface A) < THRESHOLD)
    Adjacent=TRUE;
```

Figure 2.10: The algorithm to check the adjacency relationships.

Definition 2.4.7 (border-edge) *An edge of a triangle in surface A is called a* **border-edge** *to surface B if and only if both of its vertices are close enough, that is, less than certain threshold, to surface B.*

Definition 2.4.8 (adjacent) *Two surfaces, A and B, are* **adjacent** *if surface A (or B) has at least one border-edge to surface B(or A).*

Two patches are adjacent if there exist at least two adjacent surfaces, each from one patch.

The algorithm to determine whether two surfaces A and B, are adjacent is shown in Figure 2.10. Every boundary edge of one surface is checked to see if both vertices of the edge are close enough (less than a predetermined threshold) to another surface.

In the above algorithm, it is necessary to calculate the distance from one vertex to one surface, which is essentially the minimum distance of all the distances from the vertex to each triangle in that surface. Usually the vertex is not in the plane where the triangle resides, so the vertex is projected onto the plane and thus the problem is converted into $2D$. Figure 2.11 illustrates this idea and the algorithm to calculate the vertex-to-triangle distance is shown in Figure 2.12.

In the plane, the distance from the project point V_p to the triangle T, D_p, is obtained by taking the minimum of the distances from V_p to each of the three edges

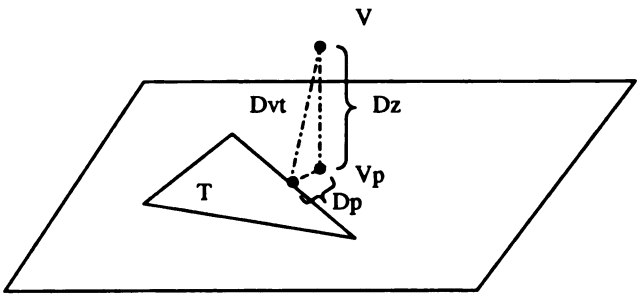

Figure 2.11: Project the vertex onto the plane.

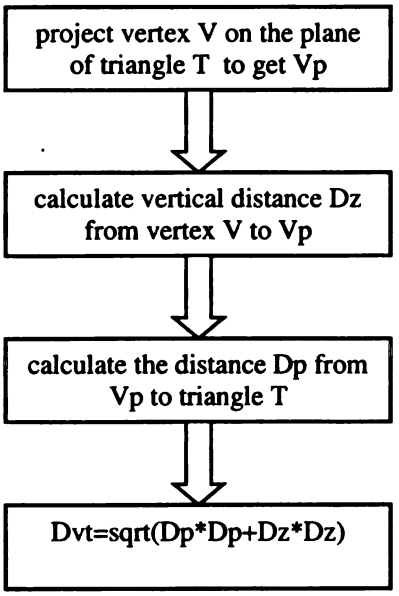

Figure 2.12: The algorithm to calculate the vertex-to-triangle distance.

of T. However, if V_p is inside the triangle, the distance $D_p = 0$. The vertex-to-triangle distance is calculated as $D_{vt} = \sqrt{(D_z)^2 + (D_p)^2}$. A similar algorithm can be found in [39].

Flat Patch Adjacency Graph

Once the adjacency relationships among surfaces are obtained, a Flat Patch Adjacency Graph (FPAG) can be constructed. Since a surface is a flat patch, this nomenclature is suitable to describe the initial adjacency graph where each node is a surface.

Definition 2.4.9 (Flat Patch Adjacency Graph (FPAG)) *A* **Flat Patch Adjacency Graph** *is a undirected graph $G = (V, E)$ where V is the set of nodes and $E \subset V \times V$ is the set of edges. Each node $v_i = \{\vec{n}_{pi}, A_{pi}, \theta_{MDAi}\}$ represents a flat patch with the triple representing the average normal, area and maximum deviation angle of the flat patch respectively. Between two nodes v_i and v_j exists the edge e_{ij} if the two nodes are adjacent. The weight of the edge, $w(i, j)$, is the angle between the average normals of the two adjacent flat patches.*

Definition 2.4.10 (merging) *A* **merging** *on a FPAG is the union of two adjacent nodes v_i and v_j into one node v_{ij} if v_{ij} is a flat patch.*

Following steps take place in a merging action.

1. $v_{ij} = v_i \oplus v_j$.

   Here, the merging operator $\oplus$ is defined as follows,

$$v_i \oplus v_j = \{\vec{n}_{pi}, A_{pi}, \theta_{MDAi}\} \oplus \{\vec{n}_{pj}, A_{pj}, \theta_{MDAj}\} = \{\frac{\vec{n}_{pi}A_{pi} + \vec{n}_{pj}A_{pj}}{A_{pi} + A_{pj}}, A_{pi} + A_{pj}, \theta_{MDAij}\} \tag{2.18}$$

2. *Replace node v_i, v_j and edge e_{ij} with node v_{ij}.*

Figure 2.13: Example compound surfaces and the corresponding initial FPAG.

3. Recalculate the edge weight of all the affected edges.

In equation (2.18), θ_{MDAij} is calculated after the new average normal is obtained. According to the above definitions, the FPAG has the following properties:

Property 1. A FPAG is a connected graph.

Property 2. When a FPAG undergoes a merging, it is still a FPAG.

Property 3. Each merging reduces the number of nodes in FPAG by 1.

Among the above three properties, Property 1 reflects the fact that all the surfaces of the part are connected. Property 2 is self-evident based on the definition of FPAG and it emphasizes that each node in the FPAG represents a flat patch at any time in the merging process. Property 3 means that the more mergings take place, the less flat patches are obtained.

The example compound surfaces and the corresponding initial FPAG are shown in Figure 2.13.

2.4.3 FPAG-based Surface Merging

Merging adjacent surfaces with similar normals into big patches resembles the segmentation problem in image analysis. Generally, the segmentation problem aims to partition an image into a number of homogeneous segments and the union of any two adjacent segments results in a heterogeneous segment [37].

Here a concept, closest normal pair, is defined.

Definition 2.4.11 (closest normal pair (CNP)) *The* **closest normal pair** *is the two adjacent nodes with the minimum edge weight in the FPAG.*

The basic idea of the merging algorithm on the Flat Patch Adjacency Graph (FPAG) is as follows: in the FPAG, find a pair of flat patches with the closest normals and test if the merged patch is a flat patch, i.e., the maximum deviation angle is less than a threshold θ_{th}. If yes, update the FPAG. Otherwise, keep the two flat patches and assign a big value BV ($BV \geq 2\theta_{th}$) to the weight of the edge between them to keep them from competing in the following iterations if neither of the two flat patches corresponding to the edge is changed. Then repeat the whole process on the updated FPAG. When updating the FPAG, the edge with big value weight BV will be recalculated if any of its corresponding flat patches is merged with other flat patches.

The ending condition is that all the edge weights are bigger than $2\theta_{th}$ (θ_{th} is the flatness threshold), which is obtained on the basis of the following observation:

If the angle between the average normal of two flat patches is greater than $2\theta_{th}$, then the maximum deviation angle (θ_{MDA}) of the merged patch must be greater than θ_{th}, which means the merged patch is not a flat patch.

This observation is obvious since the average normal of the combined patch is the linear combination of the average normal of the two flat patches.

According to Property 3, each merging reduces the number of nodes by 1, so the merging algorithm guarantees that the number of nodes, i.e., the number of flat patches decreases. At the end of the iterations, the algorithm will find out the minimum number of flat patches and further merging of any two flat patches will violate the flatness constraint. The algorithm is shown in Figure 2.14 and the merging process is illustrated in Figure 2.15.

In Equation(2.18), the non-normalized vector is used to represent the average normal of a patch. In this way the computational cost can be reduced when we

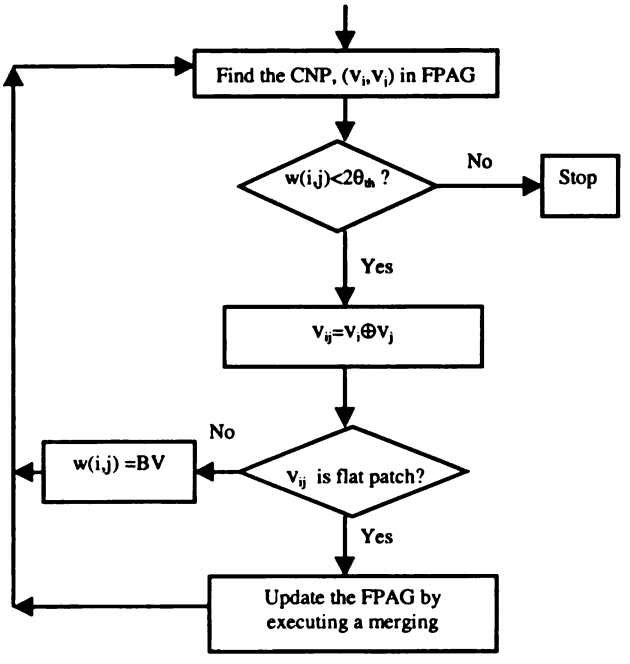

Figure 2.14: The merging algorithm based on FPAG.

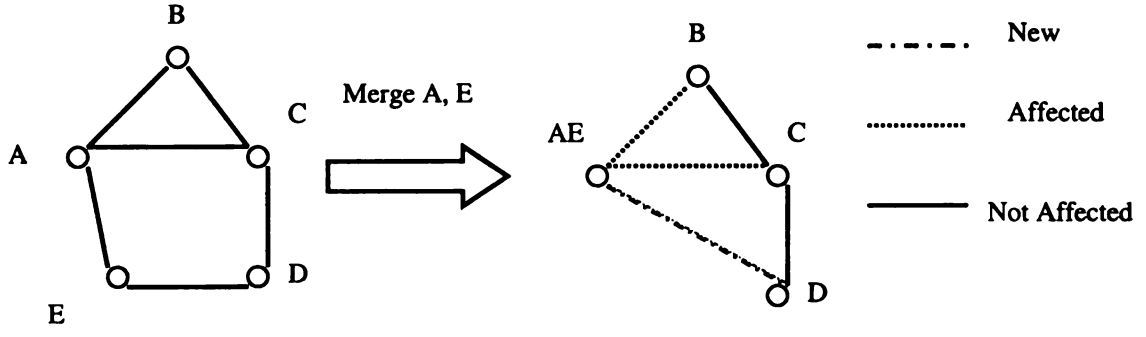

Figure 2.15: The merging process.

calculate the average normal of the merged flat patch. We simply take the area-weighted average of the normal of the two patches to get the average normal of the merged patch, thus avoid enumerating all the triangles in the merged patch.

2.5 Viewpoint Generation

Since flat patches have been generated, the critical problem is brought up: how to find the constraint-satisfying viewpoints for each flat patch in an efficient way?

A new concept called bounding box is proposed to facilitate the viewpoint planning and, based on this concept, a recursive algorithm is developed to find all the viewpoints for that flat patch.

2.5.1 Bounding Box

A "bounding box" concept is proposed to effectively combine the four task constraints and simplify the viewpoint search. The viewpoint generating algorithm based on this concept is an integration of the generate-and-test and the synthesis approach in the sense that it adopts the search technique from the generate-and-test approach and the resolution, focus constraint solution from the synthesis approach.

For a given flat patch, its bounding box is constructed by projecting all the vertices on three orthonormal directions and finding the corresponding farthest and nearest project point. Here the three orthonormal directions are: front, or the viewing direction; up and right, the directions that determine the camera rotation around the optical axis.

The opposite direction of the average normal (or the maximum-area-direction) of the given flat patch is chosen as the front direction. Then all vertices on the patch are projected to a plane that is perpendicular to the front direction. The up and right direction are determined so that the smallest rectangular field of view with specified

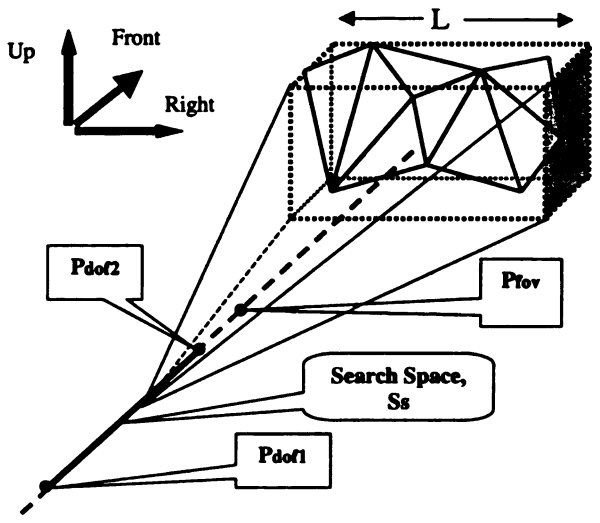

Figure 2.16: Bounding box and the potential viewpoint region.

length/width ratio can cover all the project points.

Based on the bounding box, the viewpoints that satisfy the four task constraints locate in a restricted region (see Figure 2.16). We choose the center of the bounding box as the look-at position of the camera, the up direction of the bounding box as the look-up direction. Now the look-from position of the camera is restricted on the line of sight which passes through the look-at point. To determine it, the line of sight is intersected with three constraint regions: the field of view region, which gives the lowest position P_{fov} of the camera on the line of sight with

$$P_{fov} = \frac{L}{2\tan(\alpha/2)}, \tag{2.19}$$

where L is the length of the bounding box, α is the field of view angle of the camera; the in-focus region, which determines two points P_{dof1} and P_{dof2} on the line of sight corresponding to the two planes at distances D_1 and D_2 from the farthest and nearest patch vertex; the resolution region, which is the intersection of the resolution balls for each vertex on the patch. However, due to the high computational cost in calculating the intersection of resolution balls, an indirect method is used: discretize the search space formed by the line of sight and the field of view, in-focus region and test

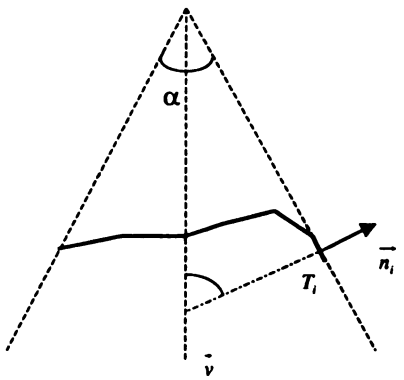

Figure 2.17: Visibility for a flat patch.

each discrete point to see if the distances between this point and the centers of the resolution balls are all less than the resolution ball radius R.

The visibility of the patch is guaranteed by the bounding box and the flatness constraint of the patch as long as the flat constraint threshold θ_{th} satisfies the following condition,

$$\theta_{th} \leq \frac{\pi}{2} - \frac{\alpha}{2}, \tag{2.20}$$

This condition is illustrated in Figure 2.17.

Any position in the intersection of the line of sight and the three constraint regions is an admissible look-from position. In practice, we choose the lowest solution position to obtain a larger image.

Some illustration is necessary for the determination of the line of sight. In the proposed algorithm, the most important question one might ask is "*is the chosen line of sight most likely to result in admissible viewpoints?*".

Among the four task constraints, it is obvious that we only need consider the resolution constraint. We have the following proposition regarding the resolution constraint and the viewing direction.

Proposition 2.5.1 *For a given flat patch, if there is one viewpoint that satisfies*

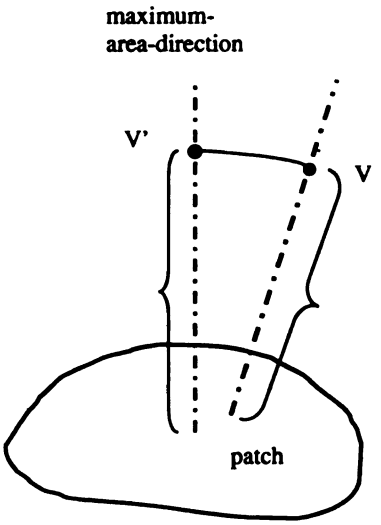

Figure 2.18: Resolution constraint and maximum-area-direction.

the resolution constraint and has a viewing direction other than the maximum-area-direction, then, there must be one viewpoint on the maximum-area-direction that satisfies the resolution constraint.

Proof.

In Figure 2.18, if viewpoint V is on a viewing direction other than the maximum-area-direction, and V satisfies the resolution constraint, then we can find a viewpoint V' on the maximum-area-direction with the same distance to the patch as V. We then get two images of the patch, image I' from V' and image I from V. Under the orthographic projection assumption and the maximum-area-direction definition, the area of image I', is no less than the area of image I, which implies that the image I' satisfies the resolution requirement too. ◁

The above proposition implies that the line of sight in the maximum-area-direction is the most likely potential solution direction to the resolution constraint.

2.5.2 Recursive Viewpoint Generation Algorithm

A recursive algorithm is used to determine all the viewpoints for a flat patch. If the intersection of line of sight and all the constraint regions of the current patch

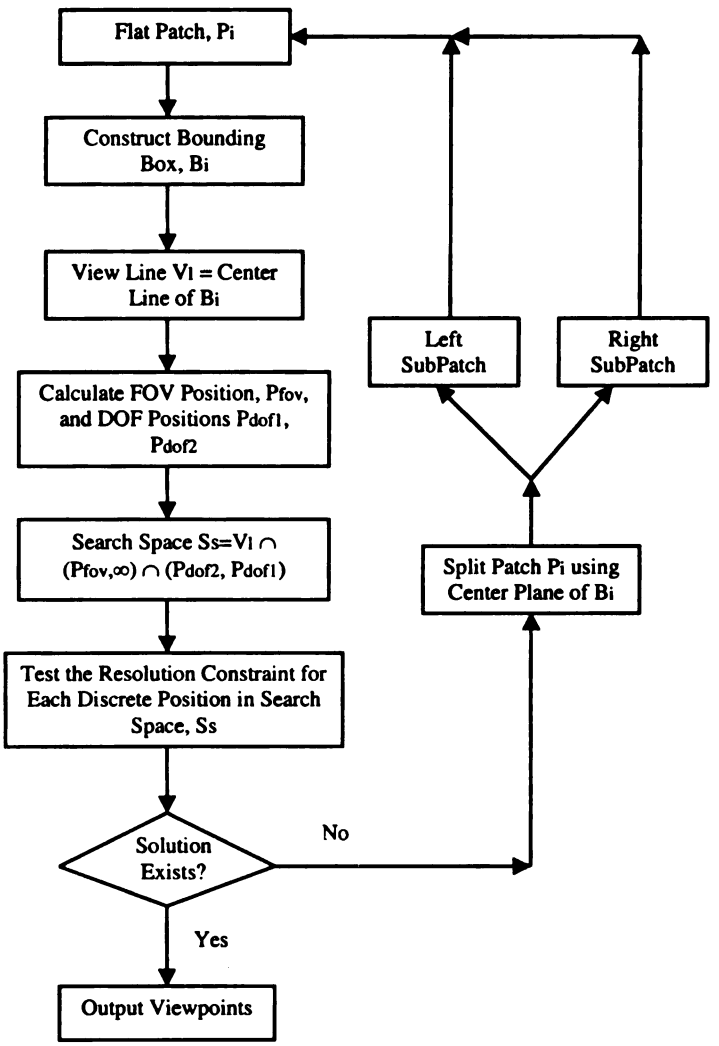

Figure 2.19: Recursive algorithm to find the viewpoints for a flat patch.

is null, which means no viewpoint exists that satisfies all constraints, we split the patch into two subpatches using the center plane of the bounding box along the length direction. For each subpatch, a new bounding box is constructed and the viewpoints are searched. This process continues until every patch or subpatch finds its viewpoints. The algorithm is shown in Figure 2.19.

2.6 Algorithm Implementation and Results

The surface merging algorithm and the viewpoint generating algorithm are implemented using $C++$ on a $Pentium^{TM}$ III 500Mhz PC and they are tested on different

stamping sheet metal parts from Ford Motor Company.

The camera we use is a Hitachi $KPD-50$ with a H0612FI lens and its parameters are:

$$d = 12.65mm,\ a = 0.78mm,\ f = 12.5mm,$$

$$p = 2.0mm,\ c = 3.75\mu m,\ \alpha = \frac{\pi}{6},$$

$$l = 2mm,\ h = 8.4\mu m,\ w = 9.8\mu m,$$

which result in

$$D_1 = 1.76m, D_1 = 0.75m, R = 0.98m, H = 0.98m$$

First all the triangles are read into the program. Then the connectness relationship among the triangles is constructed based on the vertex indices. It is found that the object-oriented feature of $C++$ is suitable to explore the hierarchical structure among the point, line, triangle, patch, etc.

Summarized here are some results on the *bpilar*, which has 293 different parametric surfaces. The VRML data of the part is generated from SDRC's I-DEAS *Master Series*^{TM}6.0 [40]. With an error tolerance of 3 mm it is tessellated into 2680 triangles. Its triangular representation is already shown in Figure 2.6. The flatness constraint threshold θ_{th} is $\frac{\pi}{4}$.

The merging algorithm generates 18 flat patches. Figure 2.20, Figure 2.21 and Figure 2.22 show three of them. In these figures, the front and back rectangles of the bounding boxes are shown, with a line connecting the center of the bounding box and the lower right corner of the front rectangle. Each flat patch consists of multiple surfaces. Obviously, the merging takes place $293 - 18 = 275$ times before it stops.

Once the flat patches are generated, the recursive viewpoint generating algorithm

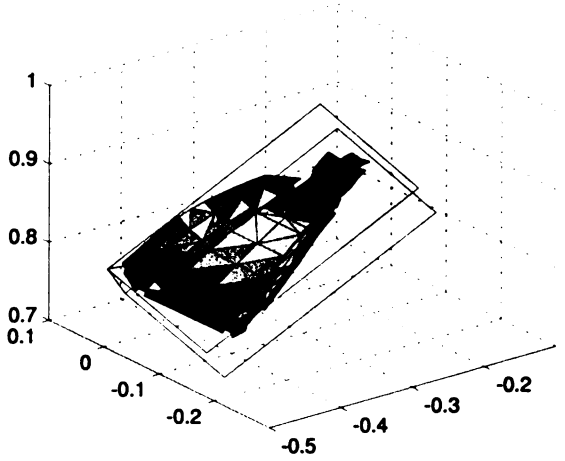

Figure 2.20: Flat Patch 1.

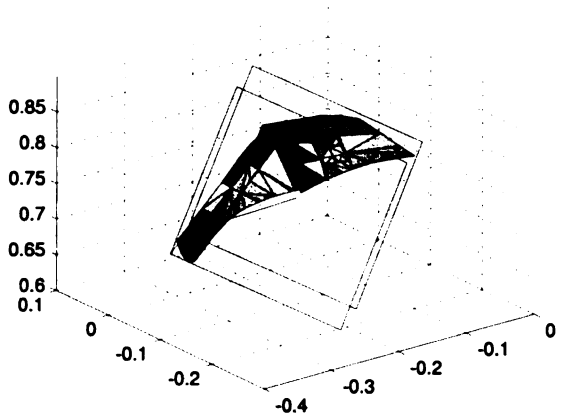

Figure 2.21: Flat Patch 2.

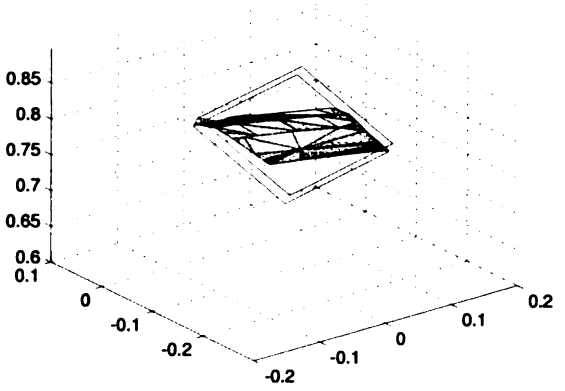

Figure 2.22: Flat Patch 3.

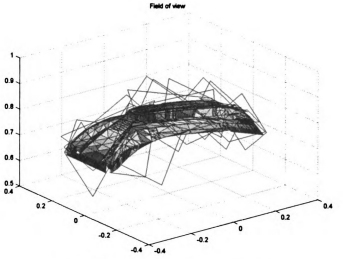

Figure 2.23: All the viewpoints.

Table 2.1: Testing results on different parts

Part	No. of Triangle	No. of Surface	No. of Flat Patch	No. of Viewpoint	Running Time (min)
bpilar	2680	293	18	24	14
fender	2602	119	10	16	10
pillar	1806	136	20	32	8
floorpan	5015	345	40	54	25

is used to compute the viewpoints that satisfy all the four task constraints. Figure 2.23 shows all the viewpoints generated. In the figure, the frames show the actual field of view of all the viewpoints. Table 2.1 summarizes the testing results on different parts.

As it can be seen from the table, the number of flat patches is much smaller than the number of surfaces, which implies that most of the surfaces have similar average normals.

We also find that a big portion of the running time is spent in building the Flat Patch Adjacency Graph.

2.7 Discussions

In this chapter, a divide-and-conquer strategy is adopted in the vision sensor planning. First, based on the tessellation representation of the surfaces of the part, a merging algorithm on the Flat Patch Adjacency Graph (FPAG) of the part surfaces is developed. This algorithm utilizes the adjacency information of the surfaces and the similarity of the surface normals to cluster the homogeneous surfaces into big flat patches. Second, for each flat patch, a recursive algorithm which combines existing vision sensor planning approaches is developed to find viewpoints that satisfy certain task constraints. Experiments on real-world parts show that the new approach is successful.

It is worth pointing out that the surface merging algorithm can also benefit many other CAD-based path planning problems like spray painting path planning because it is important to generate large, global, flat patches to obtain time-efficient spraying paths [41].

CHAPTER 3

OPTIMIZATION IN VISION SENSOR PLANNING

Inspections in automotive manufacturing usually have to deal with large parts, while a camera has a relatively small field of view, which is dependent on the focus and the resolution capability of the camera. How to determine the camera viewpoints such that the part surfaces can be inspected satisfactorily with a minimum set of viewpoints is one problem studied in this chapter. By partitioning a large flat patch into near-square cells, candidate viewpoints are constructed to cover a certain number of neighboring cells. To find the minimum set of viewpoints, a set-partitioning problem is formulated.

Another practical problem that usually arises in the implementation of an inspection system is that the viewpoints, or correspondingly, the robot hand-tip positions, should be within the robot's reachability space and, furthermore, it is desirable that the robot can reach these viewpoints "comfortably". In this sense, both the four task constraints and the robot kinematics constraint should be considered in sensor planning. Based on the same cell-partitioning scheme, kinematics performance measure of the robot is assigned to each candidate viewpoint and a weighted set-covering problem is formulated to find the fully-covering viewpoints that achieve the highest kinematics performance.

3.1 Minimum Sensor Configuration Problem

3.1.1 Background

It is obvious that the number of sensor configurations will determine the inspection time as well as other subsequent costs for inspection. It is highly desirable to obtain minimum sensor configurations to cover all the surfaces of the part. Hence, an optimal problem can be formulated:

find the minimum sensor configurations such that any point on the part surface can be "covered" by at least one of these sensor configurations.

However, the minimum covering problem is not easy to solve. Even its $2D$ version, that is, to inspect all the interior edges of a polygon with minimum viewpoints, is a NP-hard problem [42], which implies that for large size problems there is no efficient algorithm to find the optimal solution. In fact, the latter is a traditional "art gallery" problem which tries to find the minimum number of guards necessary to see all of the walls (edges) of a gallery (a simple polygon) [43].

Recently, some research work on the minimum viewpoint problem has been reported. Danner and Kavraki [44] proposed a randomized approach to select art gallery-style guards (viewpoints) and connect them to form an inspection path. They took the visibility, the length of the line of sight and the incidence angle between the line of sight and the edge being observed as three constraints imposed on each guard (viewpoint). Their approach was first applied to a $2D$ polygonal workspace, then extended to a $3D$ polyhedral environment. However, the constraints they used are not complete or accurate to model the real-world sensors, and how to guarantee the full coverage of the guards (viewpoints) is not mentioned in their paper. Another disadvantage of their approach is that the local greedy nature of the randomized algorithm does not necessarily generate the optimal or even sub-optimal solution.

Kang *et al.* [45] provided a method to determine the minimum number of viewpoints from where all surfaces of the object are visible. An index theory was developed for the analysis and computation. Based on that, the geometric problem is transferred into a searching process with purely algebraic computations. Although their method does not restrict itself in $2D$, $3D$ inspection is not discussed in their paper. Additionally, only visibility is considered since the task constraint and the visibility region calculation is costly, especially in high dimensions. However, what inspires us is that Kang *et al.* pointed out the importance of the partition, or discretization, of surfaces

into patch elements, which eventually renders the problem as an integer optimization problem on a finite set.

Kay and Luo [42] raised the minimal number of camera problem in their automatic guided vehicle (AGV) system. They proved the NP-hardness of the camera placement problem, which is essentially a set-covering problem. However, in their system, the potential camera positions are limited to predetermined locations and only a $2D$ flat floor is considered in their paper.

For most of the parts commonly seen in automotive manufacturing, the surfaces are usually in free-from shape, or they are $3D$ surfaces. As far as we know, no viewpoint optimization work has been reported on free-from surfaces with consideration of all the four task constraints: visibility, field of view, resolution and focus.

In the previous chapter, the recursive algorithm generates viewpoints by dividing a large flat patch into two halves. This algorithm, although easy to implement, does not imply the minimal number of viewpoints. Hence a new strategy is needed to find the minimum set of viewpoints. Here we propose an approach that generates candidate viewpoints in a constructive way and solves a set-covering problem to obtain the minimum viewpoints.

After the FPAG-based surface decomposition, all the flat patches are obtained. Since big curvatures occur at borders where different flat patches meet, it is sufficient to consider each flat patch individually. More generally, the minimum viewpoint planning problem can be stated as follows,

given a large flat patch which has no big curvature, find a minimum set of viewpoints that can inspect any point on the flat patch with the following four task constraints satisfied: visibility, field of view, resolution and focus.

In the following sections, the set-partitioning problem is introduced first; then the constructive candidate viewpoint generation method is described; after that, we present the implementation and experimental results.

3.1.2 Set-Partitioning Problem

Set-partitioning problem, and its generalized version, set-covering problem, are special structures in integer programming [46]. They can model many industrial scheduling and planning problems like bus crew scheduling, air crew scheduling, facility location etc. These problems can be formulated as follows,

given 1) a finite set M with m members; 2) a constraint set defining a family F of n 'acceptable' subsets of M; find a minimum collection of members of F which is a partition or a cover of M [46].

Mathematically, the set-partitioning problem can be formulated as [42],

$$minimize\ z = \sum_{j=1}^{n} x_j \tag{3.1}$$

$$subject\ to \sum_{j=1}^{n} a_{ij}x_j = 1,\ i = 1, 2, ..., m \tag{3.2}$$

$$x_j = 0\ or\ 1,\ j = 1, 2, ..., n. \tag{3.3}$$

Where $a_{ij} = 1$ means that member i belongs to the subset j. $x_j = 1$ means that subset j is selected and $x_j = 0$ otherwise.

To model the minimum viewpoint planning problem as a set-partitioning problem, we need to

1) partition the flat patch into finite cells $c_1, c_2, ...c_m$.

2) obtain finite viewpoints, $V_1, V_2, ...V_n$, with each covering, or inspecting a subset of all the cells.

Essentially, both the flat patch and the camera configuration space need to be discretized. Once the minimum viewpoint problem is mapped to the set-partitioning problem, $a_{ij} = 1$ means that viewpoint V_j can cover, or inspect cell c_i and $x_j = 1$ means that viewpoint V_j is selected as one member of the minimum set.

The set-partitioning problem has been studied extensively for many years. Generally there exist two types of algorithms: exact algorithms and heuristic algorithms. The former attempt to solve the set-partitioning problem to optimality while the latter try to find approximate solutions quickly.

One type of exact algorithm solved the problem based on cutting planes [47]. Another type of exact algorithm uses tree search (Branch-and-Bound). Various bounding strategies, including linear programming and Lagrangian relaxation, have been exploited. These include the algorithm developed by Marsten [48] and an improved version of this algorithm by Marsten and Shepardson [49]. Other exact algorithms are found in: [50, 51, 52].

There have been relatively few heuristic solution algorithms for the set-partitioning problem. Ryan and Falkner [53] provided a method of obtaining a good feasible solution by imposing additional structure, derived from the real-world problems but not already implicit in the mathematical model. Atamturk *et al.* [54] developed a heuristic algorithm incorporating problem reduction, linear programming, cutting planes and a Lagrangian dual procedure. Recently, some parallel genetic algorithms have appeared [55, 56]. Optimal and sub-optimal solutions can be obtained for many problems of even large size using genetic algorithms.

3.1.3 Candidate Viewpoint Construction

It is well-known that both the set-partitioning and the set-covering problem are NP-hard [46], which means that the size of the problem will bring difficulty to finding the optimal or even sub-optimal solution. A straightforward way is to reduce the size of the problem, that is to reduce the number of surface cells and the number of candidate viewpoints.

Square cells are used to partition the flat patch. By determining suitable dimensions of the square cell, the number of surface cells can be controlled to keep the size

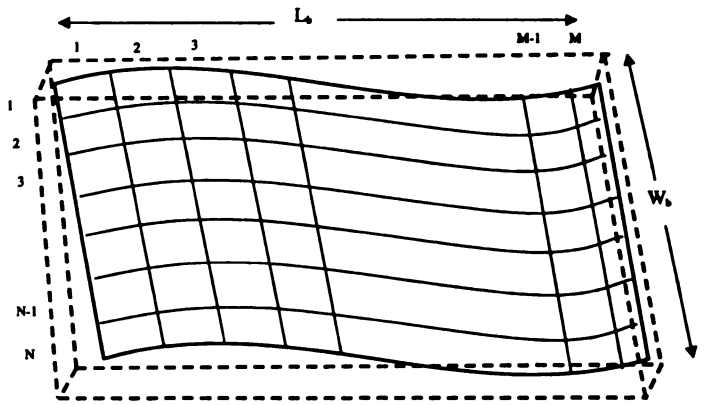

Figure 3.1: Cell-partitioning of a large flat patch.

of the set-partitioning problem moderate.

Since the dimension of the camera configuration space, or degree of freedom, considered in our system is 6, an extremely huge number of viewpoints will be generated if we use small rectangloids to decompose the camera configuration space, a scheme usually used in robot motion planning [57]. On the other hand, it is not necessary to enumerate all those candidate viewpoints since most of them are very close to each other or do not correspond to admissible viewpoints. Hence a method is needed to construct the candidate viewpoints such that they

1) distribute uniformly in the entire camera configuration space,

2) are sufficiently distinct from each other, and

3) all correspond to admissible viewpoints.

Therefore, it is necessary to obtain a certain understanding of the geometric structure of the viewpoint distribution around the flat patch. Here we develop a method, which, based on the geometric structure of the flat patch, generates all the candidate viewpoints. In this way, the number of candidate viewpoints is greatly reduced and a certain degree of redundancy is maintained.

Cell-Partitioning of Flat Patch

As shown in Figure 3.1, the procedures to partition a flat patch are:

1) construct a minimum-area bounding box for the flat patch, denote the length and width of the bounding box as L_b and W_b.

2) build parallel planes along the length and the width direction to form grids, the in-between distance in both directions is D_{in},

3) partition the small triangles into $N \times M$ cells based on the grids. Here $N = \lceil \frac{W_b}{D_{in}} \rceil$, $M = \lceil \frac{L_b}{D_{in}} \rceil$, where $\lceil x \rceil$ denotes the nearest integer that is greater or equal to x. Denote each cell as c_{ij}, $1 \leq i \leq N$, $1 \leq j \leq M$.

Here the minimum-area bounding box is constructed in a similar way as discussed in Chapter 2 except that the top and right direction are determined so that the front rectangle has the minimum area. Each cell c_{ij} can also be assigned an index number which can be calculated as $i \cdot M + j$.

Cell Combining

The strategy we use to construct candidate viewpoints is as follows: first, the neighboring cells are combined to from subpatches of different sizes, then the corresponding viewpoints for these subpatches are found. There are different ways to combine neighboring cells and they result in a sufficient number of distinct viewpoints.

It is apparent that the maximum area a camera can cover on a plane is the maximum field of view, which is determined by the depth of field distance D_1, or the resolution constraint, whichever is smaller. It is easy to understand that the depth of field determines a maximum field of view, which is corresponding to D_1. Figure 3.2 illustrates how the resolution constraint determines another maximum field of view. For the resolution constraint to be satisfied, the resolution ball of a point on the inspected surface should include the viewpoint O, which gives the farthest plane from the viewpoint as is shown in the figure.

Suppose the dimension of the largest rectangular field of view is $W_f \times L_f$, $(L_f \geq W_f)$. Similarly, it can be partitioned into $N_f \times M_f$ square cells of dimension $D_{in} \times D_{in}$,

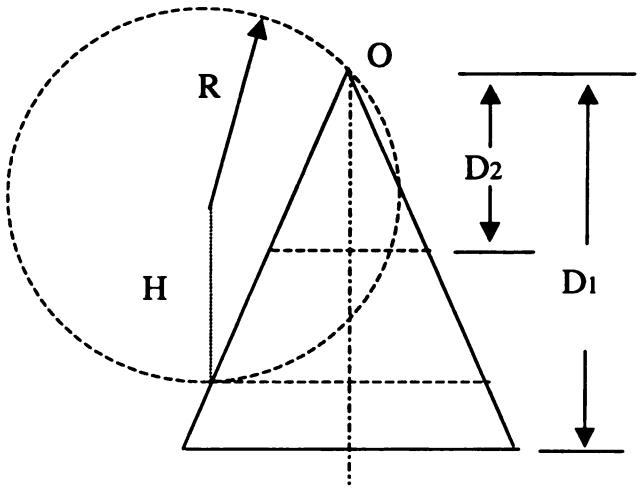

Figure 3.2: The resolution constraint and the field of view.

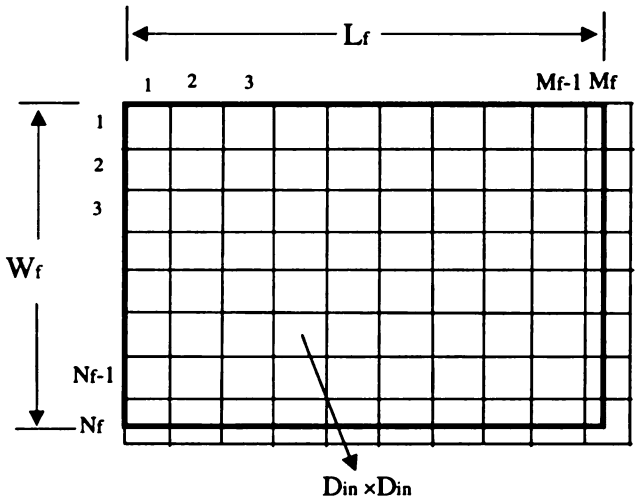

Figure 3.3: Partition the maximum field of view.

as is shown in Figure 3.3. N_f and M_f are determined as follows,

$$N_f = \lceil \frac{W_f}{D_{in}} \rceil \quad M_f = \lceil \frac{L_f}{D_{in}} \rceil.$$

Obviously, $M_f \geq N_f$.

To discuss cell combining, the following definition is introduced first.

Definition 3.1.1 (generating pattern) *A* **generating pattern,** *Pat(i,j), is a cell matrix with i rows and j columns in the maximum field of view.*

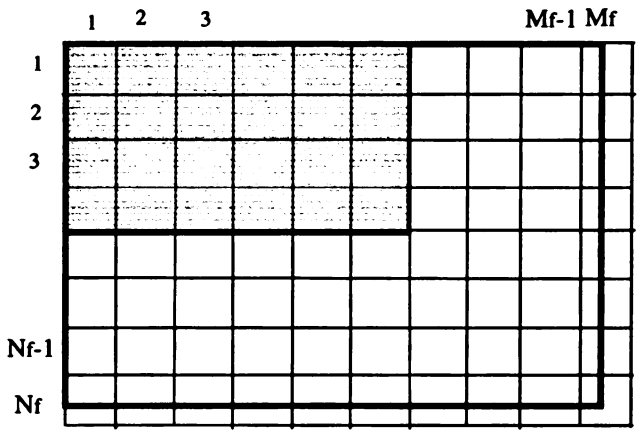

Figure 3.4: A sample generating pattern.

Essentially, the generating patterns are designed to group neighboring cells on a flat patch into subpatch that has a potential viewpoint. Figure 3.4 shows a pattern with 4 rows and 6 columns, or, $Pat(4,6)$. Obviously, the total number of generating patterns is,

$$N_{gp} = N_f M_f$$

Definition 3.1.2 (generating subpatch) *A* **generating subpatch** *is a surface area on the flat patch, which is a collection of surface cells that conform to one of the generating patterns.*

Any generating pattern can be mapped to a number of generating subpatches on the flat patch. For generating pattern $Pat(i,j)$, there are $(N-i+1) \times (M-j+1)$ different locations to map it on the flat patch. As a result, there are the same number of generating subpatches. Usually the field of view is not a square (or $L_f > W_f$), the direction of the mapping should be considered. For those generating patterns $Pat(i,j)$ with $j > N_f$, they can be mapped along both the row direction and the column direction, each resulting in different generating subpatches. The number of those generating patterns is $N_f \times (M_f - N_f)$. Hence, the total number of generating subpatches can be calculated as follows,

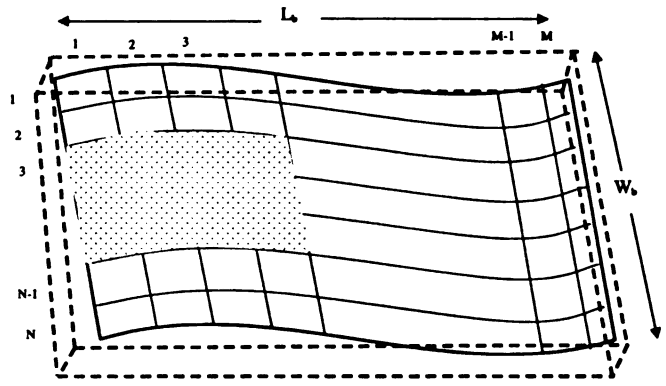

Figure 3.5: Map a generating pattern to get a generating subpatch.

$$N_{gs} = \sum_{i,j}^{1\le i\le N_f, 1\le j\le M_f} (N-i+1)(M-j+1) + \sum_{i,j}^{1\le i\le N_f, N_f+1\le j\le M_f} (N-j+1)(M-i+1)$$

In Figure 3.5, a generating pattern $Pat(3,4)$ is mapped to the flat patch and a generating subpatch is obtained.

Candidate Viewpoint Generation

For any generating subpatch GS_i, $(i \in \{1,2,...,N_{gs}\})$, the viewpoint planning algorithm developed in Chapter 2 is applied. What is different from the recursive algorithm is that the subpatch is not split if no viewpoint exists. The algorithm is shown in Figure 3.6. Suppose among the N_{gs} generating subpatches there are N_s generating subpatches each having at least one admissible viewpoint. These viewpoints are numbered from V_1 to V_{N_s}. For each viewpoint V_j, an index set S_j is assigned to it, which records the indices of all the cells that is in the corresponding subpatch, i.e.,

$$S_j = \{i_1, i_2, ..., i_{n_j}\} \quad (i = 1,2,...,N_s).$$

It should be pointed out that, for every single cell, it is reasonable to assume that there exists at least one viewpoint. The reason resides in the following fact,

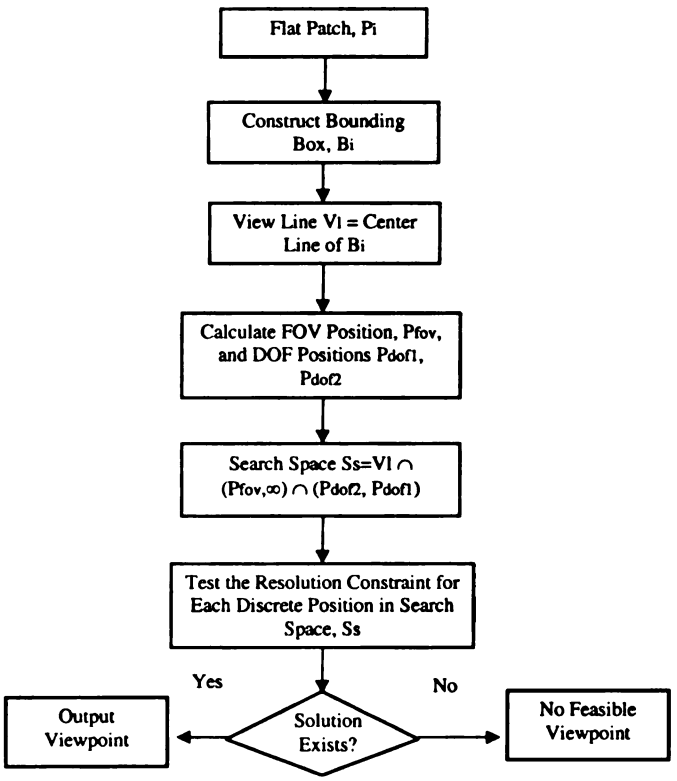

Figure 3.6: Candidate viewpoint generation algorithm.

the closer a surface area approaches the upper depth-of-field plane, or the plane corresponding to D_2, the more curvature this surface area can tolerate for it to be inspected without violating all the task constraints.

This fact is illustrated in $2D$ in Figure 3.7. Obviously, both the visibility and the field of view constraint are trivial, so only the focus and the resolution constraint need to be considered. Furthermore, the surface area can be assumed to be within the depth of field, we only need to consider the resolution constraint. As can be seen in the figure, for surface area that is closer to the upper depth of field plane, the resolution ball is much more likely to include the viewpoint O than the surface area that is far away from the upper depth of field plane.

Hence, it is reasonable to say that as long as the size of the cell is carefully chosen, each cell can have at least one feasible viewpoint.

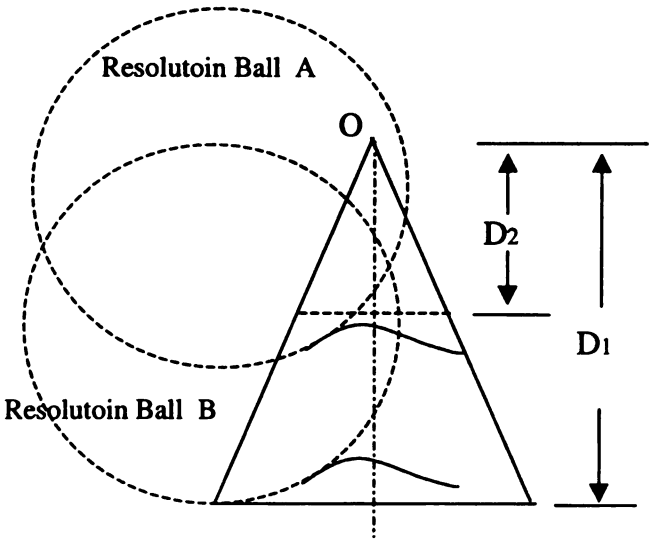

Figure 3.7: Curvature tolerance with respect to the viewpoint-surface distance.

3.1.4 Formulation of Minimum Viewpoint Planning Problem

Now the minimum viewpoint planning problem can be converted to a set-partitioning problem, which can be stated as follows,

given a viewpoint set $\{V_1, V_2, ..., V_{N_s}\}$, V_j *corresponds to an index set* $S_j = \{i_1, i_2, ..., i_{n_j}\}$, *find out the minimum set of, say,* N_m *viewpoints from* $\{V_1, V_2, ..., V_{N_s}\}$ *such that the union of the corresponding index sets* $\cup_{j=1}^{N_m} S_j = \{1, 2, ..., NM\}$.

Mathematically, the minimum viewpoint problem can be stated as follows:

$$minimize\ z = \sum_{j=1}^{n} x_j \tag{3.4}$$

$$subject\ to \sum_{j=1}^{n} a_{ij} x_j = 1,\ i = 1, 2, ..., N \times M \tag{3.5}$$

$$x_j = 0\ or\ 1,\ j = 1, 2, ..., N_s. \tag{3.6}$$

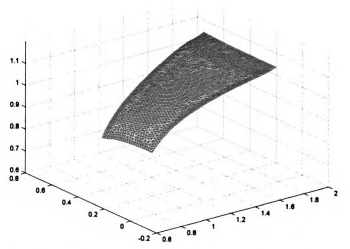

Figure 3.8: The tessellation representation of a flat patch from a car hood.

3.1.5 Implementation and Results

Implementation

The minimum viewpoint planning approach is implemented in two steps. First, the candidate viewpoints are generated. This step is implemented in C++. Second, the set-partitioning problem is solved to get the minimum set of viewpoints. This step is solved in a commercial optimization software *IPLOG Studio*^{TM} [58], which can solve various integer optimization problems.

Results

The approach is tested on sample patches generated from $I-DEAS^{TM}6.0$ and sheet metal parts from Ford Motor Company as well. The first sample flat patch is a patch from a car hood. Its dimensions are about $0.97m \times 0.58m$. Its tessellation representation is shown in Figure 3.8. The total number of triangle is 4286.

The maximum field of view of the camera is $0.3m \times 0.4m$. The grid size $D_{in} = 0.10m$. Hence, $N = 6$, $M = 10$, $N_f = 3$, and $M_f = 4$. The flat patch is partitioned into $10 \times 6 = 60$ cells. Based on the above data, the number of generating subpatch is 591. The program generates 275 candidate viewpoints totally, which are taken

as the input to the optimization software *IPLOG Studio*^{TM}. The total number of viewpoints that can cover the whole patch is 5, which are shown in Figure 3.9. The frames indicate the covering area of each viewpoint.

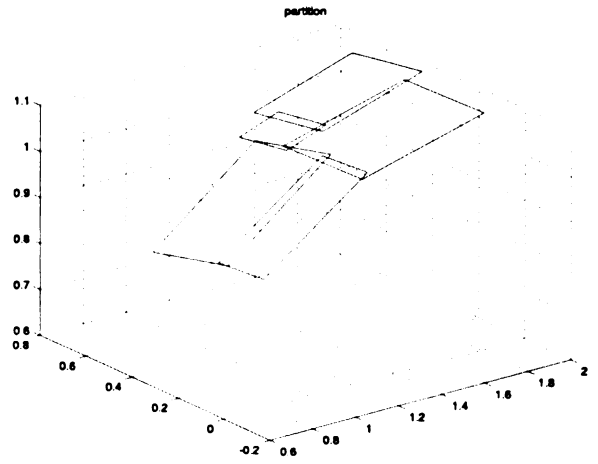

Figure 3.9: The covering area of the minimum viewpoints.

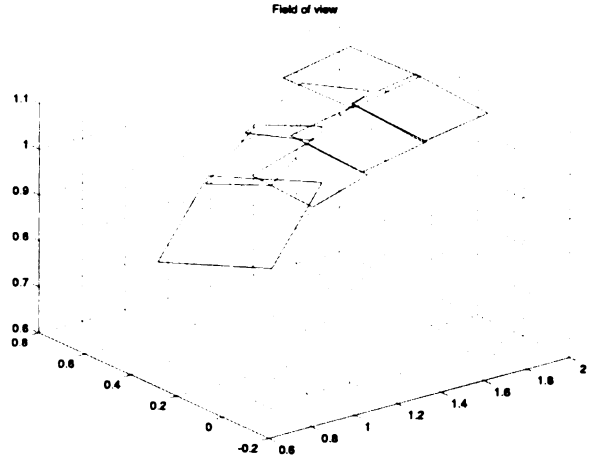

Figure 3.10: The covering area of the viewpoints generated by the recursive algorithm.

As a comparison, the recursive algorithm is run on the same patch. The total number of viewpoints is 7, which are shown in Figure 3.10.

Tests on other sheet metal parts from Ford Motor Company show that the number of viewpoints can be reduced significantly. Table 3.1 summarizes the testing results.

Table 3.1: Testing results on different parts

Parts	No. Triangle	No. Viewpoints (Recursive)	No. Viewpoints (Optimized)
Door	8235	85	64
Pillar	7820	32	23
Fender	4512	16	12

3.2 Optimal Kinematics Performance Sensor Planning

3.2.1 Background

Kinematics constraint is an important issue usually considered in robot motion planning. Constraints like reachability of the robot to task points is the least requirement for a task to be carried out. There also exist other quantitative measures that evaluate the specific performance of the robot at a task point.

However, to deal with kinematics issues, most of the research in robot motion planning takes them as two separate problems. In the first problem, the positions and orientations of the robot end-effector are generated to satisfy given task constraints. In the second problem, an optimal robot placement problem, or equivalently, an optimal task point placement problem is solved to obtain a suitable relative position and orientation of the robot with respect to the task points so that all the task points can be reached with satisfactory kinematics performance measures. A typical robot placement problem is shown in Figure 3.11, where a suitable robot base position/orientation is calculated to make the task points reachable.

The first problem is usually task specific, while the second problem is general and has received much attention recently. In [59], Hsu *et al.* presents an efficient algorithm for optimizing the base location of a robot manipulator in an environment cluttered with obstacles. In Nelson *et al.*'s work [60], using the concept of manipulability, a rating is given to different placements of the assembly task in the workspace so that a

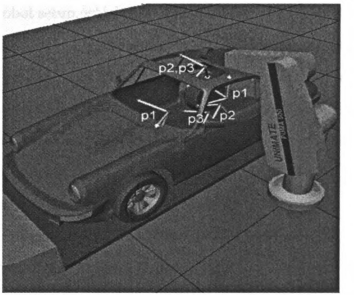

Figure 3.11: A typical robot placement problem.

robot's ability to manipulate objects can be objectively compared at various positions and orientations. To find the optimal placement a search procedure is carried out. Pamanes *et al.* [61] solved a similar optimal placement problem. Instead of using a single kinematics criterion, they used multiple criteria. Seraji [62] addressed the problem of base placement for mobile robots using a reachability analysis.

However, solving the robot base placement or the task points placement problem requires high computational effort in searching for an optimal or sub-optimal solution, and as a common headache in many nonlinear optimization problems, the search usually ends up in local minima or maxima.

3.2.2 Optimal Kinematics Performance Sensor Planning

We take a different view on kinematics issues in sensor planning. Instead of solving the two separate problems, we integrate the kinematics constraint into the sensor planning so that the generated sensor configurations are reachable by the robot and also have high kinematics performance measures. In other words, the kinematics is taken as an additional constraint in sensor planning.

Given a part/robot setup, which is usually determined based on human experience and/or physical restriction of fixtures, it is unrealistic to presume that any generated viewpoint will be reachable for the robot. However, as discussed in the first part of this chapter, there are plenty of ways to inspect a flat patch. Each way corresponds to a particular subset of the candidate viewpoint set. This provides a kind of redundancy to allow certain subsets of the viewpoints to fall in the workspace of the robot and meanwhile, for each of those subsets the full coverage of the flat patch is guaranteed. Furthermore, to choose which subset of viewpoints, another criteria, the kinematics performance measure, is taken into consideration. In this sense, the problem is rendered as a weighted set-covering problem so as to find the subset of viewpoints with optimal kinematics performance.

In the following sections, the weighted set-covering problem and its algorithms are reviewed first. Then the reachability check is discussed, followed by the introduction of kinematics performance measures. Finally, the implementation and testing results are provided.

3.2.3 Weighted Set-Covering Problem

The mathematical formulation of the weighted set-covering problem is,

$$minimize\ z = \sum_{j=1}^{n} c_j x_j \tag{3.7}$$

$$subject\ to \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, 2, ..., m \tag{3.8}$$

$$x_j = 0\ or\ 1, j = 1, 2, ..., n \tag{3.9}$$

Here c_j is the cost of subset j. The optimal solution results in a minimum cost cover of the whole set.

Similar to the set-partitioning problem, the set-covering problem is NP-hard [47]. Existing exact algorithms include cutting planes [63], column subtraction [51], etc. Some of the approximation algorithms are: a simple greedy algorithm by Chvatal [64]; a Lagrangian heuristic algorithm by Beasley [65]; a randomized rounding algorithm by Peleg *et al.* [66]. Other algorithms use neural network [67], genetic algorithms [68, 69] and simulated annealing [70].

3.2.4 Reachability Check

To check if the robot can position the camera to a given viewpoint, the first step is to find, for a given viewpoint, the corresponding robot hand position and orientation in the robot base coordinate system. This involves the transformation from a viewpoint in part coordinate system to a hand position and orientation in the robot base coordinate system. How to find the transformation is the calibration problem which will be discussed in detail in Chapter 5. In this chapter we assume the transformation is known.

Suppose for each viewpoint V_i, the corresponding Cartesian position and orientation of the end-effector, or hand, of the robot is in the form of a 4 by 4 homogeneous transformation matrix ${}^0A_{Ei}$ $(i = 1, 2, ..., N_s)$. For a 6-joint non-redundant manipulator, the inverse kinematics gives out,

$$Q_i = f({}^0A_{Ei}) \quad i = 1, 2, ..., N_s, \tag{3.10}$$

where $Q_i = [\theta_{1i}, \theta_{2i}, ..., \theta_{6i}]^T$ is the ith joint angle vector. If equation(3.10) has no feasible solution, it implies that the given viewpoint can not be reached. If there are feasible solutions, it is checked against the joint range to see if the solution falls in the joint range, that is,

$$\theta_{j\ min} \leq \theta_{ji} \leq \theta_{j\ max}\ \forall j = 1, 2, ...6.$$

If any joint angle is out of the corresponding joint range, which means it violates the physical constraints of the robot mechanism, the viewpoint can not be reached either.

3.2.5 Kinematics Performance Measure of Sensor Configuration

It is known that the kinematics performance, or the comfortableness, of a robot manipulator during the achievement of a task depends on the relative position and orientation of manipulator/task [61]. An ill relative position and orientation takes the risk of inefficient operation even as falling in singularities.

In literature, there are several kinematics performance measures that have been proposed and used in different optimal placement problems [61]. Some of them are:

1. Manipulability
2. Remoteness of the manipulator links from their joint limits
3. Condition number
4. Compatibility index

The manipulability is introduced by Yoshikawa [71] and it is quantified by $w = \sqrt{det(JJ^T)}$, where J is the Jacobian matrix of the manipulator. The manipulability is a measure of the ease to arbitrarily change the position and orientation of the end effector and it depends on the kinematics parameters and consequently on the instantaneous configuration of the manipulator defined by a joint vector. To optimize the manipulability, w should be maximized. The remoteness of the manipulator links from their joint limits is usually adopted to keep the joint angles away from their

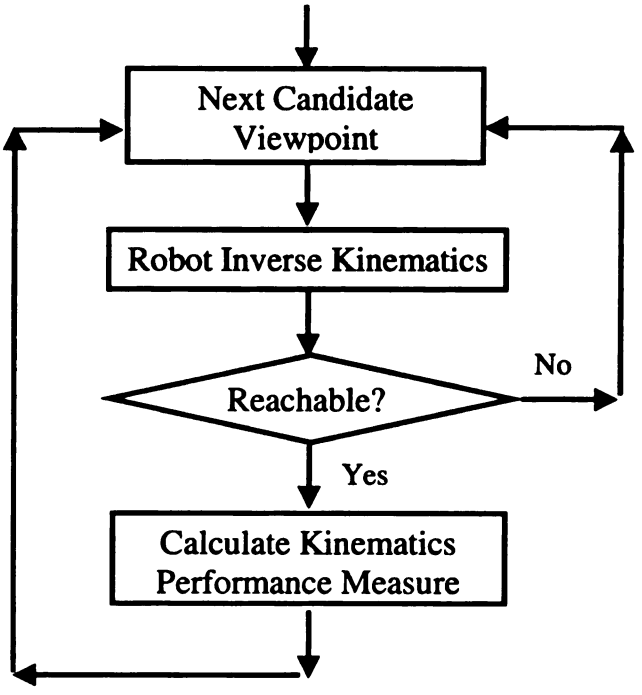

Figure 3.12: Kinematics performance calculation.

limits as much as possible [72]. The condition number of the Jacobian transpose matrix J^T of the manipulator is given by $C(J^T) = \| J^T \| \| J^{-T} \|$. The condition number can be used to minimize the error propagation from input torques to output forces. The compatibility index [73] allows to optimize the magnitude and accuracy of force and velocity of the manipulator on preferred displacement directions.

3.2.6 Formulation of Optimal Kinematics Performance Sensor Planning Problem

For part inspection, it is desirable that the manipulator is kept far away from singularities when it reaches viewpoints. Hence the manipulability is chosen as the kinematics performance measure to be optimized. The algorithm to calculate the kinematics performance measure of each viewpoint is shown in Figure 3.12.

As the kinematics performance measure is assigned to each viewpoint as a cost, the optimization problem can be rendered as a weighted set-covering problem, which

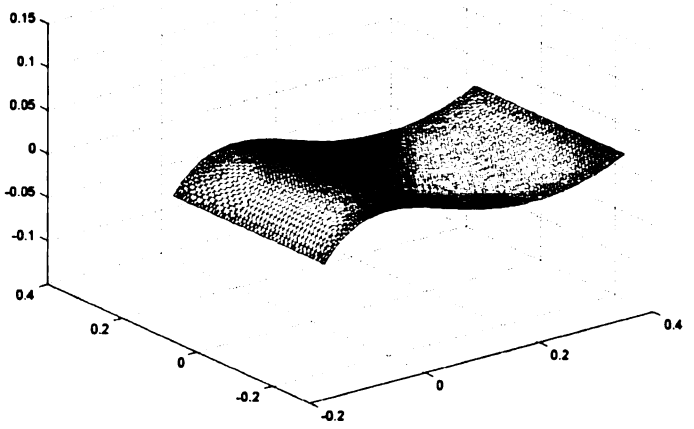

Figure 3.13: The tessellation representation of the flat patch.

is stated as follows,

$$minimize\ z = \sum_{j=1}^{N_s} c_j x_j \tag{3.11}$$

$$c_j = (\sqrt{det(J_j J_j^T)})^{-1} \tag{3.12}$$

$$subject\ to \sum_{j=1}^{N_s} a_{ij} x_j \geq 1, i = 1, 2, ..., N \times M \tag{3.13}$$

$$x_j = 0\ or\ 1, j = 1, 2, ..., N_s \tag{3.14}$$

Here c_j is the cost of each viewpoint. Since (3.11) achieves minimum, instead of the manipulability, its inverse is taken as the cost, i.e.,

The optimal solution implies a minimum-cost cover of the whole flat patch.

3.2.7 Implementation and Results

The optimal kinematics performance sensor planning approach is implemented and tested. The eye-in-hand setup is the same as we discussed in Chapter 2. The first sample flat patch is a free-form surface built in $I - DEAS^{TM}$6.0. Its dimensions are about $0.42m \times 0.53m$. Its tessellation representation is shown in Figure 3.13. The

total number of triangle is 4900.

The maximum field of view of the camera is $0.3m \times 0.4m$. The grid size $D_{in} = 0.10m$. Hence, $N_f = 3$ and $M_f = 4$. The flat patch is partitioned into $5 \times 6 = 30$ cells. The program generates 235 candidate viewpoints totally.

The *IPLOG Studio*^{TM} optimization software is used to solve the weighted set-covering problem. First, we run the software without kinematics constraint. That is, all the candidate viewpoints have the same weight. Figure 3.14 shows the field of views of the 4 viewpoints. Then we run the software with the inverse of the manipulability as the cost. Figure 3.15 shows the 4 camera field of views obtained.

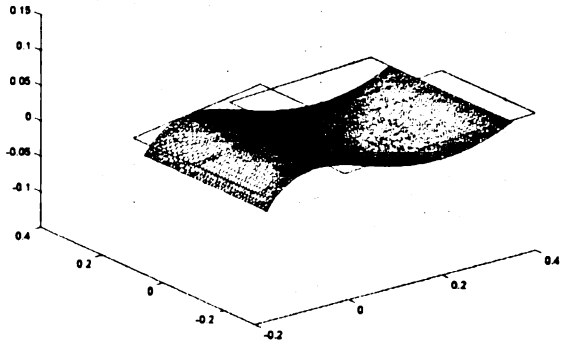

Figure 3.14: The field of views when no kinematics constraint is considered.

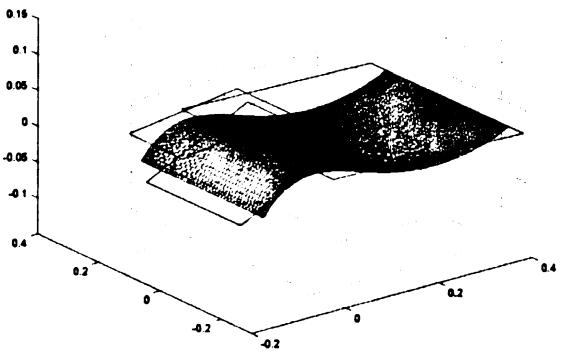

Figure 3.15: The field of views when kinematics constraint is considered.

Table 3.2 gives out the cost (or w_j^{-1}) of each viewpoint as well as the sum of the

Table 3.2: Cost of viewpoint

Cases	V1	V2	V3	V4	z
Constraint	1.35	2.65	3.26	3.70	10.96
No constraint	250.5	1000.0	10.26	1000.0	2260.76

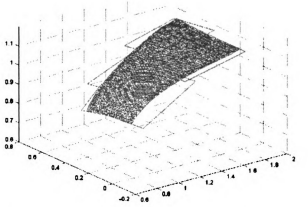

Figure 3.16: The field of views with maximum kinematics performance measure.

cost in the above two cases. In this table, a cost of 1000.0 means there is no inverse kinematics to the corresponding viewpoint. A big value means the viewpoint is close to the singularity. From this table it is obvious that the viewpoints found by the algorithm with kinematics constraint have low cost, or high kinematics performance measures.

Sheet metal parts from Ford Motor Company are also tested. Figure 3.8 shows a flat patch from a car hood. Figure 3.16 shows the 5 field of views with the maximum kinematics performance measure and Table 3.3 lists the cost of viewpoints.

Simulations of the robot movement are done in $ROBCAD^{TM}$. Figure 3.17 shows the simulation setup for the first sample patch, where the frames represent the posi-

Table 3.3: Cost of viewpoint

Cases	V1	V2	V3	V4	V5	z
Constraint	1.76	3.88	3.72	4.21	3.70	17.27
No constraint	1000.0	4.56	1000.0	15.78	6.33	2026.67

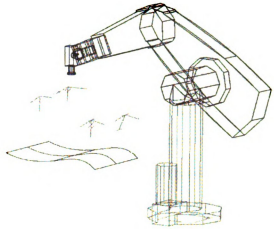

Figure 3.17: The simulation in ROBCAD for the sample patch.

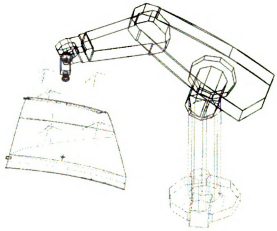

Figure 3.18: The simulation in ROBCAD for the patch from the car hood.

tions and orientations of the selected viewpoints that have the maximum total kinematics performance measure. It is found that all the 4 viewpoints can be reached by the *Puma*560 easily. Figure 3.18 shows the simulation setup for the flat patch from the car hood and the 5 viewpoints can be reached easily too.

3.3 Discussions

In this chapter, two optimization problems are addressed. One is the minimum viewpoint problem. By discretizing the flat patch and the sensor configuration space, the problem is converted to an integer optimization problem, or a pure algebraic

problem. A constructive viewpoint generation method is used to obtain a number of representative viewpoints distributed in the sensor configuration space.

The other is the kinematics performance optimization problem. By modeling it as a weighted set-covering problem, we integrate the kinematics constraint into sensor planning. Depending on different task requirements, the weight of the sensor configuration can be different from the one we used.

In their nature, both problems sample the camera configuration space to generate sufficient candidate configurations with each covering a finite set of surface cells. In this way, the geometric problems can be converted into algebraic problems. Mathematically, both algebraic problems are $NP-hard$, which implies that the size of the cell is an important parameter to control the size of both optimization problems. The value depends on the size of the flat patch, the camera capability (such as resolution and focus), etc. In other words, the cell size should not be too small considering the size of the set-partitioning/covering problem and not too big considering the resolution and focus capability of the camera for each surface cell.

CHAPTER 4

NEAR-OPTIMAL-TIME PATH PLANNING FOR INSPECTION

In this chapter, how to plan the robot path among the viewpoints is studied. The criteria used in path planning is the overall traveling distance among viewpoints. Under certain assumptions, the problem is rendered as a Traveling Salesman Problem (TSP) on a complete graph with straight-line distance as the weight of the edges.

After reviewing previous related work on TSP and its variants, we propose a hierarchical approach to solve the TSP quickly to its sub-optimality. The approach has two steps: first, a large number of viewpoints are clustered into several groups with each group containing a moderate number of viewpoints. Second, a clustered Traveling Salesman Problem (CTSP) is solved. A new algorithm is developed to solve the CTSP. Different from some previous work, the inter-group path is preferred in our new algorithm.

4.1 Background

For a given part, after all the viewpoints are generated, the path planning problem then follows: find the minimum-time movement of the robot to carry out the inspection. It is a reasonable assumption that the camera completely stops at each viewpoint and the time to execute an inspection at all viewpoints is the same, i.e., all equal to a constant time. Obviously, the time for a complete inspection of the whole part consists of 1) the time spent on the traverse among the viewpoints and 2) the time spent on the execution of inspection at all the viewpoints. Since the latter is constant, the optimization of the total inspection time is, essentially, the minimization of the time spent on the traverse of the camera among viewpoints. Mainly, two factors determine the traverse time, 1) the trajectory, or time history of joint

positions, velocities, accelerations, and torques, between each pair of viewpoints; 2) the order to visit all the viewpoints. In this dissertation only the second factor is addressed. To begin with, we briefly review some previous work on these two factors.

4.1.1 Related Work

For a robotic system, the minimum-time geometric path of the robot's end-effector is not necessarily equivalent to the path of minimum Euclidean distance, but depends on the manipulator's kinematics and its inertial parameters [74]. Furthermore, considering the dynamics of the robot manipulator, the geometric path planning and velocity trajectory should be considered simultaneously to achieve a minimum solution [74]. Shin and Mckay [75] developed a method that provides a complete solution to the global optimization problem by solving both the geometric path planning and velocity trajectory planning. Based on some assumptions, their study has shown that the near-minimum-time path of the robot manipulator is a geodesic in the manipulator's inertia space.

Based on Shin and Mckay's algorithm, Edan *et al.* developed a computationally efficient algorithm that allows the determination of the near-minimum-time path between n task points. They modeled the problem as a Traveling Salesman Problem (TSP) and took the distance between two points as the length of the geodesic path, which reflects the time to travel between the two points. A Nearest Neighbor algorithm was adopted to solve the TSP. To reduce the computation time, they divided the task points into several groups and solved the TSP group by group. However, they did not talk about how to find the paths connecting different groups, which will be a nontrivial problem that need to be addressed as the number of groups increases.

4.1.2 Traveling Salesman Problem and Algorithms

The Traveling Salesman Problem can model many scheduling and planning problems in real-world applications. It is stated as follows,

if a salesman, starting from his home city, is to visit exactly once each city on a given list and then return home, how to determine the order of the visit to all the cities such that the total distance traveled is minimum [76].

In computation complexity theory, the Traveling Salesman Problem is NP-complete [76].

Since the Traveling Salesman Problem was formally proposed in 1930's, there has been intensive research efforts in it [76]. Mainly, there are two type of algorithms to solve it.

1) Exact algorithms and

2) Approximation algorithms

The exact algorithms aim to find the optimal solutions, however, these algorithms are slow, especially for large size problems. Some of the exact algorithms are: cutting planes [77], Branch-and-Bound, [78] and dynamic programming [79]. The approximation algorithms try to find sub-optimal solutions using heuristic approaches and hence, they are much faster compared to the exact algorithms. Approximation algorithms can be categorized into: 1) construct algorithms and 2) improve algorithms. The former directly generate sub-optimal solutions while the latter find sub-optimal solutions by improving given initial solutions. Some of the construct algorithms are : Nearest Neighbor [80], Minimum Spanning Tree [81] and Christofides' algorithm [82]. The famous Lin-Kernighan algorithm [83] is one of the improve algorithms.

As variants of the Traveling Salesman Problem, the shortest Hamiltonian path problems try to find the shortest path that visits all the cities without returning to the starting city. There are three variants:

1) the shortest free-end Hamiltonian path (SFHP) problem – there is no specifi-

cation on the end cities,

2) the shortest one-end Hamiltonian path (SOHP) problem – one end city is specified, and

3) the shortest two-end Hamiltonian path (STHP) problem – both end cities are specified.

All the three Hamiltonian path problems can be transformed into classical Traveling Salesman Problems by enhancing the distance matrix [76]. They can also be solved directly using approximation algorithms. For the first two variants, namely, SFHP problem and SOHP problem, a straightforward adaptation of Christofides' algorithm yields an algorithm with a 1.5 performance bound [84], which measures the ratio of the total length of the solution returned by the approximation algorithm to the total length of the optimal solution. For the last variant, or STHP problem, there is an approximation algorithm with $\frac{5}{3}$ performance bound [84].

4.2 Problem Formulation

To model the robot path planning problem, a graph, $G = (V, E)$ is constructed in the following way: take each viewpoint as a vertex v_i, the path between two vertices v_i and v_j as an edge e_{ij} and the minimum distance to traverse between these two vertices as its cost, or weight, w_{ij}. The goal is:

find a shortest Hamiltonian path, which means that the robot visits all the vertices and does not go back to the starting vertex.

To simplify the problem, the following facts are assumed:

1) the Euclidean distance between two viewpoints corresponding to vertex v_i and v_j, $d(v_i, v_j)$, is taken as weight w_{ij}.

2) the graph is complete.

Under the first assumption, the incidence matrix of the problem is symmetric, that is,

$$w_{ij} = w_{ji}$$

and the triangle inequality is satisfied:

$$w_{ij} + w_{jk} \geq w_{ik}, \ \forall i, j, k$$

The second assumption means that there exists a straight path between any pair of viewpoints. This is usually true for automotive sheet metal parts since all viewpoints are most likely high above the part surfaces.

4.3 Hierarchical Approach to Solve Traveling Salesman Problem

It is apparent that the number of viewpoints will increase as the area of the surfaces of the part and the geometric complexity increase. Due to the nature of the Traveling Salesman Problem, the time to solve it grows rapidly as the problem size grows, even for approximation algorithms.

On the other hand, based on the generic shapes of the automotive sheet metal parts and the decomposition of compound surfaces, the resulting viewpoints tend to form groups in $3D$ space. That means the distances between viewpoints in the same group are relatively smaller than the distances between viewpoints among groups. This geometric structure of the viewpoints inspires us to solve the problem in a hierarchical way:

cluster the viewpoints based on their distance similarity into several groups; solve the TSP for each group, as well as determine the path to visit all the groups.

By taking advantage of the cluster nature of the problem, it is possible to find sub-optimal solutions very quickly. Following are some useful definitions when the hierarchical TSP is referred to.

Definition 4.3.1 (inter-group edge) *An* **inter-group edge** *is an edge* $e_{ij} = (v_i, v_j)$ *with its vertices* v_i *and* v_j *in different groups.*

Definition 4.3.2 (inter-group path) *The* **inter-group path** *is the collection of inter-group edges that visit all the groups.*

The shortest inter-group path is the shortest one among all the inter-group paths. In this sense, it is the shortest "connections" among all the groups. Apparently, if the number of the groups is N_g, the number of inter-group edges on the inter-group path is $N_g - 1$.

Definition 4.3.3 (intra-group Hamiltonian path) *The* **intra-group Hamiltonian path** *is a path that visits all the vertices exactly once in a group without leaving the group.*

Any overall path for N_g groups consists of an inter-group path and N_g intra-group Hamiltonian paths.

Due to the cluster nature of the viewpoints, it is expected that the final path solved by the hierarchical approach are close to the path obtained by solving the whole TSP as one group.

4.3.1 Related Work on Hierarchical Approach

Clustered Traveling Salesman Problem

Once the viewpoints are clustered into groups, the hierarchical Traveling Salesman Problem becomes the clustered Traveling Salesman Problem (CTSP) [85, 86]:

Let $G = (V, E)$ *be a complete graph with vertex set* V *and edge set* E*. The vertex set is partitioned into* k *groups,* $g_1, g_2, ..., g_k$*, determine a shortest path to visit all the vertices and the vertices of each group are visited consecutively.*

The clustered Traveling Salesman Problem has received far less attention than the Traveling Salesman Problem in research community [87]. Chisman [85] transformed the CTSP back to a TSP by adding big costs to the inter-group edges. However, his algorithm focuses on the group constraint instead of the computational cost. In this sense, his method does not reduce the computational cost of the original TSP. Lokin [86] provided a Branch-and-Bound algorithm to exactly solve the CTSP problem, and as can be expected, it is not time-efficient. Approximation algorithms were also developed. Among them are: three heuristic algorithms by Gendreau *et al.* [88] which, however, only consider the CTSP with predefined order of groups; the genetic algorithm by Potvin and Guertin [89]; A Tabu search heuristic using genetic diversification by Laporte *et al.* [87].

Recently, Guttmann-Beck *et al.* [84] proposed approximation algorithms with performance bounds for some variants of the CTSP. Their algorithms are based on a modified Christofides' algorithm for the shortest Hamiltonian path in each group, as well as another modified Christofides' algorithm to find a shortest path connecting all the groups. The latter utilizes the solution to a Rural Postman Problem, which tries to find a shortest path to visit a specified subset of the edges in the graph. Guttmann-Beck *et al.* considered three variants of the CTSP : 1) both the starting and ending vertices are given for each group; 2) only one end vertex is specified; 3) no end vertices are specified. For each variant, Guttmann-Beck *et al.* determined the shortest intra-group Hamiltonian paths first, then calculated the paths among groups. Furthermore, they provided the corresponding performance bounds for the three variants.

4.3.2 New Approach to Solve the CTSP

As discussed above, to solve the CTSP, Guttmann-Beck's algorithms favor the intra-group Hamiltonian paths, which implies that inter-group path may be sacrificed if

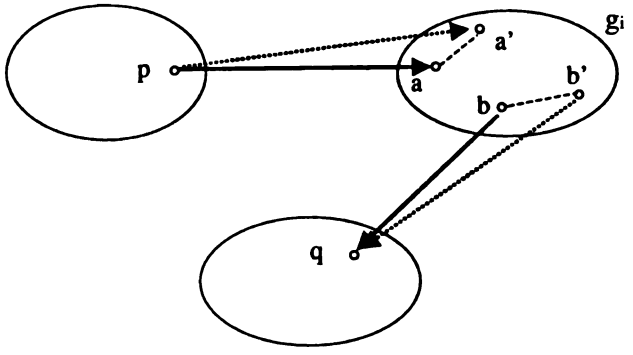

Figure 4.1: Estimate the penalty of Guttmann's algorithms.

the end vertices in each group are determined first.

To roughly estimate the penalty caused by favoring intra-group Hamiltonian paths, the following definition is introduced first,

Definition 4.3.4 (diameter of a group) *The* **diameter of a group,** $D(g)$, *is the maximum distance of all pairs of vertices in that group, i.e.,*

$$D(g) = \max_{v_i,\ v_j \in g} d(v_i, v_j).$$

For the case where intra-group Hamiltonian paths are favored, like Guttmann-Beck *et al.*'s algorithms, assume that the shortest inter-group path enters group g_i at a and leaves it at b (see Figure 4.1). While the shortest free-end Hamiltonian path (SFHP) of g_i takes a' and b' as the end vertices. Obviously, for group g_i, the penalty is less than 2 times of the diameter of the group, or $2D(g_i)$.

On the other hand, we have the following lemma,

Lemma 4.3.1 *Denote the total length of the shortest free-end Hamiltonian path of group g as $d(SFHP)$, denote the total length of the shortest two-end Hamiltonian path of the same group as $d(STHP)$, then the following inequality holds,*

$$d(STHP) \leq d(SFHP) + 2D(g)$$

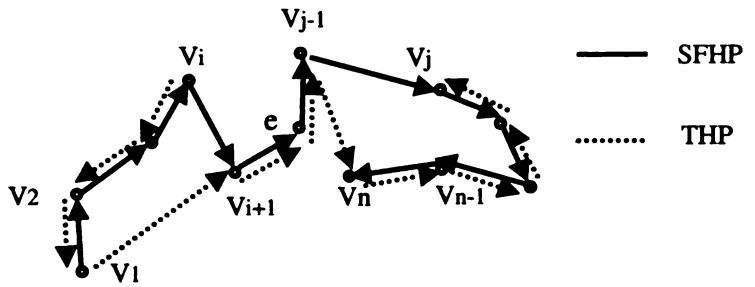

Figure 4.2: Construct a two-end Hamiltonian Path from a SFHP.

Here $D(g)$ is the diameter of group g.

Proof.

As shown in Figure 4.2, the n vertices are numbered sequentially from v_1 to v_n in the order of the shortest free-end Hamiltonian path ($SFHP$). Given any pair of starting and ending vertex, (v_i, v_j), the following method can construct a two-end Hamiltonian path.

1) v_i and v_j are adjacent Begin at v_i, trace back to v_1 in the reverse order of $SFHP$; travel directly from v_1 to v_n; then trace in the reverse order of $SFHP$ to v_j.

It is obvious that by adding an edge (v_1, v_n) and deleting an edge (v_{j-1}, v_j), a Hamiltonian path is obtained with given starting and ending vertices.

2) v_i and v_j are not adjacent.

Similarly, beginning at v_i and trace back to v_1 in the reverse order of $SFHP$; then travel directly from v_1 to v_{i+1}; on the $SFHP$ travel to v_{j-1}; directly connect v_{j-1} and v_n; trace in the reverse order of $SFHP$ to v_j.

By adding two edges, (v_1, v_{i+1}) and (v_{j-1}, v_n) and deleting two edges (v_i, v_{i+1}) and (v_{j-1}, v_j), a Hamiltonian path is obtained with given starting and ending vertices.

Hence, by adding at most two edges a $SFHP$ can be converted to a two-end Hamiltonian path with any given starting and ending vertices, which implies that the length of the $STHP$ of any pair of starting and ending vertices should be no more than the length of $SFHP$ plus 2 times of the diameter of the group, i.e.,

$$d(STHP) \leq d(SFHP) + 2D(g).$$

◁

The above discussions imply that the penalty caused by favoring either the inter-group path or the intra-group Hamiltonian paths, are comparable. This hints us to solve the CTSP in an alternative way which determines the inter-group path first.

In the following sections, the viewpoint clustering approach is discussed first, then the a new hierarchical algorithm, namely, the two-level-TSP Hierarchical algorithm is described, which is followed by the performance analysis. Finally, the implementation and results are provided.

4.3.3 Viewpoint Clustering

Cluster analysis is a mathematical tool widely used in statistics to explore the structure of a data set. It can be stated as follows,

given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that data points in one cluster are more similar to one another and data points in separate clusters are less similar to one another [90].

There exist many clustering algorithms. Basically, they can be divided into hierarchical clustering algorithms and non-hierarchical algorithms. The former include agglomerative method, the stored matrix method and the stored data method etc. The latter include MacQueen's k-Means method and its variants [90].

The requirements we consider to cluster the viewpoints are:

1) the number of viewpoints in each group should be moderate.

2) the number of groups should be moderate.

The above two requirements guarantee a reduced computational cost of the hierarchical approach to the TSP.

Based on the Nearest Neighbor clustering algorithm [91] we propose a new clustering algorithm which is described as follows,

1) begin with a small threshold t ($t > 0$) and the viewpoint set $\{V_1, V_2, ..., V_n\}$.

2) assign V_1 to group g_1; set $i = 1$, $k = 1$.

3) set $i = i + 1$; find the nearest neighbor of V_i among the viewpoints already assigned to groups.

4) let the nearest neighbor be in group m; if its distance is greater than t, increment k and assign V_i to a new group g_k, else assign V_i to g_m.

5) if $i < n$, go to step 3).

6) if the number of groups is greater than a threshold N_g, let $t = 2t$, redo step 2) to step 6); if the number of groups is less than N_g, let $s = \frac{t}{2N_{step}}$ and $t = \frac{t}{2}$, N_{step} is the number of steps.

7) $t = t + s$; redo 2) to 5); if the number of groups is less than N_g, go to 8); otherwise repeat 7).

8) recursively divide those groups with more than N_p viewpoints into equal size until all the groups has less than N_p viewpoints.

In the above algorithm, a typical Nearest Neighbor clustering algorithm is applied to obtain certain number of groups. A small t is used first so the number of resulting groups is expected to be big. Then t is doubled each time to rerun the Nearest Neighbor algorithm until the number of groups is smaller than the threshold N_g. To refine the number of the groups, t increases in small steps and the Nearest Neighbor algorithm is rerun. Finally, the groups with more than N_p viewpoints are split in halves recursively to make sure the numbers of viewpoint in these groups are less than N_p.

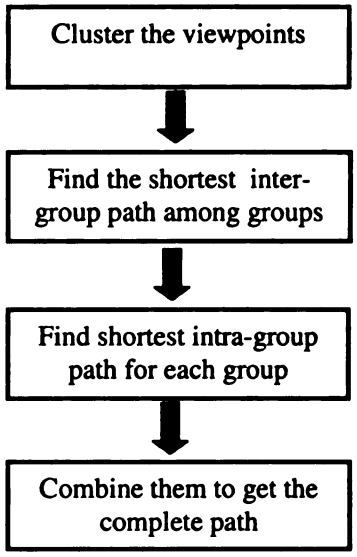

Figure 4.3: The two-level-TSP algorithm.

4.3.4 Two-level-TSP Algorithm

As implied in its name, the two-level-TSP algorithm constructs two TSPs in different levels. In the higher level, the shortest "connections", or the shortest inter-group path, among groups are determined. In the lower level, with the entering and the leaving vertex already known in each group, the shortest Hamiltonian path problem is solved group by group. By combining the paths generated in both levels, a complete path can be obtained. The two-level-TSP algorithm is shown in Figure 4.3.

In the low level, there exist different possibilities when the shortest inter-group path enter and leave each group, which will be discussed later.

Shortest Inter-group Path

To find the shortest inter-group path, a new graph, called *group graph*, is constructed with each vertex representing a group and the distance between two vertices representing the distance between the two corresponding groups. Here the distance between two groups is defined as follows,

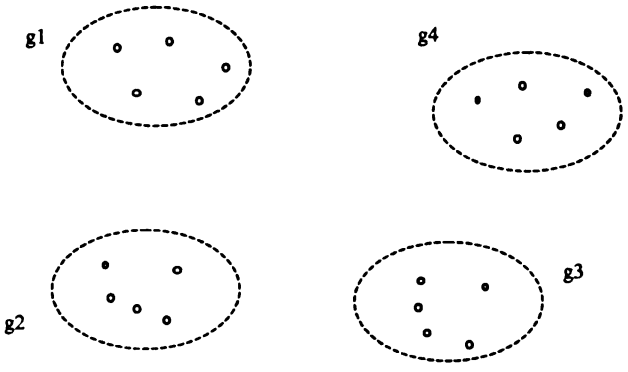

Figure 4.4: An example problem.

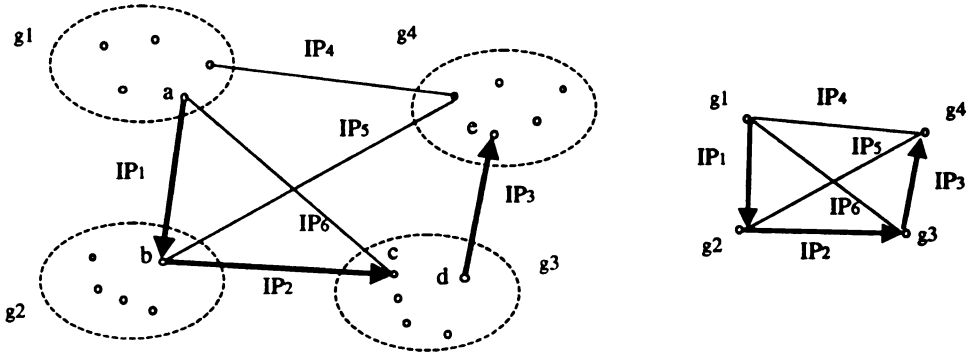

Figure 4.5: The distances between groups and the group graph.

Definition 4.3.5 (distance between two groups) *The* **distance between two groups,** $D(g_i, g_j)$, *is the minimum of the length of the inter-group edges between* g_i *and* g_j. *i.e.,*

$$D(g_i, g_j) = \min_{v_i \in g_i, v_j \in g_j} d(v_i, v_j).$$

Obviously, the group graph is complete.

An example problem is shown in Figure 4.4. The distances between groups and the corresponding group graph are shown in Figure 4.5, where $IP_1, IP_2, ...IP_6$ are the corresponding distances between groups.

The shortest free-end Hamiltonian path on the group graph is corresponding to the collection of the inter-group edges with minimum total distance that visit all the groups. In Figure 4.5, the thicker lines (IP_1, IP_2, IP_3) highlight the shortest inter-group path.

Shortest Intra-group Hamiltonian Paths

As far as the way the shortest inter-group path enters and leaves each group is concerned, There are four possibilities:

1) in the starting group of the shortest inter-group path, there is only one leaving vertex,

2) in the ending group of the shortest inter-group path, there is only one entering vertex,

3) in some intermediate groups, entering and leaving vertices are different.

4) in some intermediate groups, entering and leaving vertices coincide.

Figure 4.5 shows all the four cases, the shortest inter-group path (IP_1, IP_2, IP_3) begins in group 1 at vertex a and ends in group 4 at vertex e. In group 3, the entering vertex c and the leaving vertex d are different, while in group 2, the entering vertex and the leaving vertex coincide at b.

To deal with these four cases, correspondingly, there are four ways to find the shortest intra-group Hamiltonian paths.

1) The shortest one-end Hamiltonian path (SOHP) problem is solved to find the shortest intra-group Hamiltonian path that ends at the starting vertex of the inter-group path.

2) The shortest one-end Hamiltonian path (SOHP) problem is solved to find the shortest intra-group Hamiltonian path that starts at the ending vertex of the inter-group path.

3) A shortest two-end Hamiltonian path (STHP) is found which begins at the entering vertex and ends at the leaving vertex.

4) A shortest two-end Hamiltonian path (STHP) is found which begins at the entering vertex of the current group and ends at the entering vertex of the next group. As a result, in the overall path, the corresponding inter-group edge is replaced by the new edge joining the two consecutive groups.

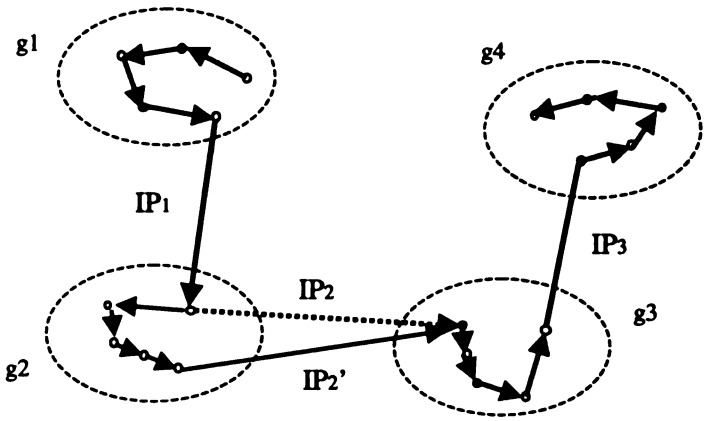

Figure 4.6: Find the intra-group Hamiltonian paths.

The four different types of intra-group Hamiltonian paths (or extended intra-group Hamiltonian paths as in the fourth case) are shown in Figure 4.6.

In the fourth case, the following lemma holds on the length of the shortest two-end Hamiltonian path that extends to the next group.

Lemma 4.3.2 *Denote the length of the shortest two-end Hamiltonian path that extends from group g_i to the next group g_j as $d(STHP_i')$, denote the length of the shortest Hamiltonian circle in group g_i as $d(SHC_i)$, IP_i is on the shortest inter-group path and it joins g_i with g_j, denote $d(IP_i)$ as its length, then*

$$d(STHP_i') \leq d(SHC_i) + d(IP_i) \tag{4.1}$$

Proof.

In Figure 4.7, assume SHC_i is the shortest Hamiltonian circle in group g_i. Delete the last edge of SHC_i, or e_p, which is the edge between the starting vertex s_i and the vertex preceding s_i on SHC_i. Connect the preceding vertex with the starting vertex s_j of the next group g_j by edge IP_i'. Thus an extended intra-group Hamiltonian path is constructed from s_i to s_j. Obviously, the length of this Hamiltonian path is no less

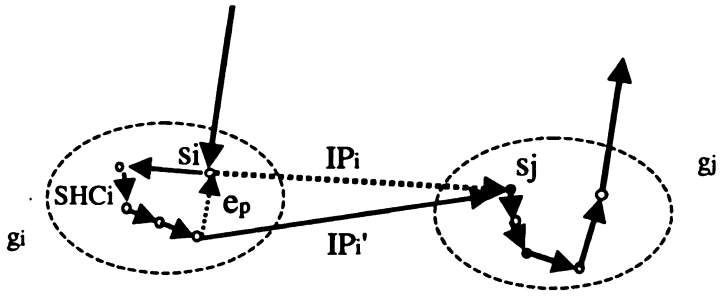

Figure 4.7: The relation between the shortest Hamiltonian circle and the shortest extended intra-group Hamiltonian path.

than $d(STHP_i')$. On the other hand, due to the triangle inequality, the length of this Hamiltonian path is less than $d(SHC_i) + d(IP_i)$. Therefore Equation(4.1) holds. $\triangleleft$

Performance Analysis

Here we analyze the performance of the two-level-TSP algorithm. That is, assuming both the shortest inter-group path problem and the shortest intra-group (or extended intra-group) Hamiltonian paths problem achieve optimal solutions, how far is the solution from the optimal solution of the CTSP?

Similarly to lemma 4.3.1, we have the following lemma,

Lemma 4.3.3 *Denote the total length of the shortest free-end Hamiltonian path of group g as $d(SFHP)$, denote the total length of the shortest one-end Hamiltonian path of the same group as $d(SOHP)$, then the following inequality holds,*

$$d(SOHP) \leq d(SFHP) + D(g)$$

The proof is similar to that of lemma 4.3.1. The difference is that only one edge needs to be added and another one be deleted to construct a one-end Hamiltonian path from the shortest free-end Hamiltonian path.

Denote the total length of the optimal solution of the CTSP as $d(OPT)$. Obviously, the following inequality holds,

$$d(OPT) \geq d(IP) + d(SFHP)$$

Here $d(IP) = \sum_{i=1}^{k-1} d(IP_i)$ is the total length of the shortest inter-group path solved based on the group graph. $d(SFHP) = \sum_{i=1}^{k} d(SFHP_i)$ is the sum of the length of the shortest free-end Hamiltonian paths in all groups. The equality holds only when in each group the two end vertices determined by the two level TSPs are the same.

Theorem 4.3.1 *Denote the total length of the overall path returned by the two-level-TSP algorithm as $d(TLT)$, then*

$$d(TLT) \leq d(OPT) + 2\sum_{i=1}^{k} D(g_i).$$

k is the number of groups.

Proof.

The overall path returned by TLT algorithm consists of the inter-group path and the intra-group paths.

For the starting and ending group where only one end vertex is given, as proved in lemma 4.3.3, $d(SOHP_i) \leq d(SFHP_i) + D(g_i)$.

For the groups with two different entering and leaving vertices, lemma 4.3.1 has shown that $d(STHP_i) \leq d(SFHP_i) + 2D(g_i)$.

For those shortest intra-group Hamiltonian paths that extend to the next group, as we have proved in lemma 4.1, $d(STHP_i') \leq d(SHC_i) + d(IP_i)$. While the length of the shortest Hamiltonian circle should be less than the length of the shortest free-end Hamiltonian path plus the diameter of the group, i.e., $d(SHC_i) \leq d(SFHP_i) + D(g_i)$, we have

$$d(STHP_i') \leq d(SFHP_i) + D(g_i) + d(IP_i).$$

It is obvious that

$$d(TLT) \leq \sum_{i=1}^{k} SFHP_i + 2\sum_{i=1}^{k} D(g_i) + \sum_{i=1}^{k-1} d(IP_i) \tag{4.2}$$

$$= d(SFHP) + d(IP) + 2\sum_{i=1}^{k} D(g_i) \tag{4.3}$$

$$\leq d(OPT) + 2\sum_{i=1}^{k} D(g_i). \tag{4.4}$$

◁

4.4 Implementation and Results

4.4.1 Implementation

The modified Nearest Neighbor clustering algorithm and the TLT algorithm are implemented in C. The shortest inter-group path, or the high level of the TSP is solved using a simulated annealing method based on a minimum spanning tree [92]. For the SFHP, SOHP and STHP, a Christofides-like algorithm is adopted [93], which, essentially, is based on minimum spanning tree and minimum matching. The computing platform is $Unix^{TM}$ with a *Ultraspark* 1 167 Mhz CPU and 512 M RAM.

Here a brief description of Christofides' algorithm and its modification by Hoogeveen on Hamiltonian paths problems is provided.

Christofides' Algorithm

Christofides' algorithm consists of three steps [76].

Step 1), construct a minimum spanning tree T on the set of all the points I.

Step 2), construct a minimum matching M^* for the set of all odd-degree vertices

in T.

Step 3), find an Eulerian tour for the Eulerian that is the union of T and M^*, and convert it to a tour using shortcuts.

Hoogeveen's Algorithm for Hamiltonian Paths

Hoogeveen provided the following modified algorithm for finding shortest Hamiltonian paths with 0, 1 or 2 points fixed [93]. The algorithm consists of 4 steps.

Step 1), construct a minimum spanning tree T of the Graph G.

Step 2), first, determine the set S of vertices that are of wrong degree in T, i.e., the collection of fixed endpoints of even degree and other vertices of odd degree. Next, construct a minimum matching M on S that leaves $2-k$ vertices exposed, where k is the number of fixed endpoints. Such a matching can be found by constructing a minimum matching on S augmented with $2-k$ dummy vertices in an obvious fashion.

Step 3), consider the graph that is the union of T and M. This graph is connected and has either two or zero odd-degree vertices. The latter case occurs only if there is a single fixed endpoint that belongs to S and is left exposed by M; in this case delete an arbitrary edge incident to this vertex. Find an Eulerian path in the resulting graph. This path traverses each edge exactly once and has the two odd-degree vertices as its end-points.

Step 4), transform the Eulerian path into a Hamiltonian path by applying shortcuts.

4.4.2 Testing Results

The TLT algorithm is tested on several sets of points. We compare the TLT algorithm with SFHP algorithm that runs on the whole set of 2D points (we call it one-level-TSP (OLT) algorithm).

First, to get some general idea on the performance of the TLT and OLT algorithm, a 2D testing data is designed which is shown in Figure 4.8.

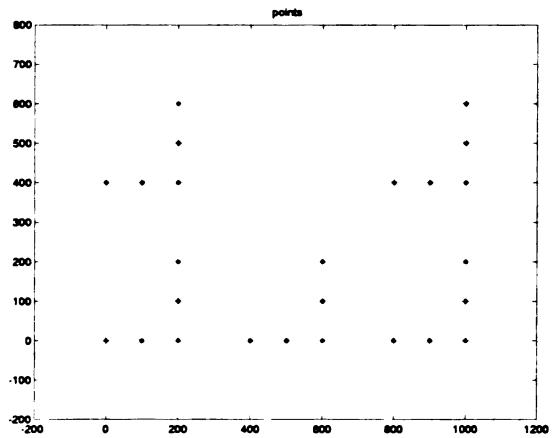

Figure 4.8: The testing 2D points for TLT and OLT algorithm.

Using the TLT algorithm, the total length of the final path is 3412.9 m. The computing time on this set of 2D data is 2.4 seconds. Using the OLT algorithm, the total length of the final path is 3842.2 m and the computing time is 4.2 seconds. Figure 4.9 and Figure 4.10 show the final paths from TLT and OLT algorithm respectively.

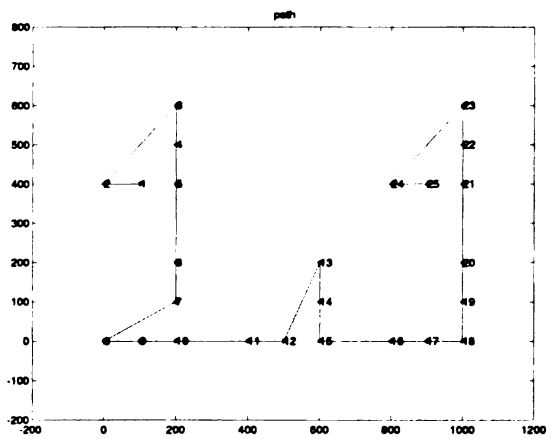

Figure 4.9: The final path from the TLT algorithm.

Then we test the algorithm on large number of random 3D points. The length of an arbitrarily-ordered initial path, the final best path length and the computing time for each set of points are shown in Table 4.1.

In the above table, it is clear that both optimization algorithms are effective by comparing the length of an arbitrarily-ordered initial path and the optimized path. It

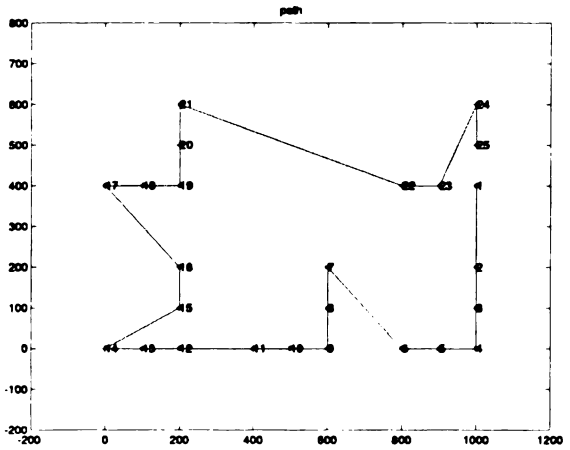

Figure 4.10: The final path from the OLT algorithm.

Table 4.1: Testing results on random point sets

	No. of Point	Initial Length (m)	Length (m)		Time (s)	
			OLT	TLT	OLT	TLT
set 1	300	599.6	100.7	133.3	62.3	4.4
set 2	400	1089.1	169.5	211.5	97.2	5.2
set 3	500	1670.3	236.4	308.1	123.2	7.5
set 4	800	4221.8	531.1	656.2	216.2	8.4

is also clear that the TLT algorithm is much faster than the one-level-TSP algorithm. This advantage becomes apparent when the problem size grows. The total lengths of the paths are comparable for both algorithms.

Then, several automotive parts from Ford Motor Company are used in testing the algorithm. First, the viewpoints that satisfy the task constraints are generated. Second, these viewpoints are clustered into multiple groups by the modified nearest neighbor algorithm. Third, by using the TLT algorithm, a complete path is obtained. As an example, a floor pan is used, which is shown in Figure 4.11. Totally 510 viewpoints are generated due to its large size and detailed geometry. An arbitrarily ordered initial path gives a length of 124.97 m. These viewpoints are clustered into 46 groups. The generated path is shown in Figure 4.12, where the stars denote the position of the viewpoints. The total length of the path is 59.96 m and the computing

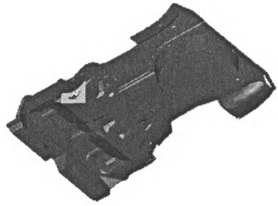

Figure 4.11: A floor pan.

time is 17.35 s. The overall path obtained by OLT is shown in Figure 4.13 with a path length of 45.26 m and computing time 264.25 s.

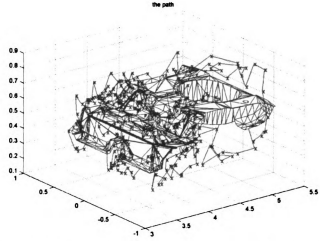

Figure 4.12: The sub-optimal path by the TLT algorithm.

Table 4.2 summarizes the performance (computing time and path length) of the two-level-TSP and the one-level-TSP algorithm on different parts.

4.5 Discussions

In this chapter, the robot path planning problem is solved so that the inspection can be carried out time-efficiently. By modeling the problem as a TSP, a hierarchical

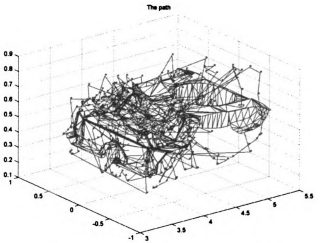

Figure 4.13: The sub-optimal path by the OLT algorithm.

Table 4.2: Testing results on different parts

	No. of Point	Initial Length(m)	Length (m)		Time (s)	
			OLT	TLT	OLT	TLT
pillar	55	35.22	16.58	17.62	17.50	1.82
door	103	97.45	35.28	36.78	37.02	6.26
floorpan	510	124.97	45.26	59.96	264.25	17.35

approach is proposed and a new algorithm is developed which can find approximate solutions quickly. Performance bounds of the algorithm is provided. This algorithm favors the highly clustered TSPs.

Some assumptions are taken to set up the TSP model. The complete graph assumption, however, may not reflect the reality because the surfaces of the part will become obstacles between a pair of viewpoints and no straight path exists between them. In this sense, the graph is not complete. Hence, there is need to investigate the hierarchical approach on non-complete graph. One way to do that is to put virtual edges with very big distance for those pairs without a straight path, thus the virtual edges make the graph complete. However, the triangle inequality does not hold on this graph and further investigation is needed to see if the hierarchical approach can be modified to work on this graph.

CHAPTER 5

IMPLEMENTATION

This chapter focuses on issues in experimental implementation. An eye-in-hand setup is used to realize the viewpoints and it requires that two calibration problems be solved: the robot localization and the hand/eye calibration. After reviewing existing methods for these two problems, we provide a new, simple method to solve the hand/eye calibration problem. Experimental evaluation of viewpoints is reported in the last section.

5.1 Implementation on Eye-in-Hand Setup

5.1.1 System Setup

To demonstrate the inspection system and evaluate the generated camera viewpoints, an eye-in-hand robot is set up in the Robotics and Automation Lab. The setup includes: a Puma560 manipulator and its controller which is implemented on real-time operating system $QNX4.25$; a Hitachi $KPD-50$ CCD camera, which is mounted on the Puma560; a Kineticsystems' Vibraplane optical table which is used to fix the part to be inspected; a monitor and a frame grabber (*Matrox Meteor*); a PC with Okino's $Nugraf^{TM}$ [94], which can import a CAD file and render it with a given camera viewpoint. The system setup is shown in Figure 5.1.

The viewpoint evaluation procedure is as follows: given an ordered sequence of viewpoints, the robot manipulator is controlled to reach each viewpoint. A picture is taken at each viewpoint and it is compared with the rendered image by $NuGraf^{TM}$ under the corresponding viewpoint.

Figure 5.1: The system setup.

5.1.2 Calibration Problems

For the eye-in-hand robot to execute inspection tasks, it is necessary to give out commands specifying where the robot hand should move to, or in other words, the position and orientation of the robot hand in the robot base coordinate frame have to be explicitly specified. However, what we generate in the vision sensor planner is the camera positions and orientations in the part coordinate frame. Thus a basic problem arises: how can we find the position and orientation of the robot hand in the base coordinate frame for a given position and orientation of the camera in the part coordinate frame?

Denote the robot base coordinate frame as $\{\mathbf{B}\}$, the hand coordinate frame as $\{\mathbf{H}\}$, the camera coordinate frame as $\{\mathbf{C}\}$, the optical table coordinate frame as $\{\mathbf{W}\}$ and the part coordinate frame as $\{\mathbf{P}\}$. All these frames are illustrated in Figure 5.2. The above coordinate transformation problem can be broken down into two sub-problems [3] :

- the robot localization problem and
- the hand/eye calibration problem.

The robot localization problem tries to solve the transformation between the robot

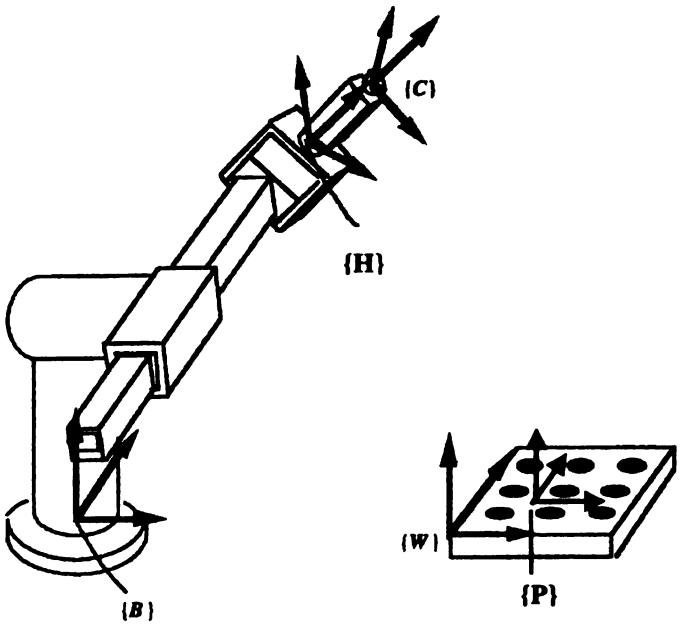

Figure 5.2: The coordinate frames (from [3]).

base coordinate frame {**B**} and a fixed world frame assigned somewhere, which, in our eye-in-hand setup, is the coordinate frame attached to the table, {**W**}.

The hand/eye calibration is the process of identifying the fixed yet unknown position and orientation of the inspection sensor, say, camera, with respect to the robot hand coordinate frame [3]. In our system, it is the transformation between the hand coordinate frame {**H**} and the camera coordinate frame {**C**}.

The transformation between the part coordinate frame {**P**} and the table coordinate frame {**W**} can be easily determined and it depends on the setup of the part on the table.

5.1.3 Robot Localization Problem

Techniques to solve the robot localization problem can be classified into those using pose measurements and those using only position measurements [3]. However, due to the fact that orientation measurements are usually expensive to obtain, we adopt the technique that uses only position measurements.

The localization procedure is as follows: select a number of points with known $3D$ coordinate in the world coordinate frame {W} (it is easy to achieve since the optical table has regular holes on it); direct the robot hand tip to approach these points with different known heights and record the joint angles of the robot; use forward kinematics to find the $3D$ coordinate of these points in the robot base coordinate frame.

Let ${}^wT_b$ and ${}^wT_h$ be the homogeneous transformations relating the robot base to the world frame and the hand to the world frame, respectively. Let ${}^wR_h$ and ${}^wp_h$ be the orientation matrix and the position vector of ${}^wT_h$, respectively. Similarly for ${}^wR_b$, ${}^wp_b$, ${}^bR_h$ and ${}^bp_h$. The following equation then holds:

$$ {}^wp_h = {}^wR_b \, {}^bp_h + {}^wp_b \tag{5.1} $$

Since ${}^bp_h$ is the function of the robot joint variables and link parameters, what we needed to do is estimate ${}^wR_b$ and ${}^wp_b$ provided that ${}^bp_h$ and ${}^wp_h$ are given at a number of robot configurations. As discussed above, the position vector ${}^wp_h$ is measured by an end-effector position measuring device and ${}^bp_h$ is computed by using the robot forward kinematics solver.

For simplicity and generality, the localization problem can be formulated as follows [3],

given m $3D$ points ${}^1p_i$ and m $3D$ points ${}^2p_i$, $(i = 1, 2, ..., m)$ in two coordinate frames, estimate the rotation R and the translation t that transform ${}^1p_i$ to ${}^2p_i$, i.e.,

$$ {}^2p_i = R \, {}^1p_i + t \ , i = 1, 2, ..., m. \tag{5.2} $$

There are several existing techniques to solve equation (5.2). A quaternion-based method [3] is adopted in our work. Here we give out the solution to R and t.

$$R = \begin{bmatrix} k_x{}^2 v\theta + c\theta & k_x k_y v\theta - k_z s\theta & k_x k_z v\theta + k_y s\theta \\ k_x k_y v\theta + k_z s\theta & k_y{}^2 v\theta + c\theta & k_y k_z v\theta - k_x s\theta \\ k_x k_z v\theta - k_y s\theta & k_y k_z v\theta + k_x s\theta & k_z{}^2 v\theta + c\theta \end{bmatrix}$$

where $v\theta = 1 - cos(\theta)$, $c\theta = cos(\theta)$, $s\theta = sin(\theta)$

Step 4), after R is solved, t can be solved as follows,

$$t = {}^2 p_i - R\ {}^1 p_i \quad i = 1, 2, ..., m$$

the least square solution for t is

$$t = \frac{1}{m} \sum_{i=1}^{m} \left[{}^2 p_i - R\ {}^1 p_i \right]$$

5.1.4 Hand/Eye Calibration Problem

Existing Method

One solution to the hand/eye calibration problem can be found in [95]. This method is to move the robot at least twice, each by a known amount, and to observe the resulting sensor motion induced by the robot motion (see Figure 5.3).

Let $\mathbf{T_6}$ be the transformation between $\{\mathbf{H}\}$ and $\{\mathbf{B}\}$, and OBJ be the transformation representing the object in the camera coordinate frame (see Figure 5.4). How to find OBJ is called the camera calibration problem, which we will address in more detail later.

The following equation holds:

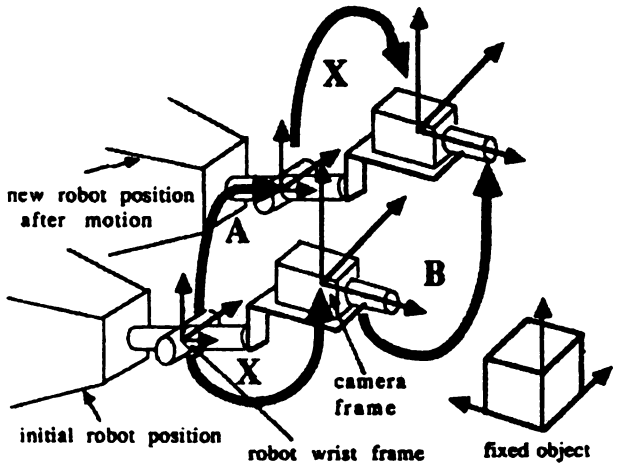

Figure 5.3: The hand/eye calibration (from [3]).

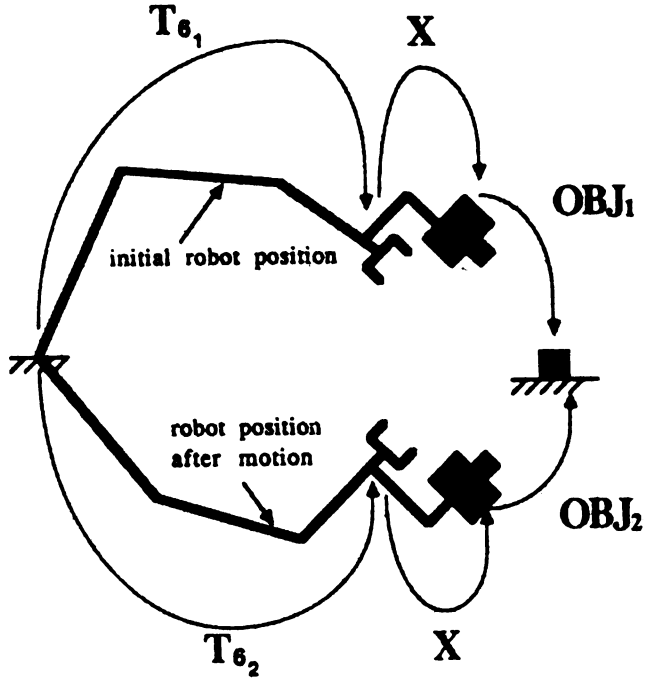

Figure 5.4: The formulation of the hand/eye calibration problem (from [3]).

$$\mathbf{T}_{61}\mathbf{X}OBJ_1 = \mathbf{T}_{62}\mathbf{X}OBJ_2 \tag{5.3}$$

The above equation can be written as

$$\mathbf{T}_{62}^{-1}\mathbf{T}_{61}\mathbf{X} = \mathbf{X}OBJ_2OBJ_1^{-1} \tag{5.4}$$

By defining $\mathbf{A} \equiv \mathbf{T}_{62}^{-1}\mathbf{T}_{61}$ and $\mathbf{B} \equiv OBJ_2OBJ_1^{-1}$, one can have the following homogeneous transformation equation,

$$\mathbf{AX} = \mathbf{XB} \tag{5.5}$$

where $\mathbf{X}$ is the 4×4 transformation matrix from the robot hand coordinate frame $\{\mathbf{H}\}$ to the camera coordinate frame $\{\mathbf{C}\}$.

To solve equation (5.5), there exist two approaches [3]. One approach first obtains the rotation part of the unknown transformation matrix and then determines the translation part of it, this approach is based on quaternion algebra. Another approach uses nonlinear iterative algorithm.

Our Method

We propose a new method to solve the hand/eye calibration problem, which does not need move the robot as in the previous method and hence, it is more straightforward and simpler.

It is noticed that after solving the robot localization problem, the transformation ${}^wT_b$ between the robot base coordinate frame $\{\mathbf{B}\}$ and the table coordinate frame

{W} is known. The transformation from the camera coordinate frame {C} to the robot base coordinate frame {B} can be obtained as follows,

$$^{b}T_{c} = {}^{w}T_{b}^{-1} OBJ^{-1}. \tag{5.6}$$

On the other hand, we can easily obtain the transformation $^{b}T_{h}$ between the hand coordinate frame {H} and the robot base coordinate frame {B} by solving the forward kinematics. So a fixed hand/eye transformation can be obtained as follows,

$$^{h}T_{c} = {}^{b}T_{h}^{-1}\ {}^{b}T_{c}. \tag{5.7}$$

Camera Calibration

Our method and the existing method both need the transformation between the world coordinate frame {W} and the camera coordinate frame {C}, which, in literature, is called the camera calibration problem [3]. Essentially, the camera calibration problem is to identify the set of extrinsic parameters (position and orientation of the center of the camera lens, or the perspective project center, in the world coordinate frame) and intrinsic parameters (focal length, scale factors, distortion coefficients, etc.) of the camera using a set of points known in both frames.

There exist rich literatures on camera calibration techniques. In the remainder of this section, some basic concepts of the camera calibration problem are introduced, followed by existing approaches to solve the calibration problem.

Camera model used in calibration

In camera calibration, people usually use either distortion-free lens model or lens distortion model. The distortion-free lens model, essentially, is the pinhole model

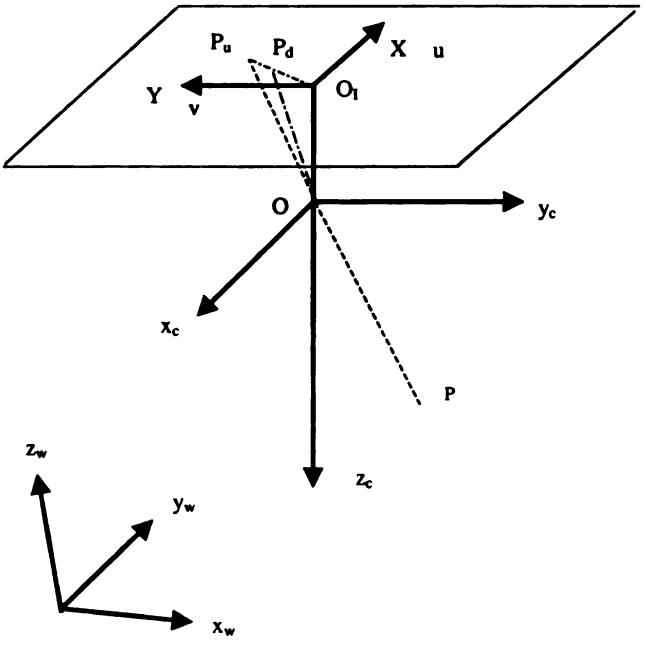

Figure 5.5: The distortion-free and distortion model.

where every object point is connected to its corresponding image point through a straight line that passes through the perspective center of the camera lens. The lens distortion model is a more practical model for most of the "off-the-shelf" lenses which sustain a variety of aberrations and do not obey a perfect model. Lens distortion effects can be classified into radial and tangential distortions. Radial distortion, which causes an inward displacement of a given image point from its ideal location is the dominant distortion effect. In Figure 5.5, both models are illustrated. In this figure, $\{x_w, y_w, z_w\}$ is the world coordinate frame. $\{x_c, y_c, z_c\}$ is the camera coordinate frame and $\{X, Y\}$ or $\{u, v\}$ is the $2D$ coordinate frame on the image plane. O is the perspective center of the camera. P_u is the distortion-free image point while P_d is the point with radial distortion.

A question naturally arises: does this pinhole model in calibration conflict with the general thick lens model we adopt for vision sensor planning? The answer is "no". In fact, when the front principle point (FPP) is taken as the perspective center, we can transform the general thick lens model into its equivalent pinhole model which is

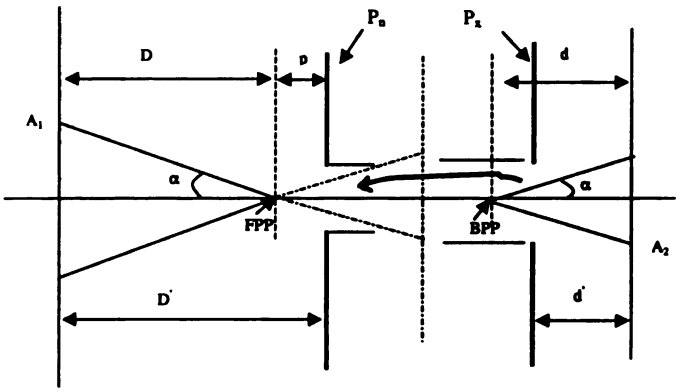

Figure 5.6: Transform the general thick lens model into pinhole model.

illustrated in Figure 5.6. By a translation (see Figure 5.6), the back principle point (BPP) coincides with the front principle point (FPP), so the resulting geometry of imaging is equivalent to the ideal pinhole model.

Formulation of the problem

Now the camera calibration problem can be stated as follows: given a certain number of $3D$ points in $\{\mathbf{W}\}$, and their corresponding image points in $\{\mathbf{C}\}$, find the transformation matrix ${}^{w}T_{c}$ between $\{\mathbf{W}\}$ and $\{\mathbf{C}\}$.

Since the optical table has regular grid points on its top plane, it can be taken as a calibration board. The camera calibration procedure is as follows,

Step 1), control the robot to move the camera to a suitable configuration. At this configuration, a certain number of grid points, or calibration points, are clearly visible in the camera. Save the picture.

Step 2), from the picture, calculate the X and Y value of the specified calibration point in the image plane. A good way to find the image coordinate of these points is: a) binarize the color image to get a black and white image, b) calculate the center of the blobs by averaging the X and Y coordinates. An alternative and easy way to find the coordinate of these calibration points is to use image software and directly read out the coordinate values by pointing the cursor to the center of each blob. A sample image of calibration points is shown in Figure 5.7.

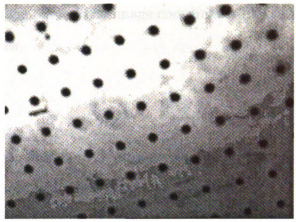

Figure 5.7: The calibration points.

Let a point in $\{\mathbf{W}\}$ be denoted as $[x_w, y_w, z_w]^T$ and a point in $\{\mathbf{C}\}$ be denoted as $[x_c, y_c, z_c]^T$, the transformation between them is

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = {}^c R_w \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + {}^c t_w \tag{5.8}$$

Where the rotation matrix ${}^cR_w$ and the translation vector ${}^ct_w$ are written as

$${}^cR_w = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}$$

$${}^ct_w = [t_x \; t_y \; t_z]^T$$

Based on the perspective project, we have

$$\mu = f\frac{x}{z} \quad \nu = f\frac{y}{z}$$

where f $(= d)$ is the (effective) focal length of the camera and (μ, ν) are the analog

coordinates in the image plane. The image coordinates (X, Y) are related to $(\mu, \nu)$by the following equations,

$$X = s_\mu \mu \quad Y = s_\nu \nu$$

The scale factors, s_μ and s_ν perform unit conversion from the camera coordinates (μ, ν) which is measured in meters, to image coordinates (X, Y), which is measured in pixels.

Define

$$f_x \equiv f s_\mu \quad f_y \equiv f s_\nu$$

and combine equation (5.8), we have

$$X = f_x \frac{r_1 x_w + r_2 y_w + r_3 z_w + t_x}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \tag{5.9}$$

$$Y = f_y \frac{r_4 x_w + r_5 y_w + r_6 z_w + t_y}{r_7 x_w + r_8 y_w + r_9 z_w + t_z} \tag{5.10}$$

The above two equations can be further rewritten as

$$X = \frac{a_{11} x_w + a_{12} y_w + a_{13} z_w + a_{14}}{a_{31} x_w + a_{32} y_w + a_{33} z_w + a_{34}} \tag{5.11}$$

$$Y = \frac{a_{21} x_w + a_{22} y_w + a_{23} z_w + a_{24}}{a_{31} x_w + a_{32} y_w + a_{33} z_w + a_{34}} \tag{5.12}$$

The coefficients $a_{11}, ..., a_{34}$ correspond to what is called the "Perspective Transformation Matrix". We can set $a_{34} = 1$ since the scaling of the coefficients $a_{11}, ..., a_{34}$ does not change the values of X and Y. So equation (5.11) and (5.12) can be combined into the following model:

$$\begin{bmatrix} x_w & y_w & z_w & 1 & 0 & 0 & 0 & 0 & -Xx_w & -Xy_w & -Xz_w \\ 0 & 0 & 0 & 0 & x_w & y_w & z_w & 1 & -Yx_w & -Yy_w & -Yz_w \end{bmatrix} \begin{bmatrix} a_{11} \\ . \\ . \\ . \\ a_{33} \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} \tag{5.13}$$

The eleven variable in this model can be solved by linear least squares and the minimum number of calibration points is six. It can be seen that the constraint on these points is that they should not be coplanar.

However, this method is not practical due to two facts: 1) a $3D$ calibration fixture is very expensive, 2) the model obtained is only sub-optimal. We have to resort to other methods. Tsai provided a RAC-based camera calibration algorithm to solve this problem [96]. Basically, Tsai found that by defining several independent parameters the RAC equation can be converted into a set of linear equation in terms of these intermediate parameters, which has unique solution under certain conditions. The rotation and translation parameters can be recovered from these intermediate parameters subsequently. The detailed solution is provided in Appendix A.

5.2 Viewpoint Evaluation

To evaluate the viewpoints, pictures taken from individual viewpoints are compared to the corresponding $3D$ scenes rendered by $NuGraf^{TM}$. Figure 5.8 shows one picture

and its corresponding rendering scene. Figure 5.9 shows another picture and its corresponding rendering scene.

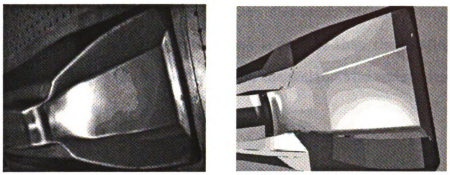

Figure 5.8: Picture 1 and rendering scene 1.

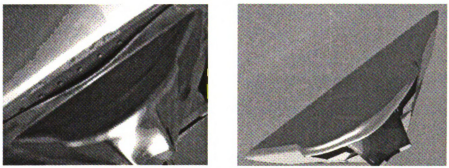

Figure 5.9: Picture 2 and rendering scene 2.

The above results show that the pictures and the $3D$ rendering scenes match quite well in the sense of the field of view and the visibility. The sharpness and resolution of the interested area in pictures are also within requirements. A thorough check on all the pictures shows that the viewpoints and their realization are satisfying, considering the positioning error of the robot and the calibration error. As another important issue, the coverage of all the surfaces by the pictures is guaranteed by the algorithm itself and the accuracy of the robot positioning.

5.3 Discussions

In this chapter, viewpoints are implemented and evaluated on an eye-in-hand robot. Some issues related to the implementation are discussed. Mainly, a new hand/eye calibration method is developed, this method is simpler than the existing method. Viewpoints are verified by comparisons between the real pictures and rendered images.

CHAPTER 6

CONCLUSIONS

6.1 Contributions of This Dissertation

In this dissertation, a CAD-based framework is proposed for constraint-satisfying robot motion planning in manufacturing applications. Under this framework, we developed an automated vision sensor planning system that can aid the dimensional inspection of automotive parts. The main contributions of this research are:

1) A divide-and-conquer strategy is proposed to solve the robot motion planning for compound surfaces. By decomposing the complex surfaces of the part into several patches with each satisfying certain geometric constraint, the robot motion planning is easy to solve.

2) A graph-based surface merging algorithm is developed to decompose the compound surfaces, which extracts the global geometric structure of the surfaces of the part.

3) A new vision sensor planning approach is proposed which combines two existing approaches and a bounding box concept is invented to efficiently integrate the four task constraints.

4) By discretizing the large flat surfaces, minimum viewpoint problem is solved by formulating it as a set-partitioning problem.

5) Robot kinematics constraint is integrated into vision sensor planning, which is set up as a weighted set-covering problem based on the discretizing scheme used in minimum viewpoint problem.

6) Robot path planning is studied and it is rendered as a Traveling Salesman Problem. A fast hierarchical approach is developed to solve it and a new algorithm is provided with bounded performance.

7) A new hand/eye calibration method is provided based on the solutions to the

robot localization and the camera calibration problem.

Although this work is mainly dedicated to the automated vision sensor planning problem for automotive parts, its general framework and basic ideas are not restricted on vision sensor planning. Many other CAD-based path planning problems in manufacturing can benefit from this framework, as well as the basic ideas. For example, as we already mentioned, automated painting gun path planning based on the CAD model of a part is a similar problem [41]. Other applications may include NC part programming, polishing path planning, etc.

APPENDIX A

TSAI'S RAC-BASED CAMERA CALIBRATION ALGORITHM

Tsai's Radial Alignment Constraint (RAC) algorithm calculates the camera extrinsic parameters, i.e. the elements of R and t, and other modified intrinsic parameters based on a set of coplanar calibration points, which is favorable in our system setup since we take the accurate grid points on the table as calibration points and they are coplannar.

The camera calibration algorithm consists of two steps.

1) calculate rotation matrix ${}^{c}R_{w}$ and the translational parameters t_x and t_y,

2) calculate the remaining parameters using the results of the first step.

As a discovery by Tsai, defining several independent intermediate parameters can convert the RAC equation into a set of linear equations in terms of these intermediate parameters, which has unique solution under certain conditions. The rotation and translation parameters can be recovered from these intermediate parameters subsequently.

The calibration algorithm is summarized as follows,

Stage 1), compute R and t_x and t_y.

a) define

$$\{\nu_1, \nu_2, \nu_3, \nu_4, \nu_5\} \equiv \{r_1 {t_y}^{-1}, r_2 {t_y}^{-1}, t_x {t_y}^{-1}, r_4 {t_y}^{-1}, r_5 {t_y}^{-1}\}$$

form a linear equation,

$$\begin{bmatrix} x_{w,i}Y_i & y_{w,i} & Y_i & -x_{w,i}\mu X_i & -y_{w,i}\mu X_i \end{bmatrix} \begin{bmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{bmatrix} = \mu X_i$$

here $x_{w,i}$ and $y_{w,i}$ are the coordinates of the ith calibration point. The minimum number of non-colinear calibration points needed to solve the equation is 5.

b) solve the $\{\nu_1, \nu_2, \nu_3, \nu_4, \nu_5\}$ using least square method.

c) define

$$C \equiv \begin{bmatrix} \nu_1 & \nu_2 \\ \nu_4 & \nu_5 \end{bmatrix}$$

If no row or column of C identically vanishes, it can be shown that

$$t_y{}^2 = \frac{S_r - \sqrt{S_r{}^2 - 4(\nu_1\nu_5 - \nu_4\nu_2)^2}}{2(\nu_1\nu_5 - \nu_4\nu_2)^2}$$

where $S_r \equiv \nu_1{}^2 + \nu_2{}^2 + \nu_3{}^2 + \nu_4{}^2 + \nu_5{}^2$. Otherwise the solution is

$$t_y{}^2 = (\nu_i{}^2 + \nu_j{}^2)^{-1}$$

where ν_i and ν_j are the nonzero elements in the appropriate row or column of C.

Assume $t_y > 0$, the following parameters are calculated,

$$r_1 = \nu_1 t_y \quad r_2 = \nu_2 t_y$$

$$r_4 = \nu_4 t_y \quad r_5 = \nu_5 t_y$$

$$t_x = \nu_3 t_y$$

Then using an arbitrary calibration point, the following coordinates are calculated,

$$x = r_1 x_w + r_2 y_w + t_x$$

$$y = r_4 x_w + r_5 y_w + t_y$$

The sign of x and X(y and Y) should be consistent, so if the above coordinates conflict the consistency, the sign of t_y is reversed.

$$r_3 = \sqrt{1 - {r_1}^2 - {r_2}^2}$$

$$r_6 = -sign(r_1 r_4 + r_2 r_5)\sqrt{1 - {r_4}^2 - {r_5}^2}$$

$$[r_7\ r_8\ r_9]^T = [[r_1\ r_2\ r_3]^T \times [r_4\ r_5\ r_6]^T$$

Once R and t_x, t_y are known, one can estimate t_z through the following equation

$$\begin{bmatrix} -X_i & x_i - x_i {r_i}^2 \end{bmatrix} \begin{bmatrix} t_z \\ f_x \\ k f_x \end{bmatrix} = X_i w_i$$

where

$$x_i \equiv r_1 x_{w,i} + r_2 y_{w,i} + t_x$$

$$w_i \equiv r_7 x_{w,i} + r_8 y_{w,i}$$

$f_x \equiv f s_u$ is the product of the focal length and scale vector. k is the radial distortion coefficient.

AUTHOR'S PUBLICATIONS

[1] W.Sheng, N. Xi, M.Song, Y.Chen, J.S, Rankin III. Automated CAD-Guided Automobile Part Dimensional Inspection. *International Conference on Robotics and Automation* , 2000, San Francisco.

[2] W.Sheng, N.Xi, J.Tan , M.Song, Y.Chen, J.S.Rankin III, P. Macneille. CAD-Based Sensor Planning for Dimensional Inspection. *Japan-USA Symposium on Flexible Automation, July*, 2000, Michigan.

[3] W.Sheng, N.Xi, M.Song, Y.Chen, J.S Rankin III, P.Macneille. CAD-Guided Robot Motion Control for Sensor Planning. *World Automation Congress (WACONG2000).*

[4] N. Xi, W.Sheng, J. Tan, M.Song, Y.Chen, J. S. Rankin III. Robot Motion Control in Automated Automotive Part Inspections. *The third Asian Control Conference*, 2000, Shanghai, China.

[5] W.Sheng, N.Xi, M.Song, Y. Chen, P. Macneille. Automated CAD-Guided Robot Path Planning for Spray Painting of Compound Surfaces. *International Conference on Intelligent Robots and Systems*, 2000, Japan.

[6] W.Sheng, N. Xi, M.Song, Y.Chen. Graph-based Surface Merging in CAD-guided Dimensional Inspection of Automotive Parts. *International Conference on Robotics and Automation* , 2001, Seoul, Korea.

[7] W.sheng, N.Xi, M.Song, Y.Chen. CAD-Guided Robot Motion Planning. *Industrial Robot*, 2001.

[8] W.Sheng, N.Xi, M.Song, Y.Chen. Sensor planning with kinematics constraint for dimensional inspection of sheet metal parts. *International Conference on Intelligent Robots and Systems*, 2001, Hawaii, USA.

[9] H.Chen, W.Sheng, N.Xi, M.Song, Y.Chen. CAD-Guided Uniformity Guaranteed Robot Trajectory Planning for Spray Painting of Free Form Surfaces. *M2VIP* 2001, HongKong.

[10] W.Sheng, N.Xi, M.Song, Y.Chen. Near-Optimal-Time path planning in CAD-guided part dimensional inspection. *15th IFAC World Congress on Automatic Control*, 2002, Barcelona, Spain.

[11] H.Chen, W.Sheng, N.Xi, M.Song, Y.Chen. Automated Robot Trajectory Planning for Spray Painting of Free-Form Surfaces in Automotive Manufacturing. *International Conference on Robotics and Automation* , 2002, Washing D.C. USA.

BIBLIOGRAPHY

[1] S. Sakane, M. Ishii, and M. Kakikura. Occlusion avoidance of visual sensors based on a hand eye action simulator system: Heaven. *Advanced Robotics*, 2(2):149–165, 1987.

[2] C. K. Cowan and P. D. Kovesi. Automatic sensor placement from vision task requirements. *IEEE transaction on pattern analysis and machine intelligence*, 10(3):407–416, 1988.

[3] H. Zhuang and Z. S. Roth. *Camera-aided robot calibration.* CRC Press, Florida, 1996.

[4] T.C.Chang, R.A.Wysk, and H.P.Wang. *Computer-aided manufacturing: second edition.* Prentice Hall, New Jersey, 1998.

[5] H.C Zhang and L.Alting. *Computerized manufacturing process planning systems.* Chapman and Hall, London, 1994.

[6] H.Lau and B.Jiang. A generic integrated system from cad to capp: a neural file-cum-gt approach. *Computer Integrated Manufacturing Sytems*, 11:67–75.

[7] J.J.Shah and M.Mantyla. *Parametric and feature-based CAD/CAM.* John Wiley and Sons, New York, 1995.

[8] A.A.G.Requicha and J.R.Rossignac. Solid modeling and beyond. *IEEE Computer Graphics and Applications*, 12(5):31–44, 1992.

[9] M.Marefat and R.L.Kashyap. Automatic construction of process plans from solid model representations. *IEEE transactions on systems, man, cybernetics*, 22:1097–1115.

[10] S.N.Spitz and A.A.G.Requicha. Hierchical constraint satisfaction for high-level dimensional inspection planning. In *Proceedings of the 1999 IEEE international Symposium on Assembly and Task Planning*, pages 374–380, 1999.

[11] K.H.Lee and H.P.Park. Automated inspection planning of free-form shape parts by laser scanning. *Robitcs and Computer Integrated Manufacturing*, 1:201–210, 2000.

[12] G. Hu and G. Stockman. 3-d surface solution using structured light and constraint propagation. *IEEE transaction on pattern analysis and machine intelligence*, 11(4):390–402, 1989.

[13] K.Tuffentsammer and G.Arndt. The influence of nc flexibility on the concept of group technology. *Annals of the CIRP*, 32(1), 1983.

[14] J.Shah, P.Sreevalsan, and A.Mathew. Survey of cad/feature-based process planning and nc programming techniques. *Computer-aided engineering journal*, (2):25–33, 1991.

[15] H.D.Park and O.R.Mitchell. Cad based planning and execution of inspection. In *Proceedings of the 1988 Computer Society Conference on Computer Vision and Pattern Recognition*, pages 858–863, 1988.

[16] F.Prieto, T. Redarce, P.Boulanger, and R.Lepage. Cad-based range sensor placement for optimum 3d data acquisition. *Proceedings of the Second International Conference on 3-D Digital Imaging and Modeling*, 1:128–137, 1999.

[17] S.O.Mason and A.Grun. Automatic sensor placement placement fo raccurate dimensional inspection. *Computer vision and image understanding*, 61(3):454–467, 1995.

[18] B.Triggs and C.Laugier. Automatic camera placement for robot vision tasks. In *Proceedings of the 1995 IEEE international conference on robotics and automation*, pages 1732–1737, 1995.

[19] E.Trucco, M.Umasuthan, A.M.Wallace, and V.Roberto. Model-based planning of optimal sensor placement for inspection. *IEEE Transactions on Robotics and Automation*, 13(2):182–193, 1997.

[20] D.J.Weir, M.J.Milroy, C.Bradley, and G.W.Vickers. Reverse engineering physical models employing wrap-around b-spline surfaces and quadrics. In *Proc Instn Mech Engrs*, volume 210, pages 147–157, 1996.

[21] R.Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(10):1016–1030, 1999.

[22] J.E.Banta and M.A.Abidi. Autonomous placement of a range sensor for acquisition of optimal 3-d models. In *Proceedings of the 1996 IEEE IECON 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, volume 3, pages 1583–1588, 1996.

[23] M.Reed and P.Allen. Constraint-based sensor planning for scene modeling. In *Proceedings of the 1999 IEEE Conference on Robotics and Automation*, pages 131–136, 1999.

[24] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE transaction on robotics and automation*, 11(1):86–104, 1995.

[25] S. Sakane, T. Sato, R. Niepold, and Y. Shirai. Illumination setup planning for a hand-eye system based on an environmental model. *Advanced Robotics*, 6(4):461–482, 1992.

[26] S. Yi, R. M. Haralick, and L. G. Shapiro. Automatic sensor and light source positioning for machine vision. In *Proceedings of 10th international conference on pattern recognition*, pages 55–59, 1990.

[27] C.K.Cowan and A.Bergman. Determining the camera and light source location for a visual task. In *Proceedings of 1989 IEEE international conference on robotics and automation*, pages 509–514, 1989.

[28] K. A. Tarabanis, R. Y. Tsai, and P. K. Allen. The mvp sensor planning system for robotic vision tasks. *IEEE transaction on robotics and automation*, 11(1):72–85, 1995.

[29] K. A. Tarabanis, R. Y. Tsai, and A. Kaul. Computing occlusion-free viewpoints. *IEEE transaction on pattern analysis and machine intelligence*, 18(3):279–292, 1996.

[30] X.Gu, M.M.Marefat, and F.W.Ciarallo. A robust approach for sensor placement in automated vision dimensional inspection. In *Proceedings of the 1999 IEEE international conference on robotics and automation*, pages 2602–2607, 1999.

[31] Y.Yao and P.Allen. Computing robust viewpoints with multi-constraints using tree annealing. In *Proceedings of the 1995 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 993–998, 1995.

[32] S.Abrams, P.K.Allen, and K.Tarabanis. Computing camera viewpoints in an active robot work cell. *The international journal of robotics research*, pages 267–285, 1999.

[33] J. Y. Lai and D. J. Wang. A strategy for finish cutting path generation of compound surfaces. *Computers in industry*, 25:189–209, 1994.

[34] K.A.Tarabanis and R.Y.Tsai. Computing occlusion-free viewpoints. In *The Computer Society Conference on Computer Vision and Pattern Recognition*, pages 802–807, 1992.

[35] E. Krotkov. *Exploratory visual sensing with an agile camera.* PhD thesis, University of Pennsylvania, 1987.

[36] The VRML Consortium Incorporated. *The Virtual Reality Modeling Language*, 1997.

[37] K.Haris, S.N.Efstratiadis, and N.Maglaveras. Watershed-based image segmentation with fast region merging. In *Proceedings of 1998 international conference on image processing*, pages 338–342, 1998.

[38] B. K. P. Horn. *Robot vision.* McGraw-Hill, New York, 1986.

[39] B. A. Payne and A. W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, (1):65–71, 1992.

[40] Structural Dynamics Research Corporation. *I-DEAS master series.* Milford, Ohio, 1994.

[41] W.Sheng, N.Xi, M.Song, Y.Chen, and Perry MacNeille. Automated cad-guided robot path planning for spray painting of compound surfaces. In *Proceedings of 2000 International Conference on Intelligent Robots and Systems*, volume 3, pages 1918–1923, 2000.

[42] M.G.Kay and R.C.Luo. Camera placement for global vision. In *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 917–924, 1992.

[43] J. O'Rourke. *Art gallery theorems and algorithms.* Oxford, New York, 1987.

[44] T.Danner and L.E.Kavraki. Randomized planning for short inspection paths. In *Proceedings of the 2000 IEEE international conference on robotics and automation*, pages 971–976, 2000.

[45] W.Kang, M.Song, J.S.Rankin II, and Y.Chen. The smallest visibility set in manufacturing applications. In *Proc. IEEE Hong Kong Symposium on Robotics and Control*, pages 447–452, 1999.

[46] *Combinatorial Optimization*, chapter 7, pages 151–210. 1979.

[47] E.Balas and M.W.Padberg. Set partitioning: A survey. *SIAM Review*, (18):710–760, 1976.

[48] R.E.Marsten. An algorithm for large set partitioning problems. *Management Science*, (20):774–787, 1974.

[49] R.E.Marsten and F.Shepardson. Exact solution of crew scheduling problmes using the set partitioning model: Recent successful applications. *Networks*, (11):165–177, 1981.

[50] T.J.Chan and C.A.Yano. A multiplier adjustment approach for the set partitioning problem. *Operations Research*, (40):40–47, 1992.

[51] F.Harche and G.L.Thompson. The column subtraction algorihtms: An exact method for solving weighted set covering, packing and partitioning problems. *Computers and Operations Research*, (21):689–705, 1994.

[52] L.H.Tasi. The modified differencing methods for the set partioning problem with cardinality constraints. *Discrete Applied Mathematics*, (63):175–180, 1995.

[53] D.M.Ryan and J.C.Falkner. On the integer properties of scheduling set partitioning models. *European Journal of Operational Research*, (35):422–456, 1988.

[54] A.Atamturk, G.L.Nemhauser, and M.W.P.Savelsbergh. A combined lagrangian, linear programming and implication heuristic for large-scale set partitioning problems. *Journal of Heuristics*, (1):247–259, 1995.

[55] D.Levine. *A parallel genetic algorithm for the set partitioning problem.* PhD thesis, Illinois Institute of Technology, 1994.

[56] P.C.Chu and J.E.Beasley. Constraint handling in genetic algorithms: The set partitioning problem. *Journal of Heuristics*, (18):323–357, 1998.

[57] J.C.Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, 1991.

[58] P.V.Hentenryck. *The OPL Optimization Programming Language.* MIT Press, 1999.

[59] D.Hsu, J.C.Latombe, and S.Sorkin. Placing a robot manipulator amid obstacles for optimized execution. In *Proceedings of the 1999 IEEE international symposium on assembly and task planning*, pages 280–285, 1999.

[60] B.Nelson, K.Pedersen, and M.Donath. Locating assembly tasks in a manipulator's workspace. pages 1367–1372, 1987.

[61] G.J.A.Pamanes, S.Zeghloul, and J.P.Lallemand. On the optimal placement and task compatibility of manipulators. In *Fifth International Conference on Advanced Robotics, 1991*, volume 2, pages 1694–1697, 1991.

[62] H.Seraji. Reachability analysis for base placement in mobile manipulators. *Journal of Robotic Systems*, 12(1):29–43, 1995.

[63] E.Balas and A.Ho. Set covering algorithms using cutting planes, heuristics, and subgradient optimziation: A computational study. *Mathematical Programming Study*, (12):37–60, 1980.

[64] V.Chvatal. A greedy heuristic for the set covering problem. *Mathematics of Operations Research*, (4):233–235, 1979.

[65] J.E.Beasley. A lagrangian heuristic for set-covering problems. *Naval Research Logistics*, (37):151–164, 1990.

[66] D.Peleg, G.Schechtman, and A.Wool. Randomized approximation of bounded multicovering problems. *Algorithmica*, (18):44–66, 1997.

[67] M.Hifi, V.T.Paschos, and V.Zissimopoulos. A neural network for the minimum set covering problem. *Chaos, Solutions and Fractals*, (11):2079–2089, 2000.

[68] J.E.Beasley and P.C.Chu. A genetic algorithm fo the set covering problem. *European Journal of Operational Research*, (94):392–404, 1996.

[69] L.Lorena and L.D.S.Lopes. Genetic algorithms applied to computationally difficult set covering problem. *Journal of the Operational research Society*, (48):440–445, 1997.

[70] S.Sen. Minimal cost set covering using probabilistic methods. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing*, pages 157–164, 1993.

[71] T.Yoshikawa. Manipulability of robotic mechanisms. *The international journal of robotics research*, 4(2):3–9, 1985.

[72] G.J.A.Pamanes. A criterion for optimal placement of robotic manipualtors. In *IFAC Proceedings of the 6th symposium on informaiton control problems in manufacturing technology*, pages 183–187, 1989.

[73] S.L.Chiu. Task comatibility of manipulators postures. *The international journal of robotics research*, 7(5):13–21, 1988.

[74] Y.Edan, T.Flash, U.M.Peiper, I.Shmulevich, and Y.Sarig. Near-minimum-time task planning for fruit-picking robots. *IEEE Transactions on Robotics and Automation*, 7(1):48–56, 1991.

[75] K.G.Shin and N.D.Mckay. Selection of near minimum time geometric paths for robotic manipulators. *IEEE Transactions on Automic Control*, 31(6):501–511, 1986.

[76] E.L.Lawler, J.K.Lenstra, A.H.G.R.Kan, and D.B.Shmoys. *The Traveling Salesman Problem.* John Wiley and Sons, Chichester, 1985.

[77] J.Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[78] G.B.Dantzig, D.R.Fulkerson, and S.M.Johnson. On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, 7:58–66, 1954.

[79] R.E.Bellman. Dynamic programming treatment of the traveling salesman problem. *J.Assoc. Comput. Mach*, 9:61–63, 1962.

[80] D.J.Rosenkrantz, R.E.Stearns, and P.M.Lewis. An analysis of several heuristics for the traveling salesman problem. *SIAM journal of computing*, 6:563–581, 1977.

[81] J.B.Kruskal JR. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of American Mathematics Society*, 7:48–50, 1956.

[82] N.Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 338, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh,PA, 1976.

[83] S.Lin and B.W.Kernighan. An effective heuristic algorithm for the traveling salesman problem. *Operations Research*, 21:498–516, 1999.

[84] N.Guttmann-Beck, R.Hassin, S.Khuller, and B.Raghavachari. Approximation algorihthms with bounded performance guarantees for the clustered traveling salesman problem. *Algorithmica*, 28:422–437, 2000.

[85] J.A.Chisman. The clustered traveling salesman problem. *Computers and Operations Research*, 2:115–119, 1975.

[86] F.C.J.Lokin. Procedures for traveling salesman problems with additional constraints. *European Journal of Operational Research*, (3):135–141, 1978.

[87] G.Laporte, J.Y.Potvin, and F.Quilleret. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, (2):187–200, 1996.

[88] M.Gendreau, G.Laporte, and J.Y.Potvin. Heusitics for the clustered traveling salesman problem. Technical report, Centre de recherche sur les transports, Universite de Montreal, 1994.

[89] J.Y.Potvin and F.Guertin. *The clustered traveling salesman problem: a genetic approach.* 1996.

[90] M. R. Anderberg. *Clustering analysis for applications.* Academic Press, New York, 1973.

[91] A.K.Jain. *Algorithms for Clustering Data.* Prentice Hall, New Jersey, 1988.

[92] S.S. Skiena. *The algorithm design manual.* Springer-Verlag, Rensselaer, NY, 1998.

[93] J.A.Hoogeveen. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10(5):291–295, 1991.

[94] Okino Computer Graphics Inc. *NuGraf rendering system.* Ontario,Canada, 1998.

[95] Y.C.Shiu and S.Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transformation equations of the form ax=xb. *IEEE Transactions on Robotics and Automation*, 5(1):16–27, 1989.

[96] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE journal of robotics and automation*, 3(4):323–344, 1987.