Stellingen behorende bij het proefschift

"Object Recognition for Flexible Assembly Using Stereo Vision",

van Johannes Buurman.

1. Kenmerkgebaseerde stereovisie heeft met betrekking tot herkenning als voordeel over andere 3-d vision methoden dat er vroegtijdig een significante datareductie plaatsvindt (dit proefschrift, hoofdstuk 4).

2. Stereovisie op basis van ellipsen kan een zinvolle aanvulling op andere kenmerkgebaseerde stereovisie-methoden vormen. Voorwaarde hiervoor is wel dat de scene veelal cylindervormige objecten bevat (dit proefschrift, hoofdstuk 4).

3. Het leren van modellen van objecten is aanzienlijk moeilijker dan het met behulp van modellen herkennen ervan (dit proefschrift, hoofdstuk 7).

4. De discipline van het beschrijven van het gehele traject van sensor tot en met herkenning verdient een zelfstandige plaats naast de disciplines patroonherkennen en beeldbewerking.

5. Door wetenschappelijke publikaties slechts in te delen naar gebruikte techniek of toepassingsgebied gaat een belangrijk inzicht verloren. Dit geldt bij uitstek voor beeldbewerking.

6. Gebruik van de juiste sensoren is vaak belangrijker voor het slagen van een project dan extra investeringen in actuatoren. Een goed voorbeeld hiervoor is het vangen van Roadrunners.

7. Gegeven dat de Nederlandse politieke cultuur wordt gekenmerkt door veelstemmigheid, besluiteloosheid en het moeizame zoeken naar consensus, valt te verwachten dat de overheersende politieke cultuur in een te vormen Europese Unie de Nederlandse zal zijn.

8. De Nederlandse arbeidswetgeving gaat er ten onrechte van uit dat particuliere bedrijven hun werknemers proberen uit te buiten en overheidsinstellingen niet.

9. Elektronische post combineert de directheid van het gesproken woord met de reproduceerbaarheid van het geschrevene. Hierin vindt het zowel zijn kracht als zijn zwakte.

10. Elk juridisch of financieel onderscheid tussen samenwonen en trouwen maakt de keuze tussen beide minder een ethische afweging en meer een voor de berekenende burger.

11. Binnenkort zijn er in Nederland meer zogenaamde "abdijbieren" dan monniken.

12. Tweede-fase onderwijs is als zeepreclame: het gaat niet om de kwaliteit van het produkt maar om de promotie.

Object Recognition for Flexible Assembly Using Stereo Vision

Object Recognition for Flexible Assembly Using Stereo Vision

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus, prof. ir. K.F. Wakker,
in het openbaar te verdedigen ten overstaan van een commissie
aangewezen door het College van Dekanen
op maandag 6 december 1993 te 10.00 uur

door

Johannes Buurman

geboren te Rotterdam,
natuurkundig ingenieur.

Dit proefschrift is goedgekeurd door de promotor prof. dr. I.T. Young.

Dr. ir. R.P.W. Duin heeft als toegevoegd promotor in hoge mate bijgedragen aan het totstandkomen van dit proefschrift.

Published and distributed by:

Delft University Press
Stevinweg 1
2628 CN Delft
the Netherlands

Telephone: (0)15 - 783254
Telefax: (0)15 - 781661

ISBN 90 - 6275 - 938 - 6 / CIP

Printed in the Netherlands.

"Oh! Piglet," said Pooh excitedly, "we're going on an Expotition, all of us, with things to eat. To discover something."
"To discover what ?" said Piglet anxiously.
"Oh! just something."

A.A. Milne, *Winnie-the-Pooh.*

Ter nagedachtenis
aan mijn grootvader,
Jacob Buurman
(1901-1971).

Table of Contents

1 Introduction

1.1 Computer vision

This thesis describes work in the field of computer vision. In this introduction, an attempt is made to delineate this field, and indicate where this research should be positioned. Then, some remarks are made on how vision research could be done. The position of robot vision will be discussed. Finally, the structure of the rest of the thesis is introduced.

Computer vision is defined in a popular textbook[8] as "the enterprise of automating and integrating a wide range of processes and representations used for vision perception". This definition is at first glance by no means clear. "Vision perception" seems to imply that there is something to be perceived, which we call a scene. From this scene, an observation or *image* is obtained using some kind of sensor using an optical (or similar) process. However, as vision is not just about "seeing" but rather about "perception", useful information has to be extracted from the image. Computer vision covers the whole trajectory from scene to final information, and hence the above definition.

However, computer vision does not attempt to solve one specific problem. Different applications can be found in the medical world, in the industrial world and in the military world. These may vary in the scene to be observed (either in the subject of the scene or the dimensionality, ranging from simple 2-dimensional black and white views to time series of 3-dimensional or coloured scenes) and in the kind of information required from the scene (other images, qualitative observations, measurements or combinations of these). As a result of this variation, different sensors and data representations are being used by different researchers for different problems. As there are many different kinds of scenes, sensors, images and useful information, computer vision covers a broad range of processes and data representations.

Because of the large scope of the definition, it is often practical to distinguish several disciplines within the field of vision. There are of course the mere practical distinct areas, such as motion analysis, stereo vision or remote sensing, where the main point is the application of a specific technique or the application to a specific problem. However, it is also not uncommon to divide the field into Image Processing and Pattern Recognition.

Image processing can simply be defined as "manipulation of images"[6]. In fact, there are several definitions used by different researchers in different fields as a result of which confusion arises. This involves whether such areas as image coding, image compression and image segmentation are part of image processing[†]. In the Pattern Recognition Group, it is common not to include coding and compression in Image Processing. This is the definition used throughout this thesis.

---

†. In the Dutch language, most of these debates can be solved by the distinction between "beeldbewerking", where the result of any operation is another image, and "beeldverwerking", where the result could br anything. Unfortunately, this cannot be translated into English.

An example of image processing could be, given an image of an object, to generate an image that has one value where the object was present and another value where the object was not present. Also, the size of the area corresponding to the object could be measured.

Pattern recognition is the traditional way of doing object recognition. From a processed image, a number of measurements is extracted. These measurements can then be used to assign one of a set of *labels* to the regions in the image. For instance, based on a measurement and the set of labels { block, pyramid }, the label "block" could be assigned.

The division of vision in image processing and pattern recognition is too rigid to describe modern vision research. A lot of research uses a *model* of the scene and objects and uses knowledge acquired about the scene to update the model and direct further vision. Furthermore, this process may require a continuous interaction between image processing and other knowledge manipulation methods. For instance, a recent development is active vision, in which the output of the vision system is used to direct its attention towards interesting objects in the scene. This can be implemented at various levels, ranging from cameras that move and are readjusted to software that exhibits similar behaviour.

The problem addressed in this thesis is a typical computer vision problem: the recognition and pose determination of industrial parts as they enter a flexible assembly cell. A scene is observed that contains an unknown number of objects, and the aims are not only to determine how many and which, but also is what position and orientation they are there. This requires a combination of techniques that will be discussed in subsequent chapters.

1.2 Methodology

There are several approaches to vision (apart from traditional pattern recognition), based on models of human perception. These include expert systems and neural networks. Although successes of each approach has been reported, no one appears to be the solution to all problems. Therefore, computer vision should be the discipline of building working vision systems for specific applications, where components can be borrowed from each of the approaches mentioned above. This idea seems to be held by most researchers in the field.

This seems to reduce vision to an engineering discipline. Most researchers are working on problems that are at least slightly different, and since they are all trying to build working systems, these systems are generally hard to compare. Moreover, researchers publish solutions to their own specific problems, resulting in a flood of publications. This seems to be one of the causes behind the recent debate on the future of computer vision[39].

The first way out of this unsatisfactory situation seems to be to focus on small sub-problems which are common to a lot of applications, and on which comparable results can be obtained. This has as a disadvantage that no working systems will be produced. The second alternative would be the development of "standard" problems of sufficient complexity, that should be made available to as many researchers as possible. As standard problems are not available yet, they should be proposed.

1.3 Robot vision

Why should one use vision in the first place, in order to solve automation problems? The required information is rarely visual (this is usually referred to as visualisation rather than vision), and other ways of getting that information are almost always available. In a recent critical paper[54], Miller describes how he made his intelligent robot vehicle work by reducing his sensor system from a stereo vision system (a complex system using several cameras in order to obtain detailed 3-d information) yielding megabytes of data to a few contact and proximity sensors yielding one byte each.

Part of this success can of course be explained by the failure of computer vision to come up with real progress, as described in [39] and [54]. It could be argued though, that it is the result of a bad specification as well. If the task is to see whether something large is in the way, stereo vision may not be the correct solution. However, if the task is to recognise an object, stereo vision can be the solution, providing the stereo vision system is designed well. In particular, such a system should deliver data suitable for recognition rather than nice range images. If the task also requires flexibility and easy reconfiguration (as is the case in this thesis), vision is even very likely to provide a possible solution, because after all, the parts to be reconfigured are in this case the contents of databases rather than mechanical devices. The challenge is to design such a system.

1.4 This thesis

In this thesis, a computer vision system is described for object recognition in a a flexible assembly cell. As parts enter the cell (lying on pallets, in separate slots), they have to be recognised and their position established. This means that scenes containing several objects are to be examined. From the way the problem is stated, it follows that the engineering approach is followed: the main goal is to arrive at working algorithms, without choosing a model beforehand.

The main reason for using vision in this case is flexibility, not the flexibility in operation that Miller[54] argues can be obtained easily with simple sensors or no sensors at all, but flexibility in configuration: the cell is meant for the production of small series and it must be possible to start production of new product series with minimal changes. A vision system can be reconfigured to recognise new, similar parts with only the change of some files, whereas mechanical devices would have to be readjusted or otherwise changed.

The system uses stereo vision for obtaining 3-dimensional data about the scene. Rather than trying to obtain a good range image of the scene, it tries to find those features that are specific for the family of parts considered: straight lines and circular arcs. A matcher is available that compares the wireframe generated by the stereo vision system to a set of CAD-derived models. This is illustrated in Fig. 1-1, which reappears throughout this thesis as the separate parts of the system are described.

This thesis has the following structure: chapter 2 supplies some background theory on the subjects of stereo vision and recognition. Chapter 3 introduces the DIAC project and states the

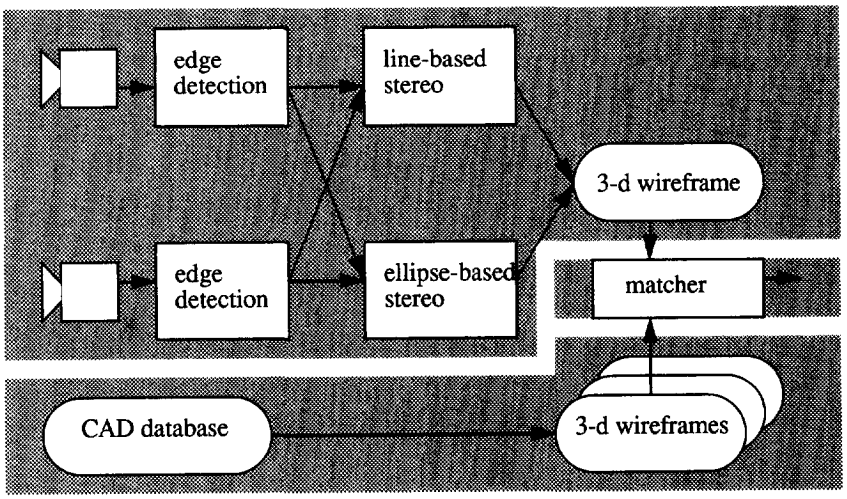


Fig. 1-1 Overview of the recognition system. The output of the stereo vision system (top, chapter 4) is compared to the CAD-derived or learned models (bottom, chapter 7) by the matcher (right, chapter 5). The system as a whole is evaluated in chapter 6.

problem to be solved there. Chapter 4 describes the stereo vision system as a separate part, whereas chapter 5 describes the recognition system. In chapter 6, the combination of the two is evaluated. In chapter 7, the acquisition of models is described, either from CAD models or from a learning procedure using the same stereo vision system and algorithms similar to those in recognition. Finally, chapter 8 concludes the thesis with a discussion of results.

1.5 Acknowledgement

This work was made possible by the SPIN/FLAIR project, which had the Delft Intelligent Assembly Cell as one of its goals. It was also partially sponsored by the SPIN 3-D image analysis project.

2 CAD-based object recognition

The object recognition system described in this thesis can be divided into two main parts: a system for acquisition of an observation of the scene, and a system for recognition of parts in the observation. This chapter deals mainly with the foundations of those: what ways are there of acquiring the observation, and what ways are there of doing the recognition. Related to these issues is the choice of a model representation. For a field as large as Computer Vision, it is impossible to be complete in these respects. Therefore, the section on acquisition is limited to an overview of sensing methods. The section on recognition focuses on the main paradigms for doing object recognition in contemporary computer vision.

2.1 Introduction

A basic outline of an object recognition system is given in Fig. 2-1. In the lower part, data is acquired (which involves image processing). Extracted from this is a description of the scene in primitives, suitable for recognition. These are offered to a recognition part which compares the scene description to one or more models. This process leads to the result: a list of object identities and poses. For more complex systems, the recognition process may influence the image acquisition process (indicated by the grey arrow).

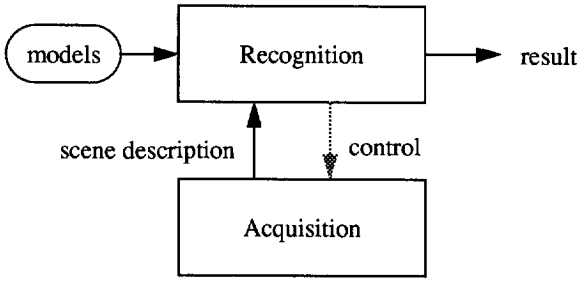


Fig. 2-1 Basic architecture of a recognition vision system. It consists of a data acquisition part that obtains images and a recognition part that uses the models generated elsewhere. In active systems, the recognition part will control the aquisition part.

In this chapter, several of the foundations of three of these steps are addressed. At first, different models are discussed. Then, different ways of acquiring the scene description are indicated. Finally, different ways for doing recognition are discussed. The fourth non-trivial step, control of the acquisition process by the recognition process, is a complex topic in itself and lies outside the scope of this thesis.

2.2 Models

2.2.1 CAD models and vision models

For the application addressed in this thesis, coupling the vision system and a CAD database is of prime importance in order to achieve the required automatic reconfiguration when new parts are defined. Unfortunately, the models used in both worlds are not necessarily compatible, a problem that was addressed earlier by Bhanu[10]. Of the various ways of describing objects in CAD systems, the two most important are Constructive Solid Geometry (CSG) and boundary representations (b-rep). These will be discussed below.

2.2.2 Constructive solid geometry

In CSG, an object is described in terms of simple geometric shapes, combined by additions and subtractions. For instance, a ring can be described as the subtraction of a small cylinder from a larger cylinder, see Fig. 2-2. Although CSG is very useful in CAD systems, its does not lend itself very well to vision systems. The reason for this is the fact that vision only sees the outside of objects, so many details introduced by geometric manipulations are invisible.

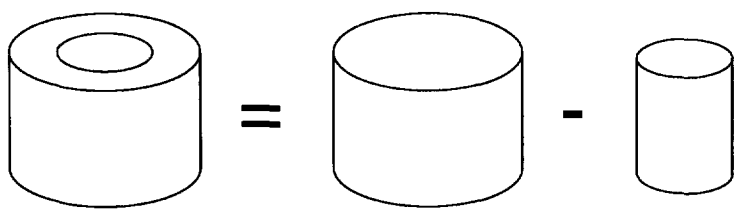


Fig. 2-2 Constructive solid geometry: an object is considered as a combination of simple geometric parts by means of addition, or, in this case: substraction.

2.2.3 Boundary representations

Boundary representations are a broad class of descriptions of the outside of objects. This outside can be described by surface patches with various mathematical representations, or by various kinds of surface contours, or both. Some kind of boundary representation can usually be produced by a CAD system. Possibilities include:

- Wire frames of straight lines: an object is described by a straight line approximation of the edges of its surfaces. This is the simplest kind of description. It can only describe polyhedra exactly (such as the objects used in chapter 6), any curved shapes have to be approximated by a number of straight lines.
- Wire frames including other curves: including circles, ellipses or splines extends the class of objects that can be represented in this way. In the application described in chapter 3, all objects can be described by straight lines and circular arcs.

- Wire frames extended with descriptions of the enclosed surface patches. This could be a plane equation, of an equation of a quadric surface, or a more complex description.
- Descriptions of objects as sets of surface patches only.

Wire frame representations lend themselves well for use with sensing methods that acquire data of object edges, while surface patch representations can best be used with sensing methods that acquire data over complete surfaces. The choice of a model representation is therefore closely related to the choice of a sensing method, as described in the next section.

2.3 Sensing methods

2.3.1 Introduction

Although other image sensors exist, the main tool for vision is the camera. This device consists of an optical system (a set of lenses, focused on the scene) and a set of sensors, which translate the incoming light into a matrix (a *digital image*) whose elements represent the brightness in a specific area. Usually, the rectangular grid of the matrix directly represents a similar grid of areas.

The following is a model of the optical behaviour of a camera. It assumes linear behaviour, and is derived from the simplest optical model, that where the sensor is placed behind a pin-hole, see Fig. 2-3.

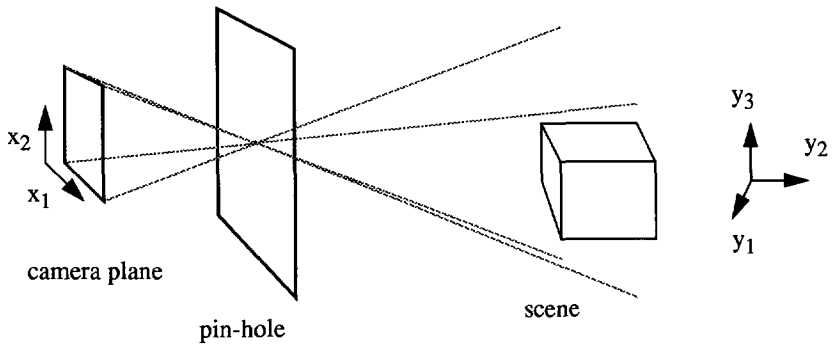


Fig. 2-3 The pin-hole camera model and associated coordinate transformation.

The optical system of the camera can be described by its mapping of world coordinates y_i to camera coordinates x_i (in homogeneous coordinates, multiplication with a *4-by-3 matrix C*)

$$\begin{bmatrix} x_1 s \\ x_2 s \\ s \end{bmatrix} = C \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix}. \qquad (1)$$

After the optical system, the remaining data is sampled by the set of sensors (typically a CCD array), such that the above mentioned matrix appears. Although this process is not perfect (the sampling is not done in points, and the grid can be distorted), it is often assumed to be.

2.3.2 2-D vision

The simplest way of acquiring images is using one camera. This gives us monocular vision. Equation (1) shows that a point in an image may correspond with a set of points in space: if C, x_1 and x_2 are known there are three equations with four unknowns (y_1, y_2, y_3 and s), or after elimination of s, two equations with three unknowns. This set of points turns out to be a line[37]. Sometimes it is possible to obtain depth information from monocular vision using secondary depth cues (perspective, shading, focus), however the use of these results in sparse and usually inaccurate information, see e.g. [1].

2.3.3 Passive stereo vision

Using two or more cameras, depth information can be obtained by triangulation. The principle is illustrated in Fig. 2-4. If a point can be identified in both images, the corresponding point in 3-dimensional space can be calculated by crossing the corresponding lines in space, if the camera geometry is known. This can be done for all points that can be identified in both images. The result of this procedure is a 2-d image, in which for a number of points the distance to the imaging plane can be calculated. Such an image, reminiscent of a height map, is called a range image.

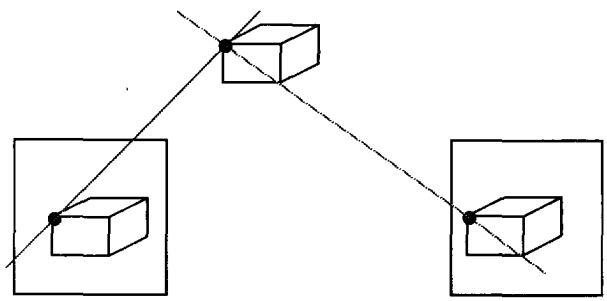


Fig. 2-4 The principle of stereo vision. A point is visible in both images. Using the corresponding camera matrices, the lines can be computed of points in 3-d dimensional space that may correspondto each. The corresponding point can be found by calculating the point where the lines cross.

It follows from equation (1) that the stereo vision problem is over-determined: once two points are known (one in each image) there are four equations with three unknowns. This means that the 3-d point must be calculated using an extra constraint. For instance, the two lines in Fig. 2-

4 will in general not cross in one point. This means that the closest point must be determined.

The main problem in stereo vision lies in the identification of corresponding points. Shirai[62] distinguishes between area-based stereo vision and feature-based stereo vision. In area-based stereo vision, corresponding points are identified by searching for similar areas around points in the two images. This is done by calculating a similarity function (e.g. the correlation of light intensity), and assigning correspondence to points for which this function shows a local maximum. As this can be a time-consuming process, candidate areas must be selected using known properties of the camera geometry. The most common setup is to have the cameras aligned so that the only difference between their parameters is a horizontal translation.

Feature-based stereo vision was developed in order to overcome two drawbacks of area-based stereo vision: the high complexity of computing the similarity function, and the dependence on photometric properties of the two cameras. In feature-based stereo vision, features (like edges) are detected, and the stereo vision process calculates depth from the correspondence of those features. As there are less features than candidate areas, this is generally a faster process, and as features are detected in both images independently, it is less suitable to photometric properties. A disadvantage of feature-based stereo is that features must always be detected in both images in order to get depth information.

In both ways of stereo vision, the depth information recovered is sparse in nature: if correspondence cannot be established, no depth information is recovered. This can either be due to a failure of the detection algorithm, or to the fact that a point is visible only in one image and not in the other (because it is occluded). In both cases, this means that either this sparseness has to be unimportant for the application (e.g. if recognition is the aim), or some algorithm must be used to fill in the missing elements of the range image. Horn[37] addresses the latter problem.

There are a number of variations on the classical stereo vision paradigm, like the use of multiple cameras, choice of different similarity functions, different features etc. For an overview, we refer to reference [27].

2.3.4 Active stereo vision

Passive stereo vision as described in the previous section has two major drawbacks: the computational cost of establishing correspondence, and the sparseness of the depth information found. If one of the cameras is replaced by an active device like a laser or another kind of projector, both can be overcome. The principle of this process is illustrated in Fig. 2-4. Points in the scene are labelled uniquely by the laser, so the corresponding point in 3-d can be calculated without having to search for correspondence. Furthermore, as any point in the scene can be marked, depth information is not restricted to feature points. Of course, the problem of occlusions is fundamental to triangulation methods and cannot be solved in this way.

There are two main possibilities in active triangulation: a laser can be used to indicate a point or rather a line in an otherwise dark scene, or a pattern can be projected over the entire scene labelling all points simultaneously. The first method requires a fast acquisition sensor because

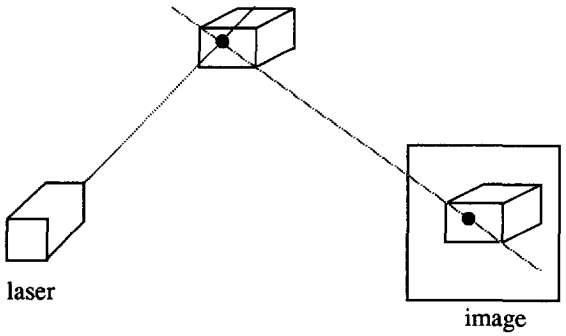


Fig. 2-5 The principle of active triangulation. One camera is replaced by a laser or similar device which sends out a ray of light of known geometry that uniquely labels a point in the scene. By locating that point in the image, the corresponding 3-d point can be calcullated immediately.

many points or lines have to be measured. The second method usually requires several patterns in order to label each point uniquely, but is still much less demanding with that respect than the first one. Shirai[62] describes both methods in more detail.

The major disadvantages of active triangulation are that the scene must be dark by itself, and projecting a pattern on it must be allowed. This means that other vision systems will be influenced by it, and reflecting objects can cause problems. Also, there is the additional task of constructing a light source capable of producing the desired pattern with known geometry at the desired resolution.

2.3.5 3-D Images

In some fields, it is possible to acquire image information for complete 3-d volumes. These include such different fields as computed tomography, magnetic resonance imaging and confocal microscopy. Although these fields are rather distant from the main stream of computer vision, they are mentioned here for completeness. With the exception of confocal microscopy, these methods do not use optical information. Here, they are mentioned for completeness.

2.4 Recognition methods

2.4.1 Introduction

Pattern recognition traditionally [6,32] deals with vectors of features, which are considered to represent observations of objects belonging to certain classes (indicated by labels). Within each class, there is a certain distribution which is either explained by variations in the objects of a class or by noise in the observation. The task of recognition is to correctly assign a label to a new feature vector. This can be based on either learning sets of feature vectors (whose labels may be known), or on knowledge of the distribution of vectors belonging to each class (a

model). Techniques used were mostly based on statistics (statistical pattern recognition). Using these however, knowledge about the structure of the objects to be recognised could not be represented, one was mostly restricted to the use of global features. In complicated scenes, global features are hard to establish. Syntactic models based on a linguistic description of the classes had the disadvantage that the incorporation of noise appeared to be difficult.

Later, other representations of observations have been proposed which allow more of the structure and local features to be represented, such as structural descriptions[35] and attributed graphs[7,16,63]. Other ways are used here to handle the variation in observations, such as (inexact) graph matching techniques. These often lead to a distance or a similarity measure. Although these representations are clearly more powerful in expressing properties of objects, the methods cannot be compared very well in statistical terms because of their different ways of evaluation. Only for specific tasks, performances can be compared.

Recently, in computer vision, the fact has been recognised that the variation in feature values is often caused by one or more underlying parameters. For instance, the way a 3-dimensional object is observed varies as a function of its position and orientation[41]. In fact, in a lot of applications, these parameters are part of the desired output result! This is only possible in those cases, where the variation as a result of the parameters is significantly larger than the variation as a result of noise, in other words: the signal-to-noise ratio must be significantly large. Most recent work in computer vision uses some kind of structural representation.

With computer vision however, a wild growth of methods has arisen. As Grimson[33] remarks, it is impossible to write a complete and up-to-date overview of the field. Therefore, in this section only an outline can be given. A good overview can be found in [33].

2.4.2 Recognition as a search problem

Recognition of objects amounts to establishing a correspondence between the local features of an observed object and those of a model. In other words, pairs must be found of features in observation and model that correspond under a certain geometrical transformation that represents the position and orientation of the observed object. In [33], three different kinds of correspondence are described:

- Selection: what subset of the data corresponds to the object?
- Indexing: what model corresponds to the data subset?
- Correspondence: what individual model features correspond to each individual observed feature?

Each of these correspondences entails some kind of search problem. Different search algorithms can be applied to solve each. Besides, the search can be performed in the space of all possible transformations, the space of all possible correspondences, or in a mixed approach. These are described below.

2.4.3 Correspondence space search

If the search space consists of all possible feature correspondences, it can easily become very

large. If there are m model features and s observed features, the number of possible correspondences becomes of the order of m^s. Clearly, careful pruning is required to reduce the search complexity to an acceptable level[7,35].

2.4.4 Pose space search

If the search is performed in the space of all possible transformations (the pose space), the transformation is searched which gives the best correspondence between model and observation. An example of this is the generalised Hough transform[8], where the pose space is divided into discrete volumes. Each pair of model and observed features increases the support for a number of volumes of pose space. After all pairs have been examined (which requires m.s examinations), the transformation corresponding to the most supported volume is chosen.

The problem with the generalised Hough transform is to find a good compromise for the number of discrete volumes in pose space. A small number leads to high speed but low accuracy, whereas a large number gives an accurate result but a low speed.

2.4.5 Mixed approach

A mixed approach exists in which a limited search in correspondence space takes place in order to compute a small number of complete transformations. This gives us a number of points in pose space, each of which is a hypothesis about a possible presence of an object in the data set. Each of these hypotheses is now tested by evaluating additional evidence for each hypothesis. This usually means transforming the model accordingly and look for additional model features in the data set. This approach is often called *hypothesis generation and testing*. Grimson[33] lists a number of references that all use this approach.

3 The Delft Intelligent Assembly Cell

In this chapter, a description will be given of the DIAC project. This project has influenced the work in this thesis in two ways. First, the stereo vision and recognition tasks described in chapters 4 and 5 have been designed for the objects to be used in DIAC. These objects are described in this chapter.

Second, the systems carrying out these vision tasks should be incorporated in DIAC. The problem that arises here is that of the use of several different vision systems in one larger environment. This is a problem that will become more frequent as vision becomes more and more integrated in modern industrial projects. Work on the use of differing vision systems in one environment is also reported in this chapter, although it should be noted that, as the DIAC project is unfinished, no definitive quantitative results can be given.

The latter part of this chapter includes a description of the verification system that is also being developed as a part of the DIAC project†.

Section 3.1 outlines the project. In section 3.2, the vision tasks in DIAC are described. Chapter 3.3 lists requirements of the interface between the vision systems and the rest of DIAC. In the remainder of the chapter (sections 3.4 - 3.9) such an interface is proposed.

3.1 About DIAC

3.1.1 Introduction

The DIAC project[44] is a five year project with the aim of building an flexible intelligent robot cell, where robots would be able to assemble small product series. Flexibility here means that upon the introduction of a new product, the cell can automatically be reconfigured to assemble the new product. Where fixed (or inflexible) robots can be used to assemble mass products effectively, the investment in automation for small series can only be justified if the adjustment to new products can be made easily. This requires the ability to derive all sorts of parameters, ranging from sensor settings to the order of assembly, from the product model itself.

Intelligence in the robot cell is required for unmanned production. If anything goes wrong during the night, when no operators are available, the cell should be able to recover from the problem itself. This may take some time, but recovery after a quarter of an hour is still better

†. The cooperation with Dave Bierhuizen on sections 3.4 - 3.9 and especially his contribution to section 3.7 is gratefully acknowledged.

than the loss of a whole night of production.

The project officially started in January, 1988, and will be finished by the end of 1992. Four faculties of Delft University of Technology are participating: Mechanical Engineering, Applied Physics, Electronical Engineering and Informatics. Eight research groups are involved, and at the end of the project fifteen Ph.D. students (mostly A.I.O.s) hope to receive a doctorate on work that is completely or partially in the project.

Organizationally, the project has been split up into six different groups, so-called packages. Packages two until six each have their specific tasks in preparing systems and subsystems to be placed in the cell. For instance, package three deals with the vision systems involved. The exception to this is package one, which had the task to determine the cell architecture. Initially, this work was done by a team including all Ph.D. students. As the architecture became clear, activities of package one were limited to actions of the project manager and configuration manager[53].

The DIAC project is part of the SPIN-FLAIR2 stimulation programme of the Dutch ministry of Economical Affairs. As the successor to the FLAIR project (FLexible Assembly and Intelligent Robots), it became a project sponsored by SPIN (Stimulerings Project Informatica Nederland). The project focuses on the design of the Delft Intelligent Assembly Cell, hence the project name.

3.1.2 Architecture of DIAC

The architecture of the cell, as proposed in[53], can be explained in brief using Fig. 3-1. Four levels are identified:

- The top level, the cell master control system that coordinates all tasks.
- The second level, the processes involved in planning and control. This involves a number of steps ranging from production planning (where assembly plans are generated upon the specification of a new product) to exception handling (active when anything goes wrong during the actual production).
- The third level defines an abstract set of (so-called 'virtual') sensors and actuators. Essentially, three abstract properties are defined: location, identity and quality. Each virtual sensor is a system capable of either sensing location (determining the position and/ or orientation of an object with respect to a reference), sensing identity (establishing a logical link between an object at a known location and the model of that object) or sensing quality (judging the matching of sensed features and model features of an object). Similarly, each virtual actuator is able to either change location (move an object), change identity (assemble two parts into one subassembly) or change quality (correct assembly errors).
- The fourth level contains the physical sensors and actuators. These include the robots, sensors (including vision systems), transport and feeder systems. The robots are actually seen both as a coarse-motion subsystem and as a fine-motion subsystem because

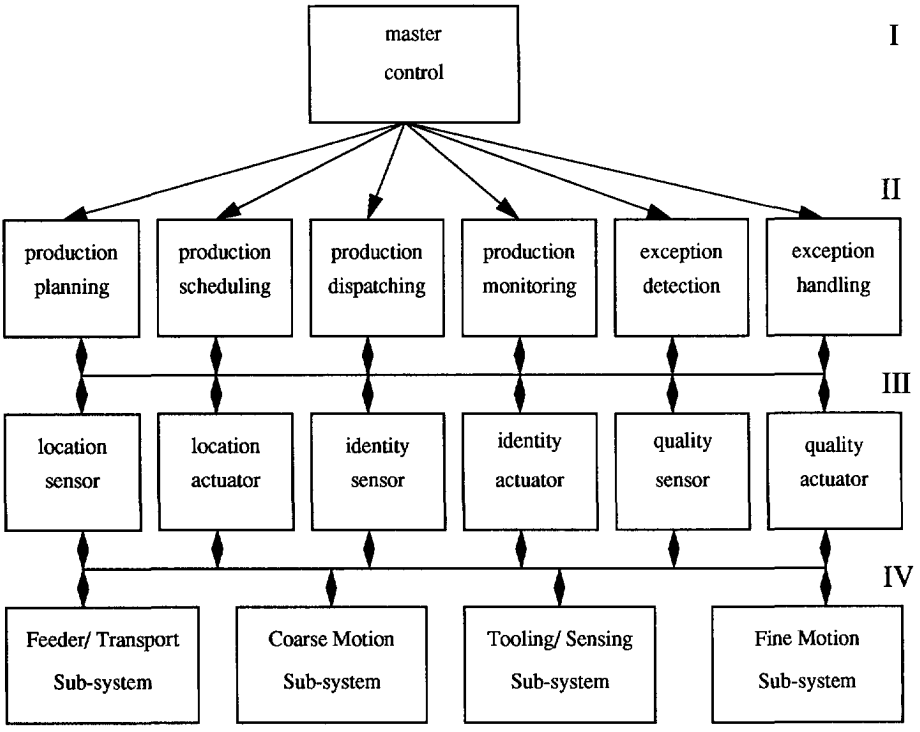


Fig. 3-1 The architecture of the cell (after [53]).

these tasks are completely different with respect to robot control and collision avoidance: while coarse motion moves from one location to another without hitting anything, fine motion for assembly generally means move until you touch something or rather while touching something.

3.1.3 Description of DIAC

A description of the cell layout can be found in [53] and Fig. 3-2. The cell is built around two different robots: one Bosch scara robot with four degrees of freedom, and one ASEA antropomorphic robot with six degrees of freedom. The robots have a shared workspace, and each has its own private area. An internal transport system provides the possibility to move and store parts and subassemblies. Small parts are input through feeder systems, whereas larger parts arrive using an Automatic Guided Vehicle (AGV). This AGV also removes the assembled products. Further included in the cell are various vision systems (see section 3.2).

3.1.4 Research topics

Within the DIAC project as specified above, the following research topics may be identified:

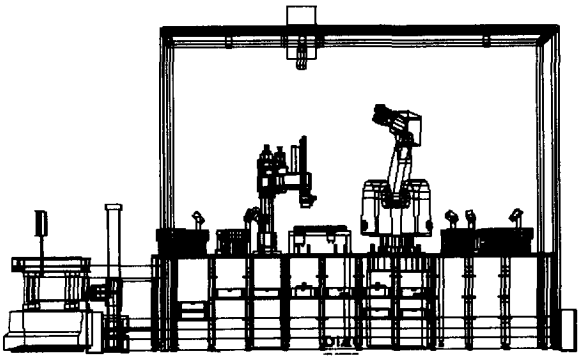


Fig. 3-2 Layout of the Delft Intelligent Assembly Cell

- Automatic assembly planning, interface to Computer Aided Design and Computer Aided Manufacturing (CAD/CAM).
- Planning of collision-free paths for multiple robots, control of robots following these paths.
- Planning, scheduling and dispatching of all tasks within the cell.
- Design of fast and powerful vision systems for identification and inspection (see section 3.2).
- Development of a natural language operator interface.
- Error analysis and exception handling for the cell.

3.2 Diac vision tasks

The following physical sensor tasks within DIAC are performed by vision systems:

- Object Identification. As they enter the cell, parts must be identified and their position and orientation with respect to a reference must be determined. Note that this does not apply to all parts: screws and such are provided by a special feeder. However, the larger parts arrive on a pallet from the AGV. For this task, sets of cameras are provided around both entry points. Two vision systems are available: the verification system developed by Bierhuizen (see section 3.7) and the recognition system described in chapters 4 and 5 in this thesis. Their respective tasks are discussed in sections 3.4 and further.
- Scene Inspection. The robot workspace is continually checked by a structured light

vision system[59] that acquires a height map of the scene. This image can be used to check collision free paths and, more generally, whether the scene corresponds to the predictions made by the scheduler.

- Part Inspection. A hand-held sensor using laser triangulation and a position-sensitive device (PSD)[64] can be used to measure dimensions of subassemblies in order to check the assembly.

For the object identification systems, flexibility is again a requirement. The set of objects to be recognized is not completely specified at the time of design. Instead, the systems should be reconfigured automatically whenever the set of parts changes. However, for the purpose of evaluation of DIAC, two test products have been designed that are sufficiently challenging for assembly planning. These products are also assumed to be representative of the class of objects that need to be recognized. They are shown in Fig. 3-3. This figure has been obtained using the models extracted from the Product Data Model[51] on behalf of the vision system (see 7.2). They have been drawn at camera resolution by the vision system. The following characteristics can be observed:

- The objects consist of cylindrical and prismatic primitives, as a result of which only straight lines and circular arcs are present (which result in straight lines and ellipses in the drawing).
- The objects are mostly metal, with reflections and places with low contrast.

-Object sizes range from 1 to 15 cm.

Of the 31 parts that are required to build the two products, 14 are small parts (such as screws) that are provided by special feeders. Two parts are such that they have no stable position and therefore cannot be transported on a pallet without special support, making recognition and pose estimation as they enter the cell a trivial problem. 18 parts remain to be recognized (see Table 3-1).

Table 3-1 Summary of the Diac parts

kind of object	objects listed
Small part	1,2,3,6,12,13,19,20,22,23,25,26,30,31
Special part	5,15
Part to be recognised	4,7,8,9,10,11,14,16,17,18,21,24,27,28,29

3.3 Interface between the recognition system and the rest of the cell

At level 4 of the DIAC architecture, two vision systems are available to find the identity, position and orientation of objects on a pallet. Typically, these produce a list of all the objects recognized on the pallet, with their respective positions and orientations (relative to a reference). At level 3, this must be translated into the virtual location and identity sensors. The informa-

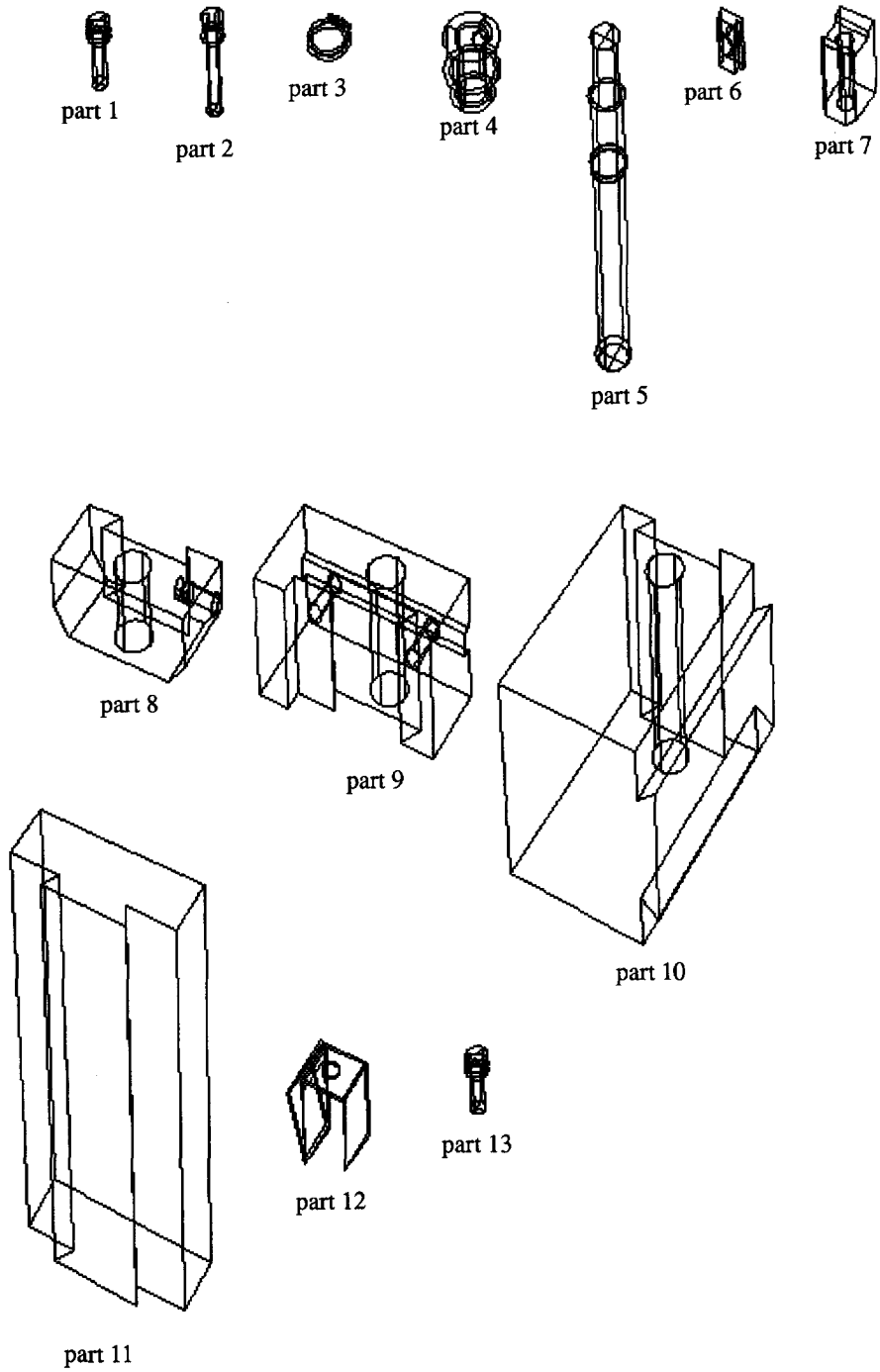
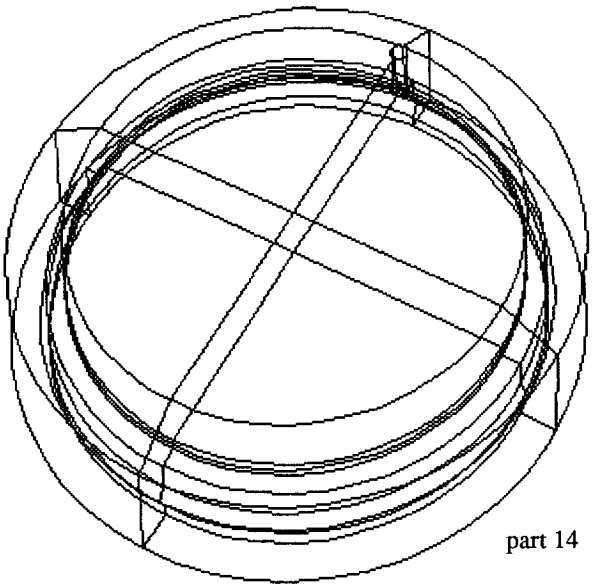


Fig. 3-3 Parts of the test products that need to be recognised, drawn at camera resolu-



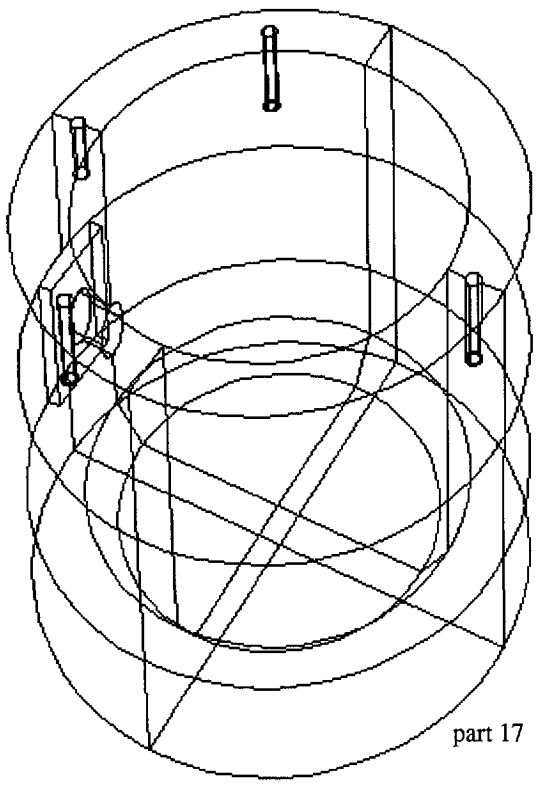


Fig. 3-3 (continued)

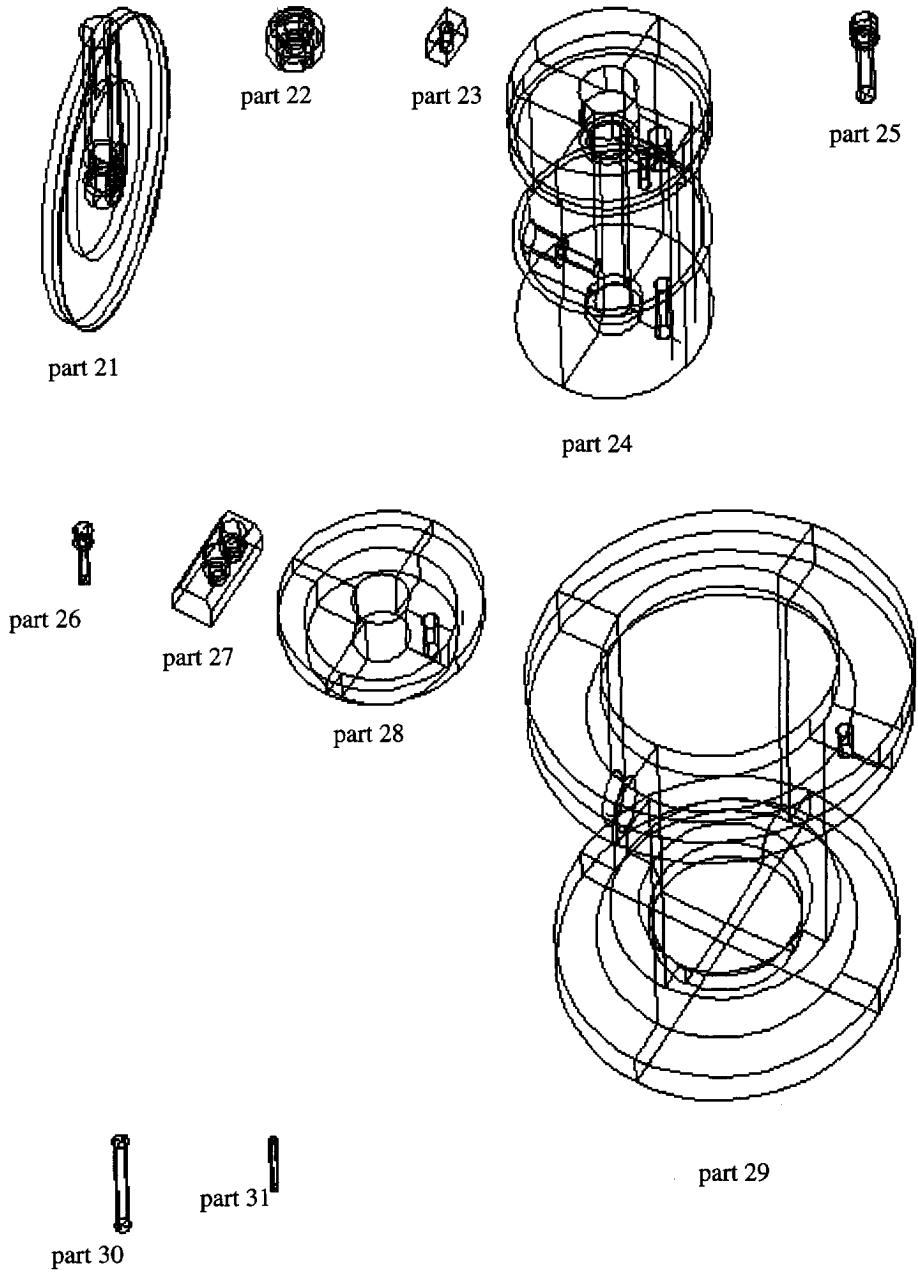


Fig. 3-3 (continued)

tion obtained by the sensor systems must be presented in a consistent way. Furthermore, planning requires that an estimate can be given of the cost (time taken and resources used) of a vision task. This means that a layer of software must be present that integrates the results, monitors the costs and decides between systems in a meaningful way. One possibility of the virtual sensor concept is that it is possible to treat different physical sensors as alternatives, the choice between them being based on cost alone. The approach proposed here is to treat them as complementary. This is described below.

3.4 Verification vs. recognition

The current developments in the field of automated assembly systems show an increasing interest in systems that are flexible in both CAD based product design and CAD based assembly. One of the main steps undertaken here is the integration of the design system with the manufacturing system in order to obtain automatic generation of assembly strategies. For the identification and localization of assembly parts vision appears to be an inexpensive solution. To be a profitable tool in the sense that it is flexible and can be fully integrated in the assembly system, it can be expected that in analogy to the manufacturing system the integration with the CAD system is an important goal. A study by Bhanu[10] supports this trend in vision research.

Another requirement is speed and robustness. The first requirement is met by using a verification system that generates hypotheses based on outside information and common object positions and accepts those if sufficient support can be found. However, this system fails for all exceptional cases. These require a more general system. For this purpose a recognition system is used which tries to find the best matching model from the set of all possible models.

In the remaining sections of this chapter an object identification system is proposed, that is composed of a verification- and a recognition system in order to meet respectively the speed and robustness requirements. Switching from verification to recognition is performed by a control mechanism that minimizes the overall error rate and the computation time. In section 3.5 the vision system's architecture is described globally. Section 3.6 describes the control strategy and the formulation of the goals. The verification system is described in more detail in sections 3.7, whereas the recognition system is described in chapters 4 and 5 of this thesis. Section 3.8 describes experiments on minimization of the overall system costs. Section 3.9 gives the conclusions.

3.5 Overview of the proposed two stage approach

In this section, a two stage approach is proposed to the integration of the vision systems in the virtual sensors for identity and location (see Fig. 3-4):

For maximum speed, a hypothesis is set up about the identity and pose of the object. This may be based on outside information e.g. from the device that filled the pallet, or on an image from a camera that has an overview of the scene. Some alternative stable positions may be considered. This hypothesis is then verified, using characteristic features of the object. Acceptance of the hypothesis means that the identity, position and orientation of the object are known.

If verification fails, a more general recognition system is used. This makes use of general fea-

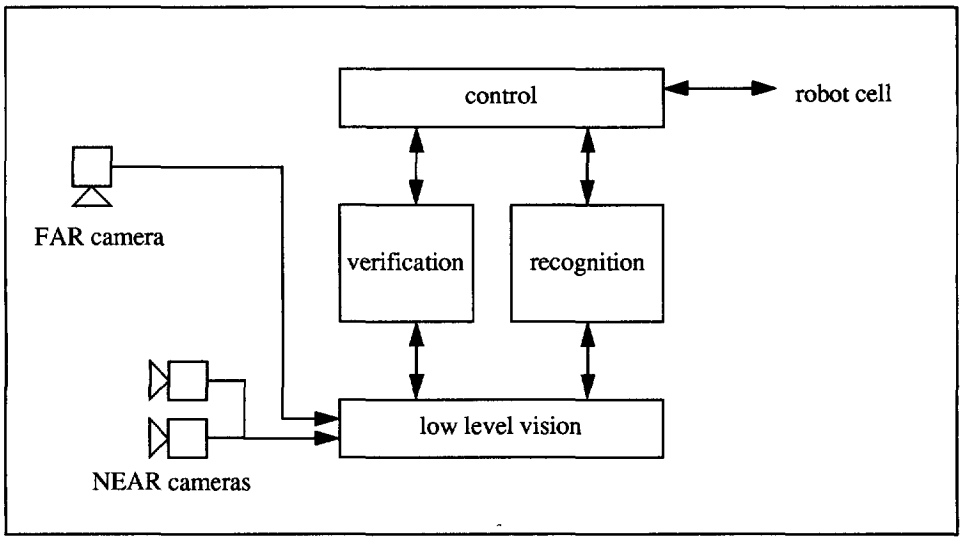


Fig. 3-4 Overview of the proposed vision system

tures, and takes into account a set of objects. This is faster than verifying all possible hypotheses, because no characteristic features have to be generated for each hypothesis. It is slower than the simple verification of one hypothesis, but it can solve more situations.

Verification means that the observation is only compared to one model, as opposed to recognition, which chooses the best model after comparing the observation to all models. In order to make the combination of the two work, it is necessary that verification does not accept false hypotheses. In case of doubt, recognition should be used.

Furthermore, some other components are needed:

The camera module is shared by both systems. It contains an overview camera called FAR, which may do a quick reduction of the set of possible objects, and do a first pose estimate, and a multiple view camera system called NEAR, which is closer to the scene and is used for accurate verification and recognition.

The control block should consider which vision system is being used for scene interpretation, switching as necessary. It should also communicate with the rest of the robot cell, extracting models from the CAD database and returning information about the scene. The control strategy will be described in detail below.

3.6 Control strategy

For the control module, we will simplify the verification system to a system with one parameter that can be controlled by the control module. Verification is able to estimate its execution time at the beginning of a job, and is able to update that estimate as new information becomes available. Similarly, the recognition system is simplified to a one-parameter system that can estimate its execution time.

The control strategy concerns two primary decisions. By default, we start by verifying the most likely hypothesis. If the result of verification is insufficient (to be decided by the control module), the recognition module is activated. Furthermore, if the predicted execution time of verification exceeds that of recognition, recognition is also activated. For ease of the discussion, we will treat these as independent decisions. Both will be described in detail below.

3.6.1 Result optimization strategy

Both verification and recognition return values for the "goodness of fit" of their resulting identification and pose, F_v and F_r respectively. Each of these is normalized, a value of 0 indicating no fit and 1 indicating a perfect match. In both cases, we accept a result if its fit value exceeds a certain threshold T, in other words verification is successful if

$$F_v > T_v \tag{2}$$

and recognition is successful if

$$F_r > T_r \tag{3}$$

These thresholds are to be determined by the control module, based on the criteria below:

Both verification and recognition have three possible results: correct classification, incorrect classification and rejection. In case of a rejection by verification, recognition is activated. The fraction of rejections by verification is a function of the verification threshold and can be written as $f_{r_v}(T_v)$. Similarly, for the fraction of errors made by verification we may write $f_{e_v}(T_v)$. For recognition, these functions are $f_{r_r}(T_r)$ and $f_{e_r}(T_r)$, respectively.

We need to assign costs to the use of the verification system, the use of the recognition system, an incorrect classification by either system, and a rejection by the recognition system. The cost of the use of each system is mostly determined by the execution time of each. The costs of incorrect classification and rejection must be determined by the cell as a whole, and cannot be found by vision alone. If the costs of verification, recognition, rejection and error are c_v, c_r, c^r and c^e, respectively, the total expected cost of object verification C is

$$C = c_v + f_{e_v} c^e + f_{r_v}(c_r + f_{e_r} c^e + f_{r_r} c^r) \tag{4}$$

or

$$C = c_v + (f_{e_v}(T_v) + f_{r_v}(T_v) f_{e_r}(T_r))\, c^e + f_{r_v}(T_v)\, c_r + f_{r_v}(T_v) f_{r_r}(T_r)\, c^r \tag{5}$$

In order to optimize total performance, we must minimize this cost as a function of both thresholds. Note that we expect

$$c_v < c_r \ll c^r \ll c^e \tag{6}$$

Using these relations and constraints, the thresholds T_v and T_r may be found. The necessary functions $f_{r_v}(T_v)$, $f_{e_v}(T_v)$, $f_{r_r}(T_r)$ and $f_{e_r}(T_r)$ must be estimated by the control module. This must be done by monitoring the actual performance of the system, starting with a sufficiently accu-

rate estimate. Estimating $f_{e_v}(T_v)$ and $f_{e_r}(T_r)$ will be especially hard, because they are supposed to be very small and identification errors can only be detected during assembly by the cell error manager which is prone to errors itself. This happens for instance when a part does not fit. It is assumed however, that these estimates can be made, assuming that all objects belong to a class of parts with more or less similar properties.

Costs c_v and c_r result from the vision system itself and will mainly be determined by execution times, whereas c^r and c^e are dependent on the whole robot cell and cannot be supplied by the vision system alone. For instance, c^r may increase as the reject buffer fills.

3.6.2 Time optimization strategy.

During the on-line operation the verification strategy monitors the analysis results and estimates the time required to perform the verification. Based upon this it is decided whether to remain in the verification stage or to switch to recognition. The system is placed in two possible scenarios: A limited execution time and minimization of the total execution time.

At every time t the expected verification time $\tau_{ver}(t)$ and the expected recognition time $\tau_{rec}(t)$ are known. Here $\tau_{ver}(t)$ is an estimate function of the time required, which is based on the number of edges to be detected. The expected recognition time is actually a constant during the verification stage. To perform the vision within a limited time (dictated by the cell) the analysis result must be available within time $t = t_{max}$. Thus a decision must be made at time $t = t_{max} - t_{rec}$. At this time a switch to recognition takes place if:

$$\tau_{ver}(t_{max} - t_{rec}) > \tau_{rec}(t). \tag{7}$$

In case of minimization of the total execution time a decision is made at any time τ. Thus a switch from verification to recognition at this time is decided for if:

$$\tau_{ver}(t) > \tau_{rec} \tag{8}$$

3.7 The verification system†

3.7.1 Overview.

The verification algorithm basically consists of two steps: the FAR and the NEAR analysis. The FAR analysis is performed first in order to rapidly generate object hypotheses based on 2D image processing. The FAR analysis outputs a coarse 3DPO (3D Position and Orientation) estimation. Consequently all NEAR steps can be performed by hypothesizing and verifying edges at predicted locations (regions-of-interest). The presence of an edge is verified by using a stereo image pair. Since edge detection is a time consuming operation it is worth reasoning about which edges are to be selected to reach the verification goals. This forms the core of the verification strategy.

The NEAR step is decomposed into three steps: Stable position/orientation discrimination,

†. This system is being built by Bierhuizen and will be reported in his Ph.D. thesis. Earlier results have been published in [19] and [12].

object discrimination and object evidence gathering. 3DPO refinement is concurrently performed for the detected edges with techniques similar to those of Horaud[36]. The FAR and NEAR steps are shown in Fig. 3-5. Every step requires the success of the previous step, which

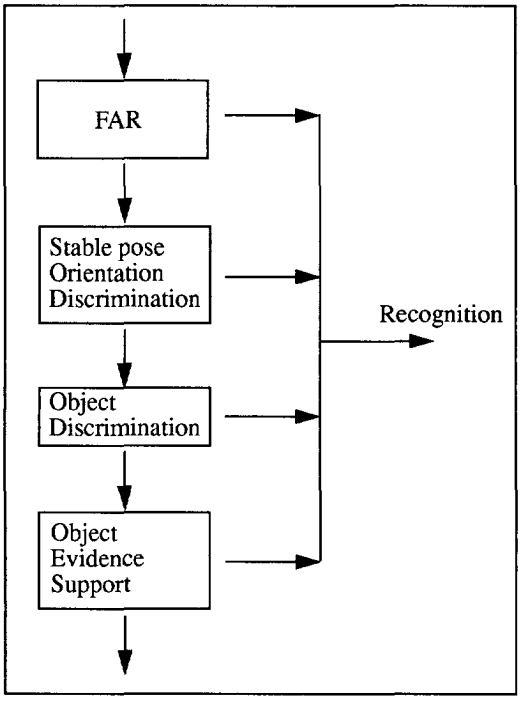


Fig. 3-5 Overview of the verification system.

implies that early escapes to recognition may occur in the presence of errors. A brief description is given below.

- FAR

  The FAR analysis implemented is based on binary image processing. The results are a classification of statistical measures and a 2DPO estimation based on geometrical moments. The used algorithms are fast and simple[46].

- Stable pose/Orientation discrimination.

  The FAR results may contain some ambiguities: Identical views for several stable object positions or symmetry in the view which have to be resolved first before further object verification is performed. The strategy here is a search through a tree that is composed of features that are space variant for different stable position and orientation combinations.

- Object discrimination.

  In this step the object is verified based on features that are distinctive against those of a set

of alternative objects (dictated by the cell task). The alternative objects are here those object hypotheses that cannot be distinguished in the FAR step.

- Object evidence support.

  The object evidence support step pursues a high overall reliability of the object verification by gathering a sufficient number of model-image matches. With respect to the type of edges verified this step is only completing. Thus edges selected for this purpose are not specific.

3.7.2 Control of the verification system.

The verification strategy is initially constructed off-line to minimize on-line computations. This strategy is explicitly represented in a vision database in the form of selected features for all steps. What is essential here is that the verification strategy is opportunistic in the sense that it tries to minimize on-line computations by minimizing feature complexity. However, due to noise and lighting effects segmentation errors may occur which may either make an edge undetectable or give rise to a false detection. The second problem is solved implicitly since the resulting acceptance of a wrong hypothesis inevitably leads to an error in one of the subsequent steps. With respect to the first problem the on-line control of the system is able to find new features that (, once detected,) satisfy the original goals. In this process the time optimization criteria are evaluated

3.7.3 Definitions of object discriminating features.

In order to be able to determine whether the hypothesized object is really present, a set of features (in this case edges) must found that completely discriminates that object from the set of all other objects, given a specific viewpoint. These sets (called edge combinations) can be determined beforehand (off-line). This is described in this subsection in two steps: first we will describe how an object can be discriminated from some other objects using a pair of edges, afterwards this is extended to discriminating the object from all other objects using a larger set of edges.

Edge combinations v_i are defined as subsets of the set of observable edges of object A, $Ea = \{e_1, e_2, .., e_n\}$, with n the number of edges. The edges are observed in a stereo configuration, so all 3-D coordinates are known. For every pair of edges e_j, e_k in an edge combination the 2-dimensional feature vector $[\alpha_{j,k}, \delta_{j,k}]^T$ can be defined, with as elements the angle $\alpha_{j,k}$ and the shortest distance $\delta_{j,k}$ between the edges (see Fig. 3-6). In case the edges intersect or intersect after extrapolating one or both, $\delta_{j,k}$ is zero. The measurements of the feature vectors will be corrupted with noise, so instead of instead of a fixed value for the parameter vector we can expect to measure a value on an interval around that value. For an edge combination, all feature vectors of all possible edge pairs are considered together. So, for each edge combination v_i an uncertainty domain Ua_{v_i} is defined as the union of all intervals of feature vector values associated with edge pairs from v_i.

Ideally, edge combinations of object A exist which have a unique feature vector thus do not appear in other object descriptions. Now v_i discriminates object A from object B, expressed by

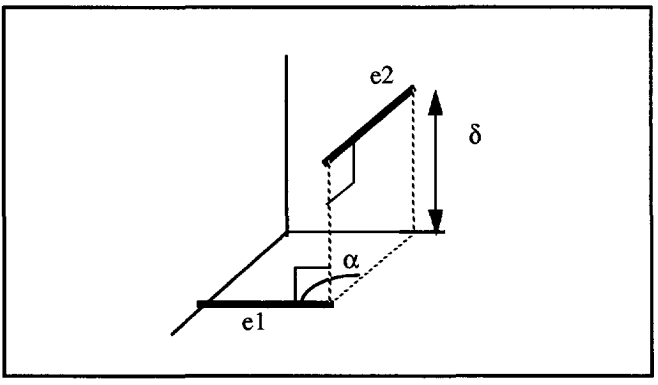


Fig. 3-6 The distance and angle features of an edge combination

the boolean discrimination function $C_{v_i}(A, B)$, if there is no overlap between Ua_{v_i} and all Ub_{v_j}, or:

$$C_{v_i}(A, B) \Leftrightarrow Ua_{v_i} \cap Ub_{v_j} = \varnothing, \forall v_j \subset Eb. \tag{9}$$

In general an edge combination consisting of a single edge pair will only discriminate object A from a subset S_{v_i} of the complete set of alternative objects *S*. An edge combination that discriminates A from the complete set will then have size greater than two. In order to construct this complete discriminating edge combination, the following property is given:

$$\text{if } |v_i| > 2 \text{ and } C_{v_i}(A, S_{v_i}) \text{ then it follows that } S_{v_i} = \bigcup_j S_{v_j}$$

$$\text{for all } v_j \subset v_i \text{ with } |v_j| = 2 \text{ and } C_{v_j}(A, S_{v_j}), \tag{10}$$

or in words, an edge combination v_i of size greater than two discriminates object A from an object set S_{v_i} which is the union of all discriminated subsets S_{v_j} which follow from the subset edge combinations v_j of size two. Thus to construct an edge combination that discriminates A from an alternative object set we may build it from edge combinations of size two.

3.8 Experiments

Experiments are performed to show that minimization of the overall cost function is possible by using data from two working systems.

For the experiments we choose a two class classification in which the classes are two models of DIAC assembly parts, parts 10 and 11 in Fig. 3-3. From each part 10 image triples (FAR and a NEAR pair) were acquired, each with the parts in a different pose or orientation. The parts are shown in Fig. 1.

For both systems variable thresholds for the goodness of fit are used to estimate the relevant error- and reject fractions. These thresholds are chosen at the interval (0..1) having an equal

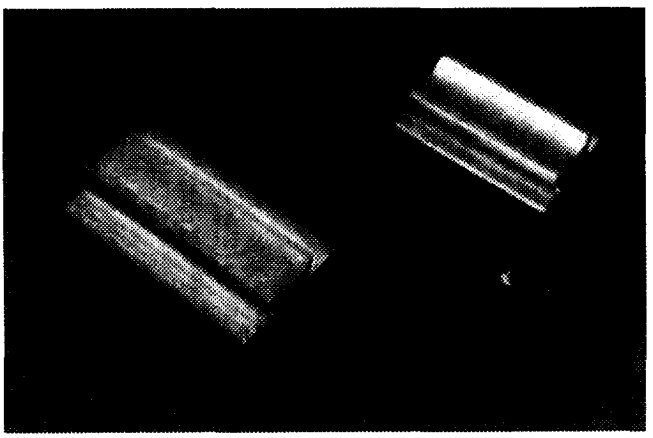


Fig. 1. Both objects considered in the two-class experiment.

distance of 0.1. In the verification system the FAR classification step is deactivated, since this classification step is not very interesting here. Further, for a hypothesis acceptance at least one discriminating feature has to be found and for the object evidence support step the goodness of fit threshold has to be exceeded. In the recognition system the goodness of fit threshold has to be exceeded for acceptance of the best fitting model.

The reject- and error fractions as a function of the thresholds for verification and recognition are shown in Fig. 3-7 and Fig. 3-8.

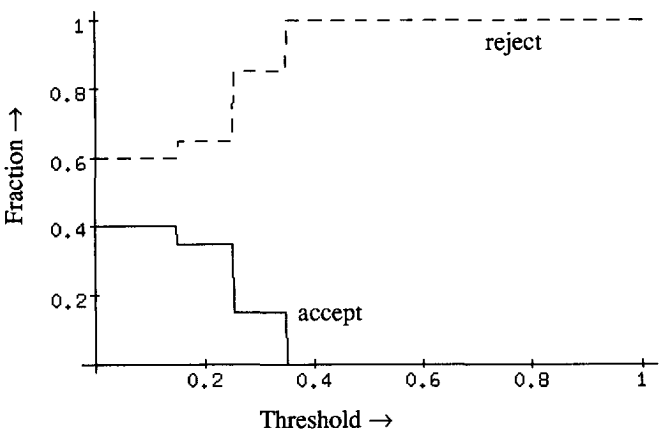


Fig. 3-7 Accept and reject (dashed) rates for the verification system as a function of the verification threshold. the error rate for this system in this experiment was 0.

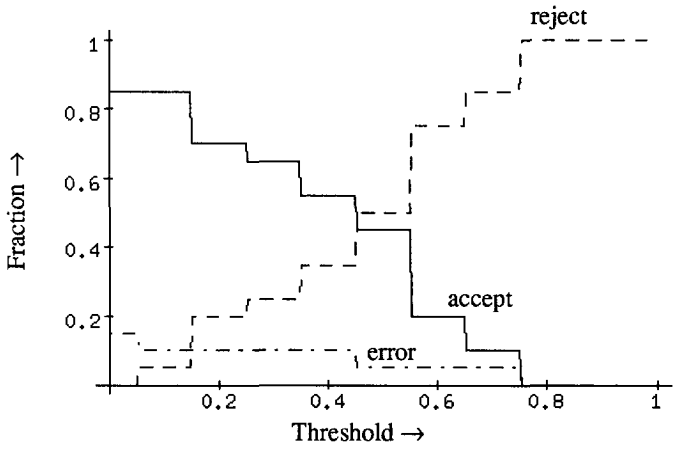


Fig. 3-8 Accept, reject (dashed) and error (dotdashed) rates for the recognition system as a function of the verification threshold.

The costs are expressed as the time (in seconds) required to perform the action (verification- and recognition costs) and the delay of a classification result (reject- and error delay). They are chosen as follows:

$c_v = 5$, $c_r = 25$, $c^r = 400$ and $c^e = 1000$.

The value for rejection is based on the assumption that parts are partly offered in kits (sets of parts necessary for one product), so that it is expensive to wait for another, identical part. The cost of error is still higher, however.

The cost function as a function of both thresholds is shown in Fig. 3-9. Minimization of the overall cost function results to an optimal verification threshold $F_v = 0$, an optimal recognition threshold $F_r = 0.1$ and an overall cost $C = 71$. The reason for a zero valued optimal verification threshold is that the fraction of errors is zero for all thresholds in the experiments.

3.9 Conclusions

In this chapter, the DIAC project has been described. The tasks to be performed by DIAC define the demands on the recognition system described elsewhere in this thesis. Parts have to be recognized and their position and orientation determined in a data driven way. The parts consist of straight lines and circular arcs.

A two stage system for object identification in DIAC has been proposed. The first stage (the verification system) tries to verify a hypothesis based on outside information and common object positions. In case of insufficient result, the more general recognition system is activated. Criteria have been found for switching from verification to recognition. Minimization of an overall cost function leads to optimal thresholds for both verification and recognition.

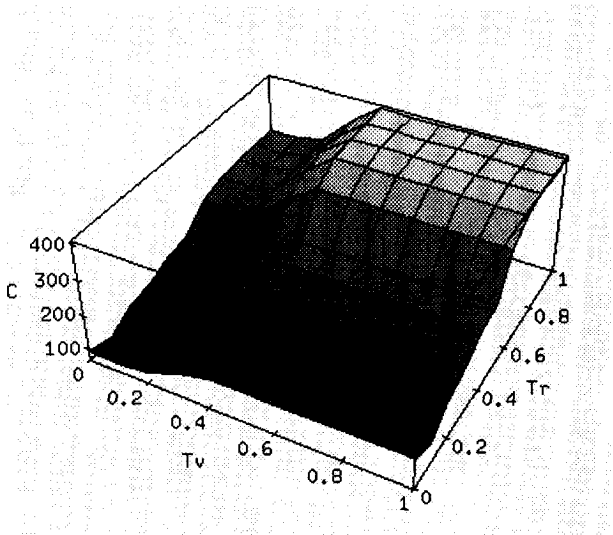


Fig. 3-9 Total cost as a function of both thresholds for this experiment.

Our current assumption is that the reject fraction of recognition can be determined independently of the reject fraction of verification. Although the systems are based upon completely different principles, this assumption is unlikely to be valid. The reason for this is that a rejection of verification due to a low image quality will inevitably result to a rejection by verification too. On the other hand, a rejection by verification due to a wrong hypothesis (for example a non-stable position) is not related to the classification result of recognition. Therefore a clear distinction should be made between possible causes for rejection by verification, and this should be expressed in the cost function. While this is impossible, the current solution (treating the systems as independent) seems a reasonable alternative.

4 The stereo vision system

In this chapter, the stereo vision system is described that is part of the Diac object recognition system. Its task is to observe the scene, and produce an output wireframe description of the relevant part of that scene. This description should be sufficient for recognition of the relevant objects. It is not necessary though to reconstruct the scene.

This chapter starts with an introduction. In section 4.2, the image acquisition and preprocessing steps are described. Section 4.3 describes the transition from images to 2-d geometrical features, whereas sections 4.4 and 4.5 describe the two main stereo vision algorithms that derive 3-d features from 2-d features. Section 4.6 deals with the calibration of the whole system, and section 4.7 evaluates the stereo vision system.

4.1 Introduction

4.1.1 Why feature-based stereo vision?

A general discussion on stereo vision techniques has been given in section 2.3. They differ with respect to efficiency and sparseness of their output data. For the recognition of objects, a full 3-d reconstruction of the scene is not necessary. Instead, we will require that a sufficiently powerful set of 3-d features has been observed adequately. The features in each of the two images are well-known in advance (see section 3.2). They will be straight lines and ellipses, since they result from straight lines and circles in 3-d. This allows feature-based stereo to be used. From the stereo vision techniques mentioned in chapter 2, feature-based stereo vision should be an appropriate solution for the following reasons:

- Feature-based stereo vision is efficient as it does not need to calculate correspondence for every pixel in the image, but instead only processes a much smaller number of features.

- The result of feature-based stereo is a list of 3-d features, which can be used immediately for recognition purposes. This differs from stereo vision techniques that describe the scene in terms of a $2^1/_2$-d (range) image, that still needs segmentation etc.

There are of course some disadvantages to feature-based stereo:

- If the two corresponding features in both images of the stereo vision pair are not both detected correctly, no 3-d feature will be found. If the features are straight lines and ellipses, missing one feature implies that an entire line or circle will not be found.

- The stereo vision system will be limited to scenes in which these features occur. If objects of other shapes (such as objects with elliptic or B-spline shapes) need to be detected, the stereo vision has to be redesigned. The present system will try to decompose these shapes into

known features.

For the DIAC application, the efficiency arguments in favour of feature-based stereo vision are preferred over the other arguments against it, provided that the stereo vision system is able to detect sufficient features for recognition. Such a system will be described in the remainder of this chapter.

4.1.2 Overview

The position of the stereo vision system in the recognition system is indicated in Fig. 4-1. It processes images, resulting in 3-d wireframes that are compared with model wireframes by the matcher. The image processing takes a number of steps. First, two images are acquired. As the features in the stereo vision system are edges, edge detection is applied to both. Subsequently, straight edges and elliptical edges are separated. To both kinds of edges, a different stereo vision algorithm is applied.

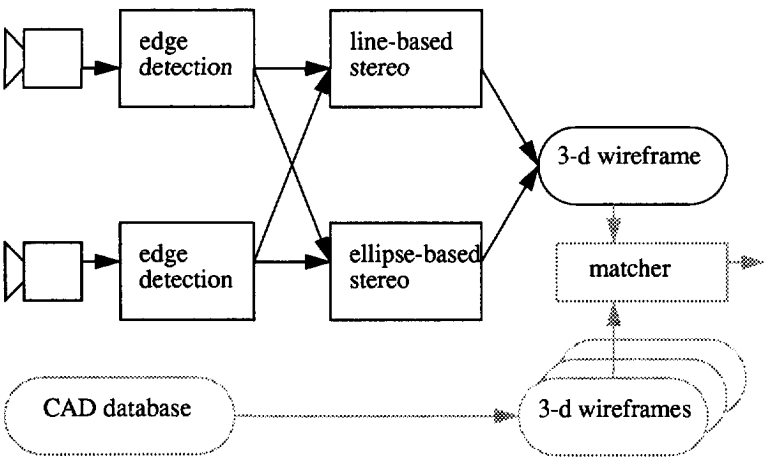


Fig. 4-1 The stereo vision part of the recognition system.

A prototype of the system is running at the Pattern Recognition Group. This will be referred to in this chapter simply as the prototype. A final implementation will be built in the DIAC cell in the near future. Wherever necessary, it will be referred to as the final implementation.

4.2 Image acquisition and processing

4.2.1 Image acquisition

The prototype is organized as follows: two cameras are overlooking the scene that is lighted by a number of fluorescent tubes. Nearby is an ASEA IRB 6/2 robot that is able to grab objects from within the camera area. The robot is also used in the calibration procedure. The cameras grab 512 x 512 pixel images, and are connected to the image processing computers. These are: a 68020-based system running OS-9 (currently only for frame grabbing) and a Sun SPARC

workstation running SunOs (for all image processing). This setup is sketched in Fig. 4-2, and a picture is shown in .Fig. 4-3

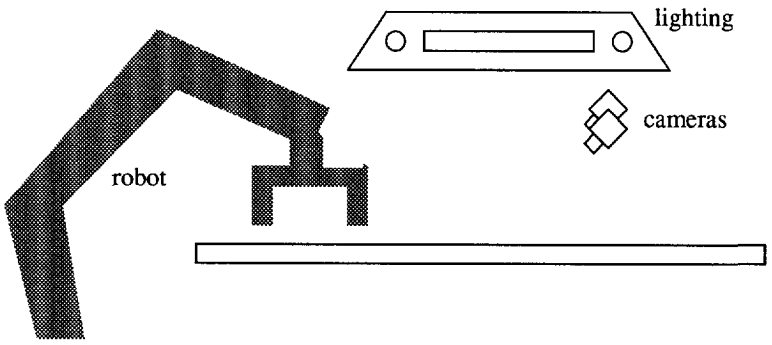


Fig. 4-2 The setup of the prototype.

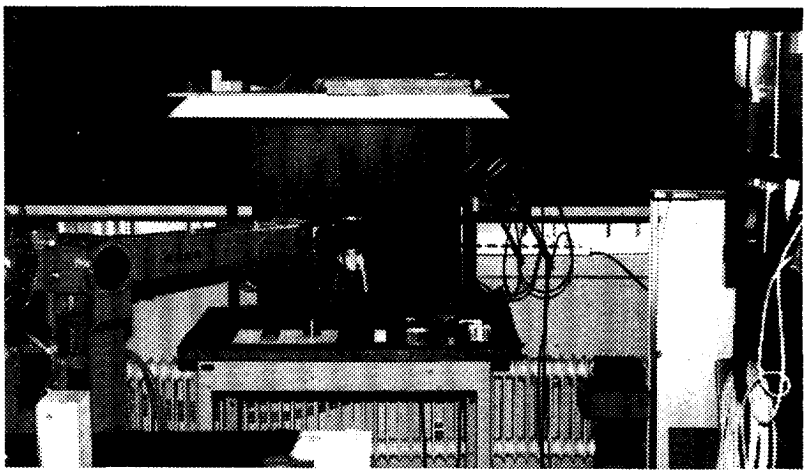


Fig. 4-3 Picture of the prototype.

The final implementation will run on a SPARC/ SunOs - 68030/ OS-9 combination. Imaging Technology (Woburn, Massachusetts, USA) special purpose hardware will be present in the cell and may be used to speed up the processing.

The stereo vision first does edge detection, then the list of edges (represented as chaincodes) is searched for candidate ellipses. If these are found, an ellipse is fit to them. To all edges a straight line fit is applied, splitting up those edges that cannot be fit. The result is a list of ellipses and a list of straight lines for each image. To the ellipses in each image an ellipse-based stereo algorithm is applied, and to the straight lines in each image a line-based stereo

algorithm is applied.

This procedure may imply that edges are treated twice, both by the ellipse and by the straight line procedure. The impact of this is reduced by the fact that few edges give meaningful results after stereo vision for both procedures: real straight lines cannot be represented well by ellipses and although ellipses can be represented by straight lines, this generally happens differently for both images resulting in poor correspondence. The few remaining cases, where an edge in the scene is doubly represented, must be handled by the recognition procedure.

The outputs are lists of circular arcs and straight lines that are combined into a 3-d wireframe representing the scene. We will discuss the separate steps below, using the detailed Fig. 4-4.

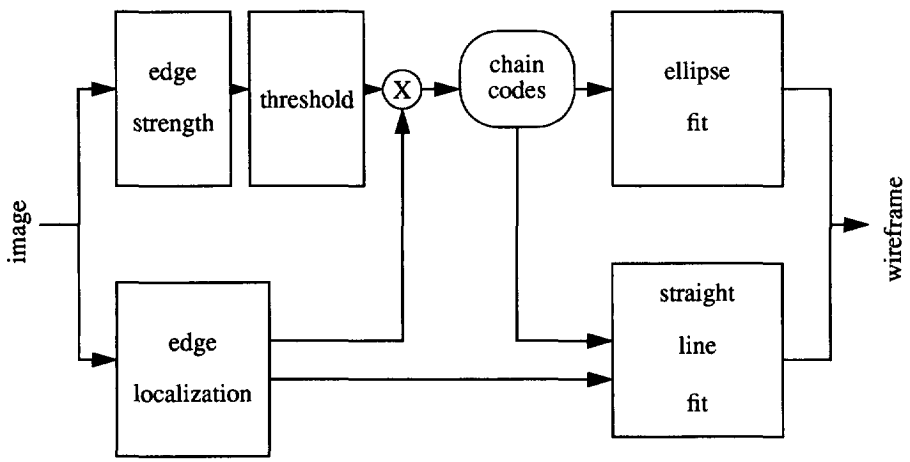


Fig. 4-4 Detailed description of the segmentation procedure.

4.2.2 Edge detection

Edge detection may be defined as the detection of places in images where rapid changes of intensity occur, usually in conjunction with an estimate of the "speed" of this change. It is desirable that these edges are represented by one pixel thick lines. This is generally seen as the combination of two problems: those of finding both the location of an edge and its strength. While edge strength is not critical, the determination of edge location is of importance in this situation. There are two main approaches:

- Detection of the maxima of the gradient of the image. This is relatively hard on a digitized grid. The most popular edge detector of this kind is Canny's[22].
- Detection of the zero-crossings of the derivative of the gradient in the gradient direction. Although this is fairly straightforward, the main problem that arises is the impossibility of dealing with edge junctions, see Fig. 4-5. While the sign of the gradient clearly changes at each separate edge, it cannot change for every edge at junctions of

an odd number of edges. A number of these edge detectors are given in [67].

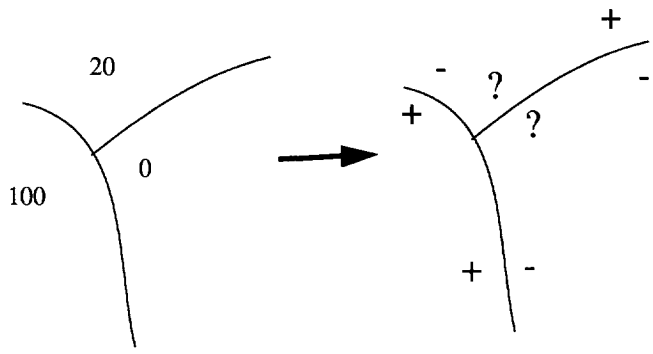


Fig. 4-5 Problems with zero crossings-based edge detection. While the sign of the zero-crossing is clear for each individual edge (the numbers are grey values for each region), edge junctions cannot be described.

There is a vast literature on edge detection, see e.g. [67]. Rather than doing more research in this field, it was decided to pick an available, relatively fast detector, that produces edges that lend themselves well to further processing. The latter requires that the edges are one pixel thick, and few short false edges are introduced. For this purpose, the DYC edge detector[65] has been chosen. Although this edge detector produces contours, a simple binary neighbourhood operation (e.g. one Hilditch skeleton iteration) is sufficient to transform these into a real skeleton.

This edge detector uses the filter by Lee et al.[47] to find edge strength and the zero crossings of the DYG filter[8] to find edge location. The two results are combined and the resulting edges after thresholding are represented by chaincodes. We keep the edge location image for the straight line fitting stage to obtain sub-pixel accuracy at low computational cost. Unfortunately, it is not useful to do this for ellipse fitting as most edge detectors are slightly biased in the presence of curved edges[66].

Both the Lee and DYG filters are implemented using filtering with local minimum and maximum operation. A gaussian blurring is applied beforehand, as in [8]. Summarizing, the edge detection procedure consists of four steps:

(1) Initial smoothing filter a gaussian filter.

(2) Determining the edge strength by applying the Lee filter [47] to the output of (1).

(3) Determining the edge location by the zero-crossings of the DYG [65] of the output of (1).

(4) Calculating the skeleton of (3).

(5) Multiplying the outputs of (2) and (4).

4.3 Segmentation

4.3.1 From edges to chaincodes

The output of the edge detector is an image $e_{i,j}$ still containing grey values, which indicates the strength of the edge. These grey values are zero for non-edge pixels or positive for edge pixels. The edges are one pixel thick. This image is thresholded at a low threshold T_1 and the resulting binary edge image is transferred into strings of Freeman codes, starting and ending at end points and junctions of edges. Along each edge, the grey values $e_{i,j}$ are summed and this sum is again thresholded with a threshold T_2, discarding chaincodes that are as a whole to weak.

This approach is based on the observation that edges may have a local strength below the noise level, but still show up as one edge in the edge detector image. It can be summarized as

$$e_{i,j} > T_1, \sum_{\text{edge}} e_{i,j} > T_2 \tag{11}$$

and the resulting set of edges is a more useful set than when a single threshold had been applied, as illustrated by Fig. 4-6. The effect is similar to that of hysteresis thresholding [22].

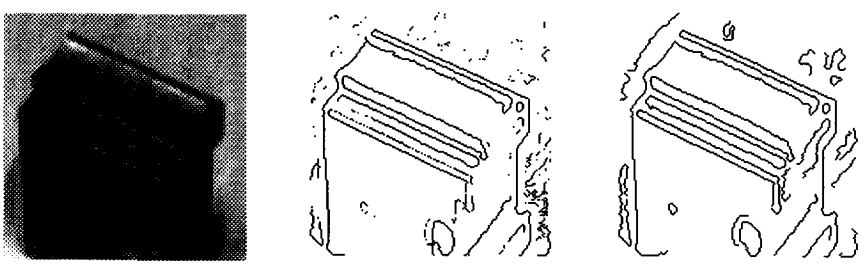


Fig. 4-6 The edge threshold method illustrated. From left to right: original image, edge image after applying a pixel threshold of 6 to the output of the DYC edge detector, image after applying a pixel threshold of 2 and an edge threshold of 100 to the output of the DYC edge detector, as described in the text. The latter method keeps edges connected while suppressing individual noise pixels. For the stereo vision methods used in the system described here, keeping edges connected is of great importance.

4.3.2 Straight line fit

To each remaining string of chaincodes, a straight line is fit using a minimal square error estimator. In this procedure, we apply a first order fit to the edge location image in order to get sub-pixel accuracy. Because we only look at "interesting" points in the image, this can be done at very little computational cost. If the chaincode string corresponds to points in the image that are too far away from the fitted straight line, the string is split at the point of highest distance, and the procedure is repeated. This is illustrated in Fig. 4-7.

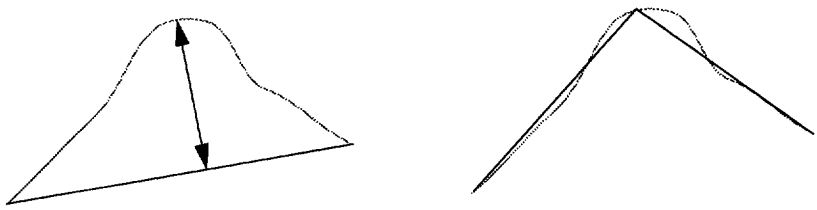


Fig. 4-7 On the left is shown how the maximum distance of the chaincode (grey) to the single straight line (black) is too large. The chaincode is then split up at that point, and the two parts are approximated separately. This procedure is repeated until the distance no longer exceeds the threshold at any point.

Once all the straight lines have been found, junctions are recovered by crossing the fitted lines. Care must be taken to ensure that distorted junctions are not misinterpreted, see Fig. 4-8.

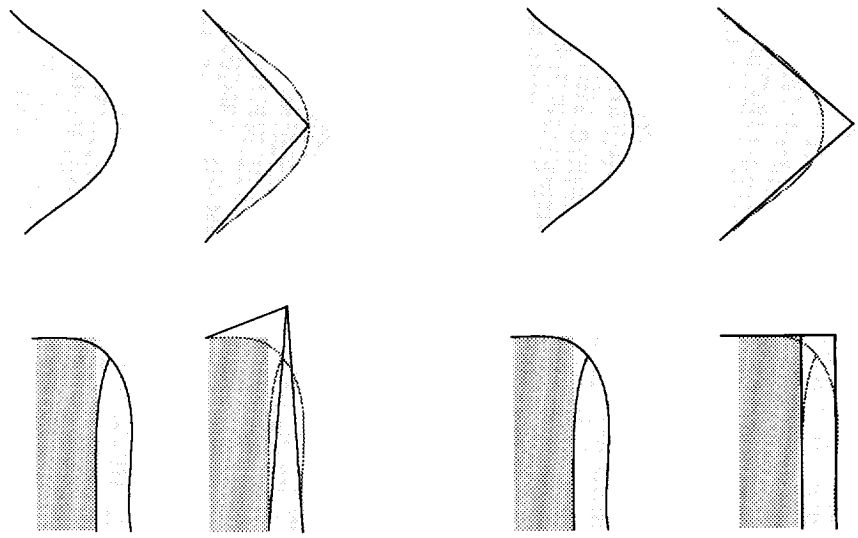


Fig. 4-8 Recovering junctions. Four times the data and the result are shown. Left: wrong approach. Right: correct approach. The top example shows two lines meeting, the bottom one shows three lines meeting, of which two are parallel. In each drawing shading has been used to indicate the grey level.

This figure illustrates two cases where careless calculation of junction points can lead to erroneous results. In the top case a rounded corner is shown. Because of its smoothing property, every edge detector will round corners. For a correct calculation of the corner location, it is

necessary to disregard those edge pixels that are a result of rounding. The second example of Fig. 4-8 shows the case where three lines are meeting, of which two are parallel. If the two lines are close enough, a single crossing point will result which may have a large positional error. The correct result will include a small extra edge linking the two parallel lines.

The result of the straight line fit is a graph, with as its nodes vertices in the image and with its edges representing straight lines in the image.

4.3.3 Ellipse fit

In order to determine which ellipses are present in each image, we resort to fitting, but this requires that the set of points belonging to one ellipse are identified first. We need to split up strings of chaincodes, until each string only contains one ellipse. Accuracy of the fit requires that these strings are as large as possible. Splitting based on the derivative of the curvature at each part of the string does not work very well, because of the eccentricity of some ellipses. Instead, we apply a simple grammar to the strings and split where the grammar no longer applies. This algorithm is simple and fast, and works as long as the smoothing effect of the edge detector is sufficient to deal with most of the noise on each ellipse.

Let c_i be a Freeman code for direction i, and c_{i+1} (c_{i-1}) be the code for the next direction (counter-) clockwise. Let c* denote zero or more, and c+ denote one or more occurrences of c. Likewise, let (c)* be zero or more occurrences of (possibly different) realizations of c. A string belonging to an ellipse can either be written as:

$$(c_i+ c_{i-1}+)^*(c_i)^*(c_{i+1}+ c_i+)^*(c_{i+1})^*(c_{i+2}+ c_{i+1}+)^*... \quad (12)$$

or

$$(c_i+ c_{i+1}+)^*(c_i)^*(c_{i-1}+ c_i+)^*(c_{i-1})^*(c_{i-2}+ c_{i-1}+)^*... \quad (13)$$

These grammars can be interpreted as follows: a straight line of any orientation can be represented by strings of chaincode consisting of at most two different codes, which two codes are determined by its orientation (see e.g. [28]). Eight special orientations can be represented by strings containing only one chaincode. Any curve, for which the curvature does not change sign (such as an ellipse), will monotonously assume a range of orientations. Therefore, all strings representing it can be split into substrings in which either only one chaincode occurs, or two chaincodes alternate, where the order of the substrings depends on the sign of the curvature.

Each parenthesized expression corresponds to an orientation in one of sixteen directions or ranges of directions. In (12), these directions are passed in a clockwise direction, and in (13) counterclockwise. Our splitting algorithm tries both grammars (12) and (13) to find out how far from the start of the string they still apply, and splits the string after the longest of these substrings. This is done for each string of chaincodes, repeatedly if necessary.

To each of the resulting strings of chaincodes, the fitting algorithm is applied. An ellipse in an image with axes x_1 and x_2 can be written as

$$\alpha_0 x_1^2 + \alpha_1 x_1 x_2 + \alpha_2 x_2^2 + \alpha_3 x_1 + \alpha_4 x_2 + \alpha_5 = 0, \quad (14)$$

where $\alpha_{0..5}$ are constants, of which α_0 can be set to 1 without loss of generality. Some constraints apply, see e.g. [8]. Ellipse fitting can be described as finding the set of $\alpha_{1..5}$ that satisfies (14) "best". Different interpretations of "best" are possible, e.g. finding the maximum likelihood estimator (when a noise model is available) or finding the ellipse that the points represented by the chaincode are closest to (using the average, or mean square average). However, in this case an appropriate noise model (taking into account all preprocessing) is not available. Also, minimizing the average distance or average square distance of an ellipse to a set of points is a complex and time-consuming procedure. Therefore, following [30] we minimize the average squared left hand side of (14):

$$\min_{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5} \left(\sum_{\text{all points on edge}} \left(x_1^2 + \alpha_1 x_1 x_2 + \alpha_2 x_2^2 + \alpha_3 x_1 + \alpha_4 x_2 + \alpha_5 \right)^2 \right) \quad (15)$$

which, although somewhat biased towards eccentric ellipses[30], is a good and efficient estimator of ellipse parameters. If the minimal error per pixel is below a threshold, and the set of parameters that minimizes it corresponds to an ellipse (and not another conic section), we accept the ellipse. This threshold has been tuned over several hundreds of images and has been applied successfully to a large number of images obtained afterwards.

As a final step, we go through the set of ellipses found to see whether two ellipses can be merged, keeping the average error over the union of the pixels below the threshold. This is done repeatedly until no more mergers are possible.

4.4 Line-based stereo

4.4.1 Problem description

A relatively fast way to do stereo vision is based on finding corresponding straight line segments, and computing their 3-d equivalent. A summary of related research is given in 4.4.2. In our case, this is made easy by the relatively low complexity of the scenes: they consist of a few objects and the pallet as a background, all edges confined to a small volume. This results in a fairly straightforward algorithm. The main algorithm (described in 4.4.3) allows incomplete edges by using edge orientation as the key feature. This approach fails for certain cases in which no unique 3-d edge can be found. In this case, an alternative algorithm using edge endpoints is used. This algorithm (described in 4.4.4) uses more information. Both algorithms use the same constraints, as described in 4.4.5, and are evaluated in 4.4.6.

4.4.2 Related research

The line based stereo vision approach has been used by a number of researchers. Their work mainly differs in the way they find the best global correspondence. In [52], an iterative procedure is used to minimize a disparity criterion in a certain neighbourhood. In [4], a search procedure over the tree of possible matches is used to minimize disparities. In [59], only local information is used initially, and one iteration over neighbouring segments is used to correct

the list of correspondences somewhat.

In the work described here, we have chosen to use local information only. Using the strong constraints that are made possible by the nature of the problem, a simple and fast procedure can be obtained that performs with sufficient results.

4.4.3 Main algorithm

Our straight line stereo algorithm takes pairs of edges in both images of the stereo pair, and computes the corresponding 3-d lines. If these lines are possible, they are stored with a certain confidence value. All combinations of lines are processed that meet certain constraints as described in 4.4.5. The resulting list of edges is merged with that of the alternative algorithm. Finally, the best match for each line is found. Each step is described below:

The camera is modelled by its mapping of world coordinates y_i to camera coordinates x_i (in homogeneous coordinates, multiplication with a matrix C)

$$\begin{bmatrix} x_1 s \\ x_2 s \\ s \end{bmatrix} = C \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix}, \tag{16}$$

where both camera matrices C_L and C_R are assumed to be known. The correspondence between lines is found in the following way: to a line in an image with dimensions x_1 and x_2 with equation

$$n_1 x_1 + n_2 x_2 + n_3 = 0 \tag{17}$$

corresponds a plane in 3-d plane with equation

$$(n_1 C_1 + n_2 C_2 + n_3 C_3) \cdot \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0 \tag{18}$$

where C_1 .. C_3 are the rows of the camera matrix C. If we have lines in both images, we can compute the intersecting line of the two planes in 3-d, and we can compute the projections of the endpoints of each 2-d line onto the 3-d line. By averaging the corresponding endpoint coordinates, we can find the 3-d line, see Fig. 4-9 and Fig. 4-10. The confidence value ξ here is

$$\xi = \left(\frac{r_1 - l_1}{l_2 - l_1}\right)^2 + \left(\frac{r_2 - l_2}{l_1 - l_2}\right)^2, \tag{19}$$

where r_1 etc. are the coordinates of the points indicated in Fig. 4-10 in any 1-d coordinate system along the intersecting line. This value is zero if the two edges correspond perfectly, and higher when they overlap less along the crossing line. Large values correspond to little overlap

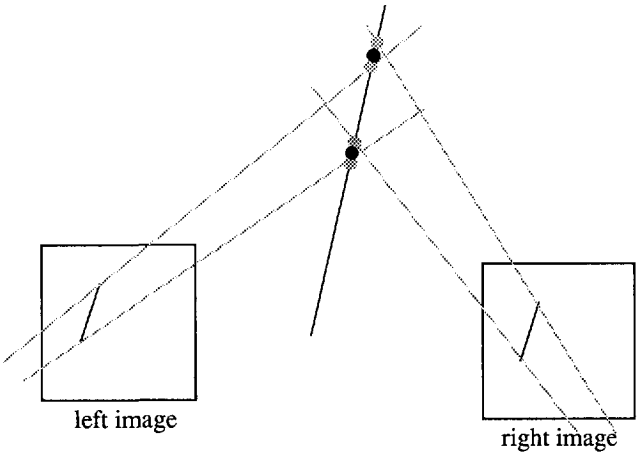


Fig. 4-9 The line-based stereo algorithm. The 3-d line is found by crossing the planes through the lines in each image. Endpoints are found by averaging projected endpoints.

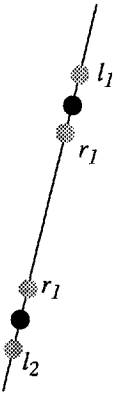


Fig. 4-10 . 3-d edge based on the projected endpoints. The points l_1 and l_2 are the projected endpoints of the left edge.The points r_1 and r_2 are the projected endpoints of the right image, where r_1 is chosen closest to l_1. The endpoints of the final 3-d edges s_1 and s_2 are in the middle of (r_1,l_1) and (r_2,l_2), respectively.

and can be rejected immediately. Therefore, edges with a confidence value above a certain threshold are rejected. This value is only symmetric when both edges, projected on the crossing line, are equally long. As this is the case for good correspondences, this is not a problem.

If the straight lines are nearly parallel to the epipolar lines of the camera system, this approach becomes inaccurate, as the two crossing planes are nearly parallel. In that case, an alternative strategy is used as described in the next subsection.

4.4.4 Algorithm for near-epipolar lines

Epipolar lines are defined[37] as the lines of points in one stereo image that can correspond to a single point in the other image. Because of the equivalence of both cameras, complete lines can be found in both images, for which every point on one line could match every point on the other, see Fig. 4-11. From this definition it is clear that epipolar line segments cannot be matched without assuming the correctness of the endpoints: without point information any point on one segment could match any point on the other segment.

Near-epipolar lines in the images are lines (corresponding to edges) that are nearly parallel to the epipolar lines in that part of the image. Without using endpoint information, there would be infinitely many possible 3-d edges corresponding to near-epipolar pairs. Therefore, the algorithm for near-epipolar lines is based on this correspondence of the end points. It is applied to lines that make an angle with epipolar lines in the image that is smaller than a certain threshold. Because of the use of endpoint information, the algorithm will only work well if both endpoints can be found accurately in both images, for instance at corners. However, it does help to solve a number of cases.

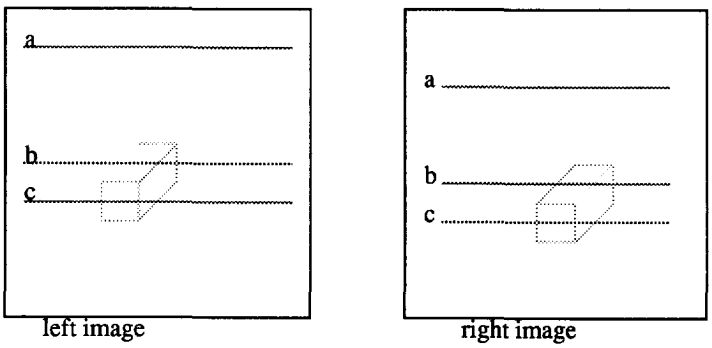


Fig. 4-11 Examples of epipolar lines. Any point on one of the lines a,b, or c in the left image could match any point on the lines a,b or c in the right image, respectively. In a general camera setup, epiolar lines are not necessarily parallel.

Given a point in an image and the camera matrix C, we can calculate the line of all points in space that can correspond to that point[8]. This can be done for all four endpoints considered, yielding two lines projecting from each image into space. The endpoints of the new 3-d edge are found at the point where corresponding lines are closest, in the middle of the lines that connect corresponding projection lines at the closest point, see Fig. 4-12. The confidence value in this case is given by

$$\xi = \frac{d_1^2}{l^2} + \frac{d_2^2}{l^2}, \tag{20}$$

where d_1 and d_2 are the distances at the closest point, and l is the length of the resulting 3-d edge. In this way, the confidence value is related to the correspondence of the end points. It is unitless, symmetric and zero for a perfect match, and similar to the confidence value for general lines. l.

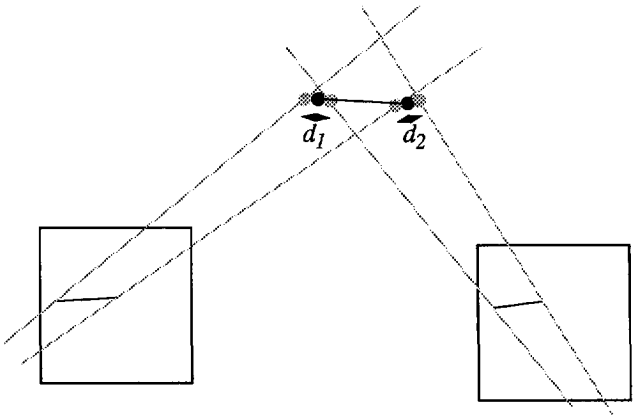


Fig. 4-12 Algorithm for the correspondence of near-epipolar

4.4.5 Constraints

A line is accepted if the projections of the two 2-d lines are sufficiently close and the endpoints of the 3-d line are both within a certain volume (region of interest). The last condition is made sufficiently simple by the fact that we are working with a robotics application: the actual volume in 3-d space that contributes to our images is of a very limited extent. This also reduces the number of possible matches.

Other constraints taken into account are:

- uniqueness. Each edge in one image may only match one edge in the other image.

- continuity. Connected edges in each image may correspond to either a corner or a discontinuity in 3-d. If a discontinuity is found that is sufficiently small, continuity is restored. This results in a well-connected 3-d wireframe.

- sign. A light-to-dark transition (assuming reasonable similarity between viewpoints) cannot match a dark-to-light transition. For this purpose, the direction of each edge is stored at the time of the straight line fit.

4.4.6 Experiments

These algorithms have been implemented in the DIAC system using TCL-image. A typical image sequence is shown in Fig. 4-13.Shown are images of two of the DIAC parts, the edges detected in those images, straight lines fitted to these edges, and the straight lines found in 3-d. The images illustrate the fact that the images are not necessarily aligned or equally bright. The side view of the 3-d scene shows that errors occur, especially in the vertical direction. Also,

not all edges visible are found. This may not be a problem, as the aim is recognition. This will be examined in (chapter 6).

It is very difficult to make quantitative statements about the performance of this kind of stereo vision system. One cannot just measure a large set of edges and conclude from them a few numbers describing the system completely. There are simply too many parameters that one can vary. For instance, the length of edges may vary, but also the difference in grey value (which is not necessarily constant over the entire edge). The orientation of the edge in space may vary (over two angles) and the presence of other edges (and reflections) also influences the problem. Finally, there is the influence of noise.

One solution to this problem would be the availability of standard scenes, with a known "ground truth". Ideally, these should be shared with other researchers in the field in order to allow for comparable results. These, however, are not generally accepted. The prototype setup does allow one to create known scenes though using the robot, up to the accuracy of the robot. Objects can be put in front of the camera in a known position and orientation. This possibility has been exploited.

A limited experiment has been performed to evaluate the accuracy of the system. In this experiment, a rectangular black block with a white top is put in front of the camera on a black background by the robot in several positions (differing by a 15 degree rotation around the object's z-axis). In each position 10 image pairs are taken and the resulting lines calculated. The setup is such that mainly lines originating from the top of the object are analysed, because the experiment is limited to these in order to have a ground truth. The lines found during this experiment are shown in Fig. 4-14 (for the x- and y-coordinates) and Fig. 4-15 (for x and z). In 12 cases, an expected line was not detected (out of 480), whereas there were 34 other detected lines. Most of the latter could be attributed to other sides of the block.

These figures show that although measurements of an individual edge reproduce fairly well over the 10 images of each position (most of the clusters of lines in both figures are less than 1 mm thick), the accuracy obtained is worse than that. Most of the clusters in these figures lie within 5 mm of each other, while there are some outliers that are much further away. The fact that the precision is much better than the accuracy suggests that the cause of these outliers is deterministic in nature. The outliers in Fig. 4-14 correspond to the angles of 105 degrees for the short side of the rectangle and 15 degrees for the long side. Since the two sides have an angle of 90 degrees, these correspond to the same direction in the image. The latter position is shown in Fig. 4-16. In this figure, the directions of the epipolar lines in the image have been indicated by white lines. This indicates that these outliers are unrelated to the near-epipolar problem.

It can be concluded that the line-based stereo vision system can find individual, well-defined edges up to a few (2-3) mm, while there are some directions where the accuracy of the system may be 6 mm. The precision σ of the system is on the order of 1 mm.

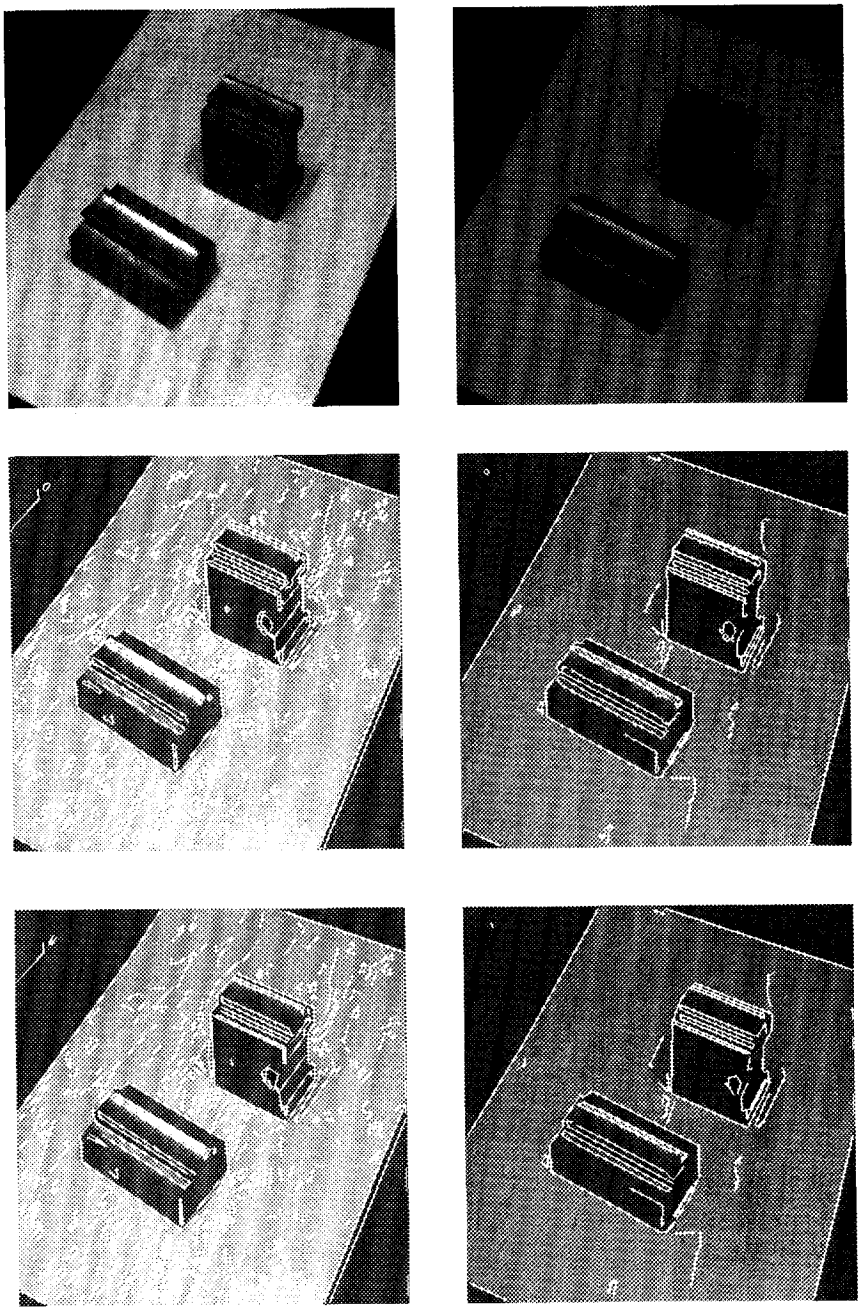


Fig. 4-13 Image sequence illustrating the line based stereo vision algorithms at work.. From top to bottom are shown: the two images, edges detected and straight lines fitted to those edges.

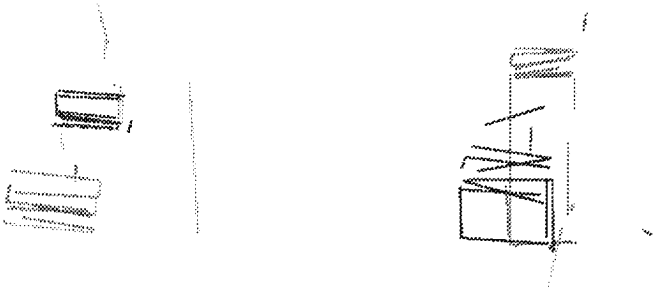


Fig. 4-13 (continued). 3-d straight lines found in the two images. On the left a top view of the scene, with darker lines indicating higher edges, and on the right a side view. The actual volume drawn is the region of interest, which accounts for different scales along different axes.

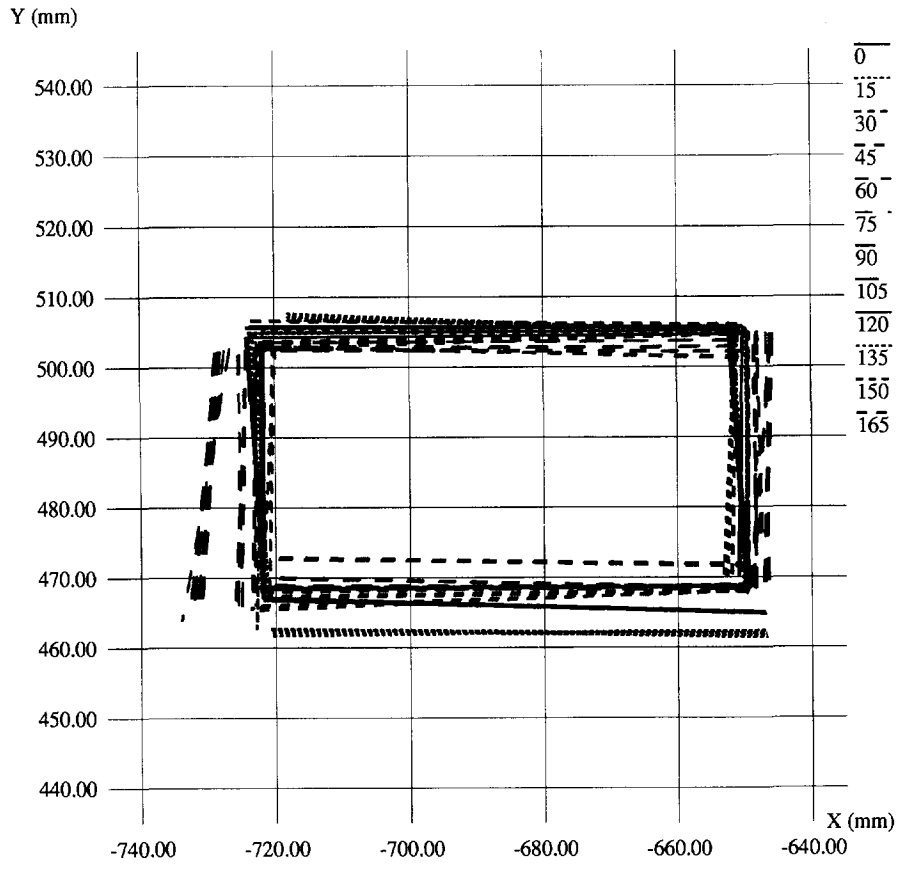


Fig. 4-14 Experiment to determine the accuracy of the straight line stereo algorithm. At different orientations (indicated by different line styles, with angles in degrees) ten image pairs were obtained and the corresponding 3-d lines calculated. The lines corresponding to the object top are shown here, rotated back to the original orientation. Coordinates are relative to the robot, in mm.

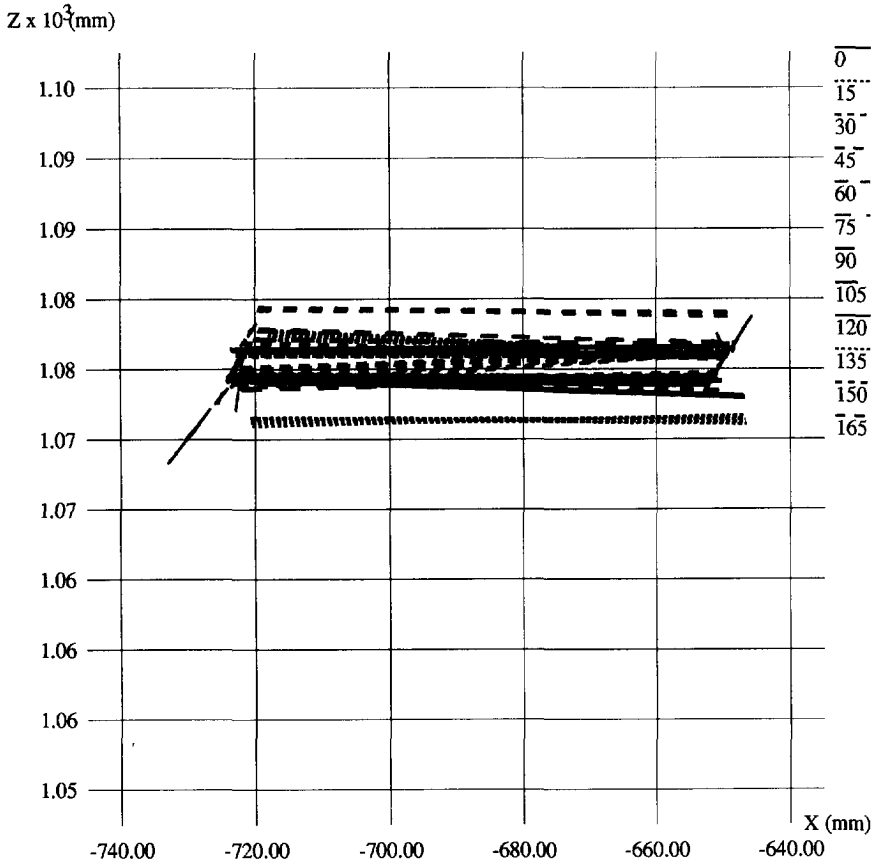


Fig. 4-15 Experiment to determine the accuracy of the straight line stereo algorithm (continued). This figure shows the x- and z-components of the lines found.

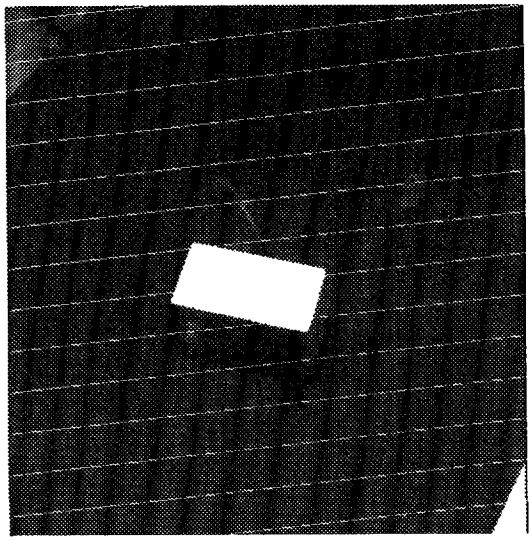


Fig. 4-16 The block used in the experiments, in the position corresponding to a rotation of 15 degrees. Overlaid are lines with directions corresponding to the epipolar lines in the camera setup used.

4.5 Ellipse-based stereo

4.5.1 Problem description

When is circle is seen under perspective projection, the result is an ellipse. Projecting lines from the focal point through each point on the ellipse back into 3-d space yields a "cone" with an elliptical section. Stereo vision results in one ellipse in each image. However, computing the section of the "cones" does not lead to a single circle. If the ellipses found were exactly correct, we would find the two different interpretations that are possible: the circle and a very eccentric ellipse (see and also [60].

This is illustrated in Fig. 4-17 and Fig. 4-21. In the first figure, the two interpretations are shown, left the circle and right the ellipse, from three orthogonal viewpoints. In extreme cases, the ellipse may become another conic section, like a hyperbola. In Fig. 4-21, corresponding points on the circle and the ellipse are indicated, with lines connecting the points and ending in the focal point of the camera (left).

In general however, the two 2-D ellipses will not be exact, and the axes of the two "cones" will not meet in one point. The section of the two cones will then look like the example in Fig. 4-19, a single curve that is not planar, but instead switches between the circle and the eccentric ellipse. Parts of the circle and the ellipse can still be distinguished. Of course, if two entirely unrelated ellipses are matched, the section of the "cones" may well be empty.

So, the problem of finding 3-d circles through stereo vision can be split into three steps:

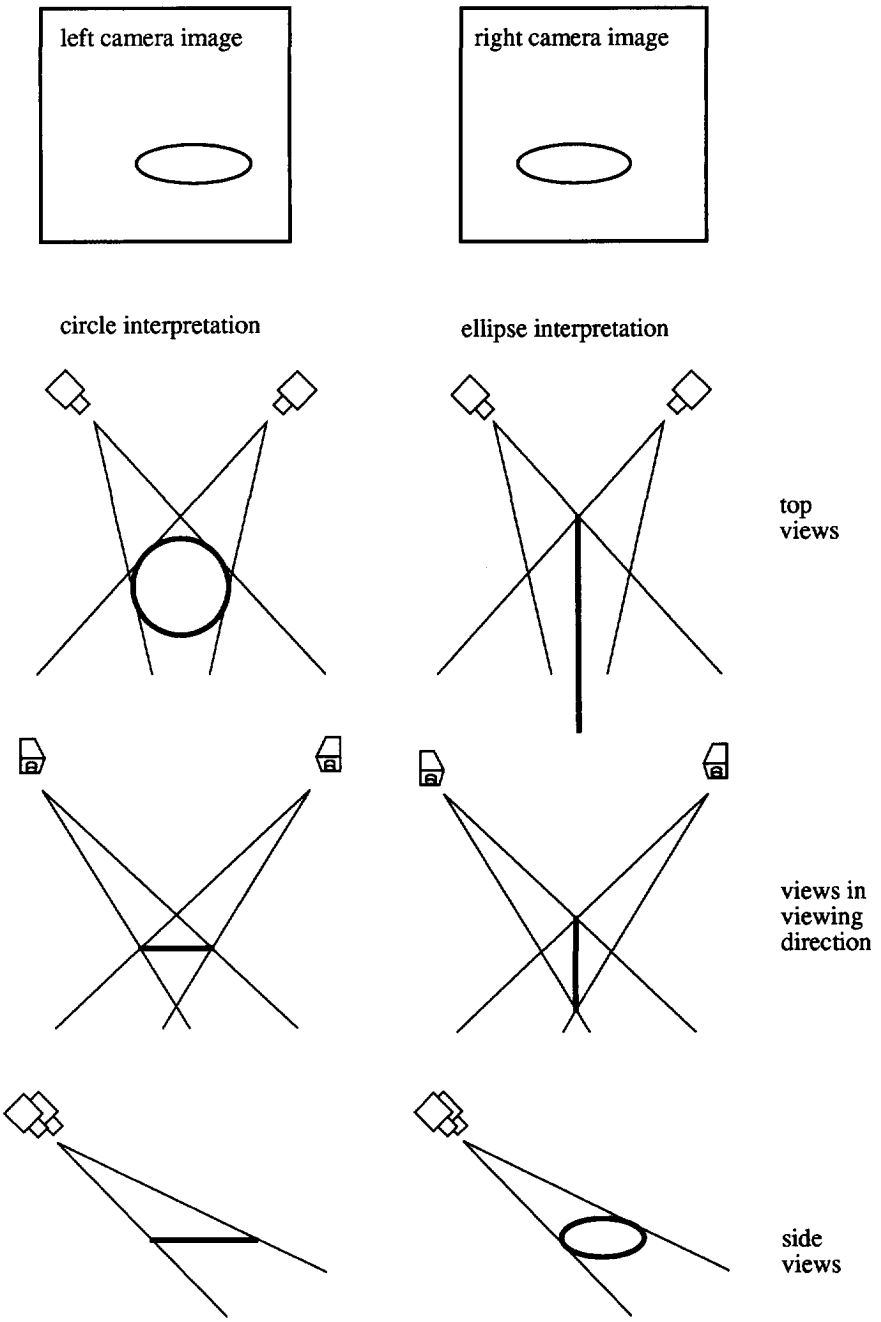


Fig. 4-17 Two images of a 3-d circle (top), and the different interpretations of a stereo ellipse pair. In the circle interpretation (left) the left circle corresponds to the right image, in the elongated ellipse interpretation (right) the left circle corresponds to the mirror image of the right circle.

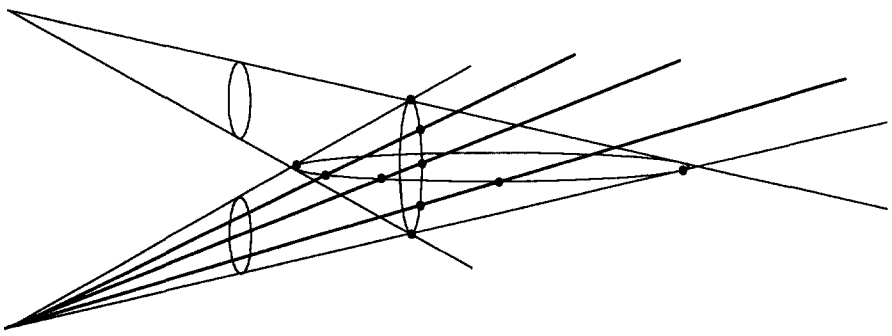


Fig. 4-18 Two interpretations of ellipses in both images. In this figure, corresponding point for the lower viewpoint have been connected by lines.

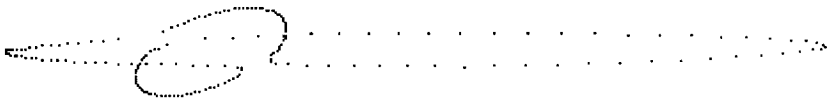


Fig. 4-19 100 points on a typical section of inexact "cones". Both the circle and the ellipse are still visible.

- Finding ellipses in 2-D images.
- Computing the section of the corresponding "cones".
- Identifying the circle in each of these sections.

In this section, we have written "cones" to denote that the set of all lines through one point and an ellipse does not, in general, have a circular section. In the remainder of this chapter, we will use the word "cone" assuming that the reader is aware of this.

We will use the following two descriptions of ellipses in 2 dimensions:

The ellipse equation is given by

$$\alpha_0 x_1^2 + \alpha_1 x_1 x_2 + \alpha_2 x_2^2 + \alpha_3 x_1 + \alpha_4 x_2 + \alpha_5 = 0 \tag{21}$$

and has been described in section 4.3.3 An ellipse can also be described by its major and minor axes A_{maj} and A_{min}, centre (x_{c1},x_{c2}) and the angle of the major axis and the x_1 axis ϕ_a, using a parameter ϕ:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_{c1} \\ x_{c2} \end{bmatrix} + \begin{bmatrix} A_{maj}\cos\phi_a & -A_{min}\sin\phi_a \\ A_{maj}\sin\phi_a & A_{min}\cos\phi_a \end{bmatrix} \cdot \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} = c + M \cdot \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix}, \tag{22}$$

with vector c and matrix M defined appropriately. Transformation of one form into the other is possible through the following equations:

$$x_{c1} = \frac{2\alpha_2\alpha_3 - \alpha_1\alpha_4}{\alpha_1^2 - 4\alpha_0\alpha_2} \tag{23}$$

$$x_{c2} = \frac{2\alpha_0\alpha_4 - \alpha_1\alpha_3}{\alpha_1^2 - 4\alpha_0\alpha_2} \tag{24}$$

$$\phi_a = \frac{1}{2}\operatorname{atan}\frac{\alpha_1}{\alpha_0 - \alpha_2} + \frac{\pi}{2}, \alpha_0 < \alpha_2,$$

$$\phi_a = 0, \alpha_0 = \alpha_2,$$

$$\phi_a = \frac{1}{2}\operatorname{atan}\frac{\alpha_1}{\alpha_0 - \alpha_2}, \alpha_0 > \alpha_2 \tag{25}$$

$$A_{maj} = \sqrt{\frac{-2K_1}{\alpha_0 + \alpha_2 - K_2}}, \tag{26}$$

$$A_{min} = \sqrt{\frac{-2K_1}{\alpha_0 + \alpha_2 + K_2}}, \tag{27}$$

where K_1 and K_2 are given by:

$$K_1 = \alpha_5 - (\alpha_0 x_{c1}^2 + \alpha_1 x_{c1} x_{c2} + \alpha_2 x_{c2}^2), \tag{28}$$

$$K_2 = \sqrt{\alpha_1^2 + (\alpha_0 - \alpha_2)^2}. \tag{29}$$

Note that these equations are defined if and only if (21) describes an ellipse, and not another conic section. The circle is included as the special case $\alpha_0 = \alpha_2$. Some of these equations are also found in [8], however the coefficients in (21) are chosen slightly differently, and some of the equations in [8] are incorrect.

4.5.2 Related research

In recent years, there have been a number of publications on the subject of ellipse finding. A number of researchers use the Hough transform to find (possibly partially occluded) ellipses, e.g. [25], [31], [38] and [70]. These methods are computationally expensive, due to the nature of the Hough transform. Ellipse detection based on moments [48] cannot deal with partially occluded objects. Our method of ellipse finding tries to estimate the parameters corresponding to a single edge string, similar to [30]. It differs from [30] in the addition of a merging step, and the use of a simpler ellipse selection algorithm. If the noise level is not too high, it performs sufficiently well at low computational cost. In [56], ellipse parameters are estimated using a bias corrected

Kalman filter. That method has the advantage of the bias correction, but no speed figures are given there.

A recent overview of the stereo vision literature is found in [27]. There is very little reference to stereo vision based on parametrized curves of higher order than a straight line. Initially, we only found [55], where it is mentioned without detail and [59] which is a predecessor to our work. The reason for this may be that it is mainly attractive when the curve considered is really a primitive of the scene. Such is the case in our work, where cylindrical objects must be recognized.

In very recent conferences, other references appeared. In [69], a partial solution is described. The approach described there is similar to our approach, which was presented at the same conference[18]. However, no effort is made to obtain the final circle equation. In [50], a closed form solution to the ellipse based stereo problem is described. This solution is overdetermined and therefore yields a matching criterion that indicates how well the circle corresponds to the actual data. It is difficult to relate this criterion to actual distances so that it is hard to say whether noise in the ellipse parameters is dealt with in a meaningful way. In [60], a closed form solution for obtaining the plane of the circle only is described, however not just for circles but for general quadric curves. No effort is made to recover the actual curve though.

4.5.3 Overview of the method

An overview of the method is shown in Fig. 4-20. Both images are treated identically by the preprocessing steps: edge detection is applied and the detected edges are stored as chaincodes. From these code strings, candidate elliptical edges are selected to which an ellipse fit algorithm is applied. The output of this algorithm is a set of ellipse equations.

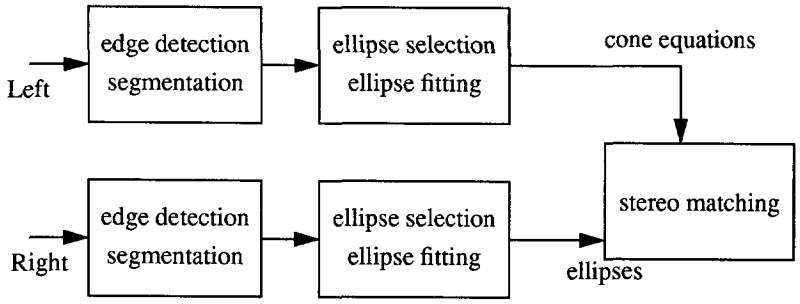


Fig. 4-20 Overview of the ellipse stereo algorithm.

One of the sets of ellipse equations is first converted to a set of cone equations using the appropriate camera matrix. This set of cones and the other set of ellipse equations with its camera matrix are then passed to the stereo algorithm which outputs a set of 3-d circles. This is done by trying to find a corresponding circle to every combination of ellipses from the two sets that meets certain constraints (as described in 4.5.6).

4.5.4 Computing the 3-d circle

Our method is general in that it does not require the two cameras to be parallel. Rather, it assumes that each camera can be described by (16). The set of all world points y that correspond to a given image point (x_1,x_2) is the line described by the two equations

$$(C_1 - x_1 C_3) \cdot y = 0 \tag{30}$$

$$(C_2 - x_2 C_3) \cdot y = 0 \tag{31}$$

where y is the column vector $(y_1,y_2,y_3,1)^T$ and C_1, C_2 and C_3 are the rows of *the camera matrix* C, see e.g. [37]. The line corresponding to (30) and (31) can also be written in parameter form, as is show for instance in [8].

The set of all points in 3-d space that correspond to a given ellipse and camera matrix can be found in the following way: to any point on the ellipse parametrized by (22) corresponds a set of two equations based on (30) and (31)

$$\left(\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} - \left(c + M \cdot \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \right) \times C_3 \right) \cdot y = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{32}$$

where x is the symbol for the cross product of two vectors. (32) may be rewritten as

$$\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \times C_3 \cdot y = M^{-1} \cdot \left(\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} - c \times C_3 \right) \cdot y = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \times y \tag{33}$$

where t_1 and t_2 are row vectors, defined appropriately. Squaring both sides (which is allowed because of symmetry) and adding the two equations yields

$$(C_3 \cdot y)^2 = (t_1 \cdot y)^2 + (t_2 \cdot y)^2, \tag{34}$$

the equation of the ellipse cone.

Let E_L be the equation belonging to the left ellipse and camera matrix. We can vary the value of the parameter ϕ in equation (22) to obtain a number of points on the right ellipse, and, by solving E_L with (30) and (31) for these 2-d points, obtain a set of 3-d points P_i on the section, as indicated in Fig. 4-21. An example of such a set is shown in Fig. 4-19. Of course, it is possible that such a set could not be found, in which case we decide that the two ellipses did not match.

Subsequently, we try to identify planes in the point set. We do that by moving a window of width n over the set of points, trying to fit a plane to P_i, P_{i+1}..P_{i+n-1}. If one or two such planes can be found, we try to fit a circle to all points in each plane. If two circles can be fit, we keep the best (this is a rare special case). If only one circle can be fit, it is presumed to be the correct circle. Otherwise, we decide that the two ellipses do not match. Both the plane fit and the circle fit are least squares fits. The circle fit returns a measure of fit: the average relative distance of each point to the circle (which is zero for a perfect fit).

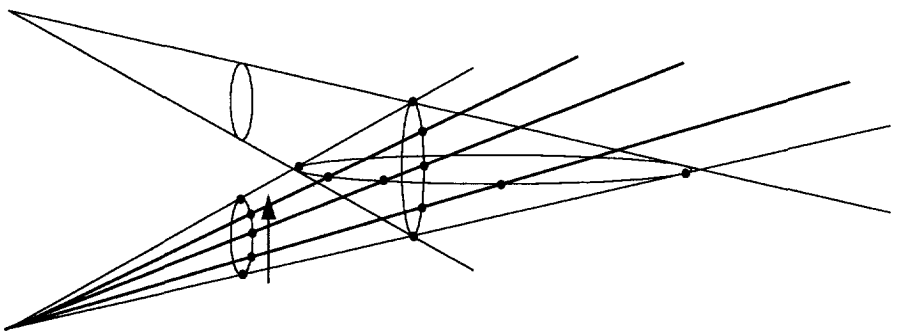


Fig. 4-21 Finding points on the section of two cones. By varying the parameter of one cone and crossing the resulting line with the other cone, a set of points is obtained in which both interpretations are present.

The whole procedure can be written in pseudo-code as shown in Table 4-1

```
CIRCLE find_3d_circle(left_image,left_camera_matrix,right_image,right_camera_matrix)
{
    circleset = {};
    for (L = all ellipses in left_image){
        E_L = cone_equation(L,left_camera_matrix);
        for (R = all ellipses in right_image){
            pointset = {};
            circle1 = NULL;
            circle2 = NULL;
            for (p = points on R)
                pointset=pointset+section(projected_line(p,right_camera_matrix),E_L);
            no_of_planes = find_planes(pointset,plane1,plane2);
            if (no_of_planes > 0){
                circle1 = fit_circle(pointset,plane1);
                if (no_of_planes > 1){
                    circle2 = fit_circle(pointset,plane2);
                    if (measure_of_fit(circle1) < measure_of_fit(circle2))
                        circleset = circleset + circle1;
                    else
                        circleset = circleset + circle2;
                }
                else
                    circleset = circleset+ circle1;
            }
        }
    }
    return(circleset);
}
```

Table 4-1 Pseudo-code of the stereo algorithm.

4.5.5 Parameters

As mentioned in the preceding section, there are a number of parameters to the ellipse stereo procedure. These are:

- the number of points N used to describe one ellipse. A number that is too small decreases the accuracy. A number that is too high leads to an unnecessarily large computation time. A typical value is 100, which provides accurate results while still being

sufficiently fast.

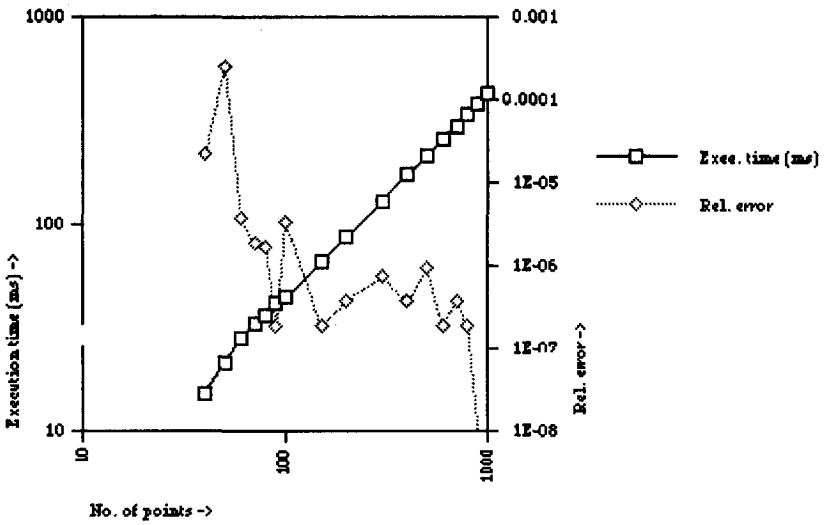


Fig. 4-22 Illustration of the importance of the parameter N of the ellpise-based stereo vision algorithm, using a test image containing one circle. The horizontal axis indicates the value of N. The left hand y-axis and the squares show the increasing execution time of the algorithm, while the right hand y-axis and the diamonds show the convergence of one of the circle parameters (the radius). Considering the fact that the algorithm has an accuracy of the order 10^{-2} and other steps in the stereo vision process have execution times of the orther 10^4 ms, the choice of N=100 is safe in both respects.

- the number of points n in the window used for plane fitting. This size is a fraction of N, enlarging this will make the algorithm more critical. A typical value is N/20.
- the thickness d_{max} allowed to a plane (distance beyond which points are no longer considered part of the plane). This is of course related to the physical dimensions of the scene. For the prototype, a size of 4 mm is used.
- an average error ε allowed for each circle found. When the radius of the circle is r, and the distance of point i to the circle centre is d_i, this error is defined as

$$\varepsilon = \frac{1}{Nr^2}\sum d_i^2. \qquad (35)$$

Smaller values will make the algorithm more critical. A typical value of 0.15 works well.

Except for plane thickness these are relative parameters and depend only on the noise.

4.5.6 Constraints

The process of stereo matching can be sped up considerably by the use of constraints. Without constraints, the matching procedure would have to be applied to every ellipse found in one image combined with every ellipse found in the other image. Our current implementation uses the following constraints explicitly:

- The epipolar constraint. This well-known constraint indicates that points in one image can only match points on a single line in the other image. This line is the section of the plane of the other image with the plane through the point and the centres of projection, see [37]. In our implementation it is applied to ellipse centres: if the centres of two ellipses are not within a given distance of an epipolar line, they are not considered in the matching process.

- The similarity constraint. Most structures in one image look almost like corresponding structures in the other image if that image is obtained from a viewpoint which is sufficiently close (which is usually the case in stereo vision). If difference in length of the major axes of two ellipses is not below a certain threshold, a match is not considered.

- Spatial constraints. Only ellipses within a specific volume in space are accepted. Because of our application, we know very well where real ellipses can be. This constraint is applied after a circle has been found, to determine whether it is a valid solution.

4.5.7 Experiments

The algorithm has been implemented and applied to a number of tests, using the following parameter values: $N = 100$, $n = 5$, $d = 4$ mm, $\varepsilon = 0.15$. First, the principle of the method is illustrated in Fig. 4-23. Shown are: the stereo pair, edges extracted from each image, ellipses found in each image and the circles found, projected back into the scene. The object shown is DIC part 29 (see Fig. 3-3), with four circles of which three are partially visible. Note that only the top side of the discs at each end are found, which is sufficient for recognition. The object is about 10 cm wide and 5 cm high, and is one of the largest parts in the cell. Both cameras were placed at 65 cm from the scene, 12 cm apart, without any special alignment.

Processing was done on a Sun 4/330 workstation using 512*512 images. For edge detection about 25 s (image processing) was required, chaincode handling and ellipse fitting took 1 s and the ellipse stereo algorithm 0.2 s (CPU times). Note that the image processing time (for the edge detection) is independent of scene complexity and can be improved upon using edge detection hardware.

In order to get more information on the accuracy of the algorithm, a set of experiments has been performed. The (experimental) analysis of a scene consisting of circles is much easier done automatically then that of a scene of straight lines (as in 4.4.6). More specifically, it is easy to derive a "relative ground truth" from the relative positions of a set of circles. Therefore, the experiment could be simpler than in the straight line case.

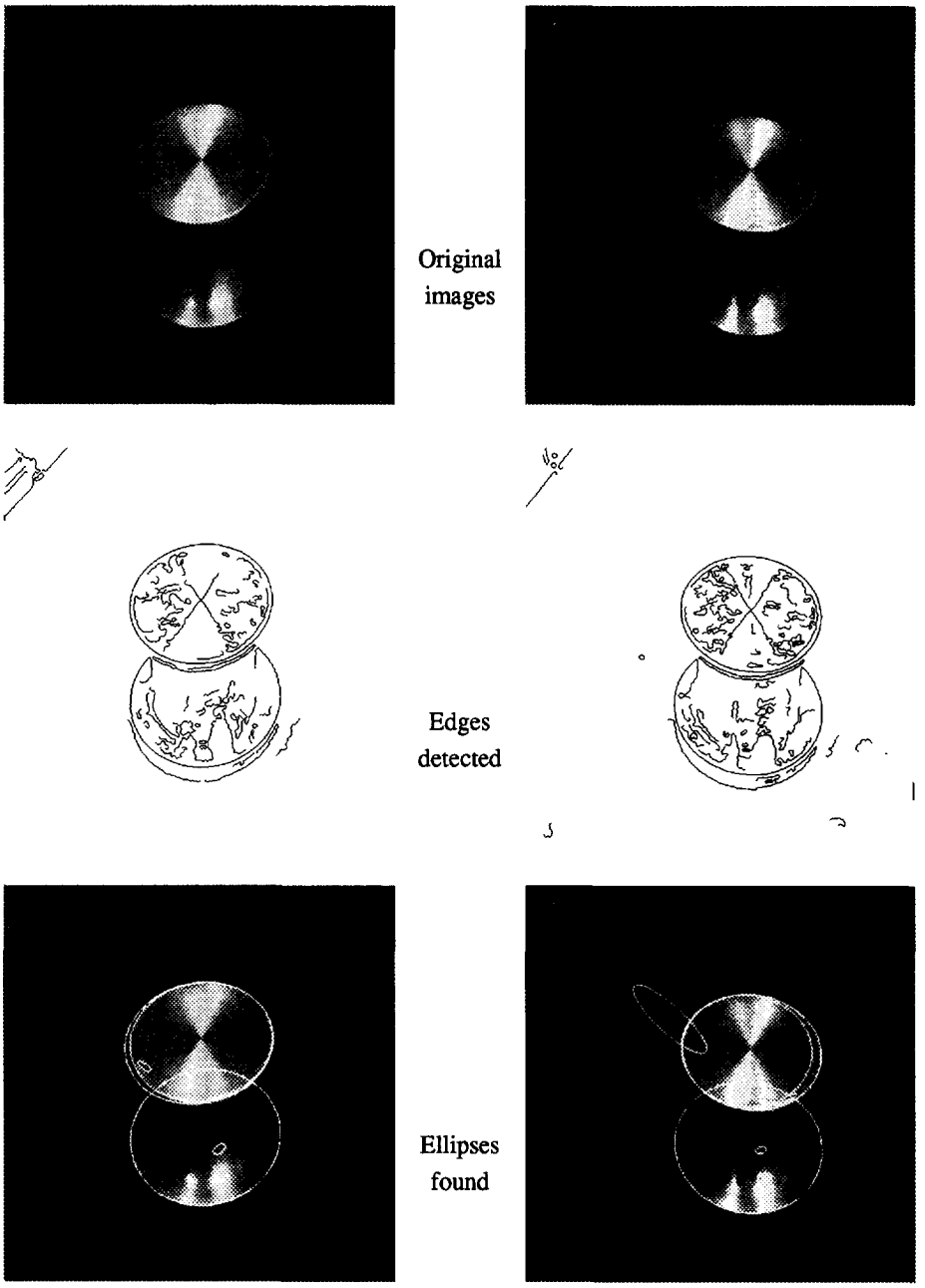
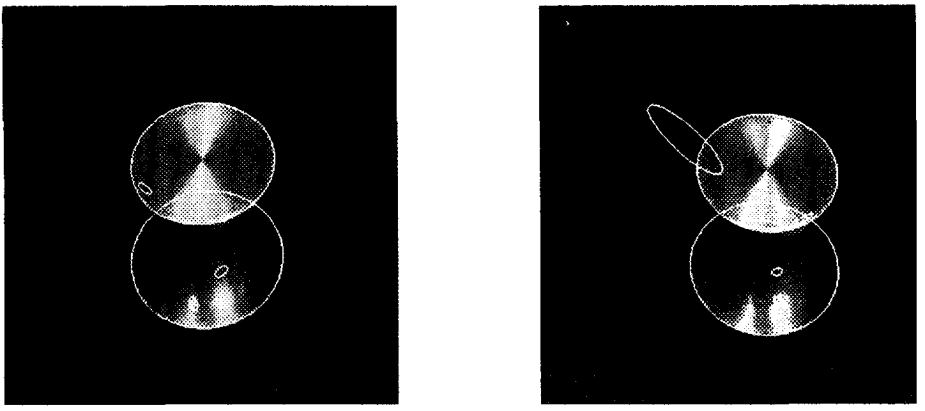
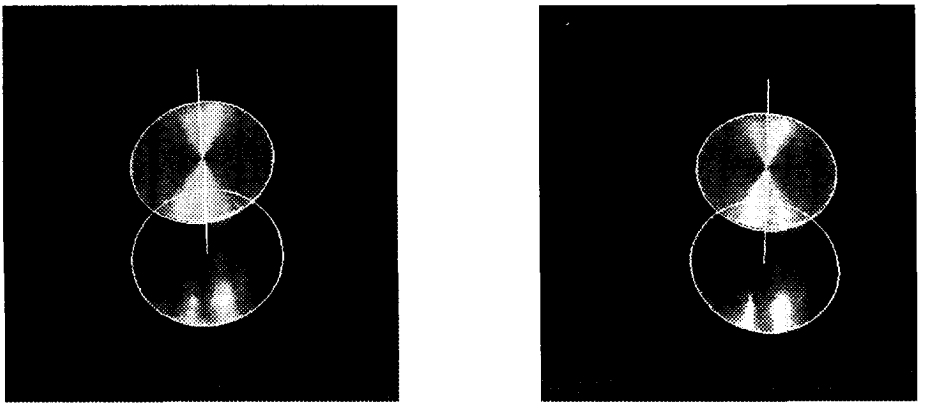


Fig. 4-23 The stereo algorithm applied to a cylindrical part. The left and right columns above show the steps as each image is processed. Note that there is no special camera alignment. Shown are the original images, the edges detected and the ellipses found. The excentric ellipse in the bottom right hand image is a false detection.



Merged ellipses



Circles found

Fig. 4-23 (continued). The last two stages of the ellipse based stereo vision. Top row: the ellipses found after merging. In both cases the two ellipses describing the top of the object are merged. Bottom row: the two circles found, drawn in the two images using the known camera matrices. For each circle, a line perpendicular to its plane and with its radius as length is also drawn.

The ellipse based stereo vision algorithm was applied to a number of drawings of circles obtained using a drawing program and a laser printer. For example, the sheet shown in Fig. 4-23 contained 15 circles of radius 2 cm, at 5 cm distance (centre to centre). For this scene, chaincode handling and ellipse fitting cost 2 s and ellipse stereo 0.7 s of CPU time. Each sheet was replaced a number of times to obtain a sufficient number of circles. First, the distances between centres of adjacent circles were calculated. We found a systematic difference of 1%, which was actually due to our laser printer. In our tables, the corrected circle sizes are mentioned. All distances and measured standard deviations are mentioned in Table 4-2. The figures show that the method allows position estimates of circle centres with a standard deviation on the order of 1% of the circle's radius.

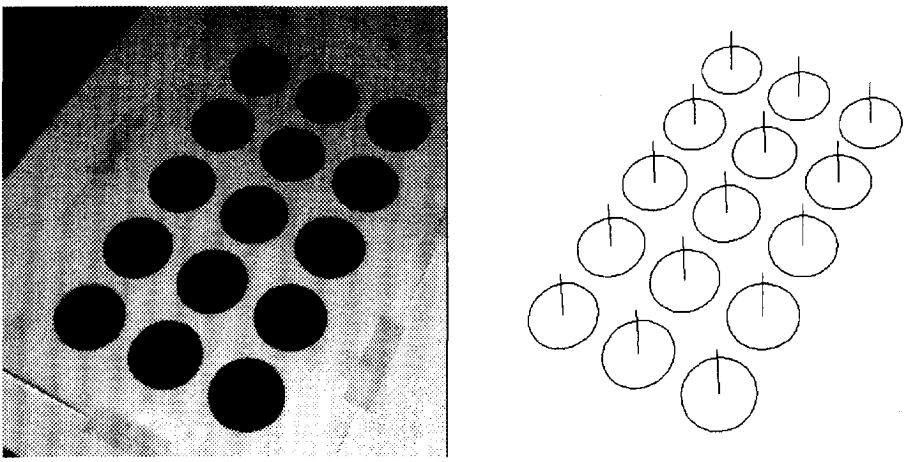


Fig. 4-23 A typical testing image, and the circles found.

Table 4-2 Measured distances between circles.

true distance (cm)	no. of distances	average measured distance	s.d. of distance	coefficient of variation
3.96	72	3.969	0.079	1.99
4.95	62	4.952	0.086	1.74
5.94	68	5.929	0.129	2.17
6.93	42	6.939	0.131	1.88
7.92	42	7.905	0.197	2.49

A summary of the radii and orientations measured (with standard deviations) is shown in

Table 4-3 It shows that no significant systematic error is made in the circle radius. The stand-

Table 4-3 Measured circle radii and angles with z-axis.

true radius (mm)	no. of circles	average radius (mm)	s.d. of radius	x-z angle (degrees)	s.d. of x-z angle	y-z angle (degrees)	s.d. of y-z angle
14.85	46	14.948	0.242	0.062	0.967	-1.325	1.589
19.80	43	19.921	0.279	0.468	1.133	-0.402	1.446
24.75	48	24.887	0.497	0.452	1.368	-0.202	1.564
29.70	36	29.986	0.335	0.177	0.499	-0.479	0.965
34.65	36	34.850	0.530	0.249	0.508	0.032	1.249

ard deviation of radii is about 1.7%. The angles measured are expected to be 0, because of the horizontal table. The orientation angles show standard deviations on the order of 1 degree. The difference in x- and y-direction was caused by the view direction, which was almost parallel to the y-axis.

Further research is needed for quantitative analysis of the performance of the algorithm where ellipses are only partially visible. A significant increase of errors is to be expected. Furthermore, the integration of ellipse based stereo in the full stereo vision system needs further evaluation.

4.6 Calibration

4.6.1 Description

Obtaining full camera matrices is necessary for the stereo vision algorithms as described in this chapter, because these allow for a general camera setup. Calibration generally implies that a number of well known points in 3-d space must be observed and their coordinates measured in image coordinates. If more than 5 points are found, the camera matrix can be estimated from the sets of points using a standard least squares procedure([8]). Usually, a larger number is taken for improved accuracy.

The camera matrices are obtained in the following way: a simple robot tool has been made, containing a single LED. Using the robot, the led is positioned in front of the cameras in a number of well-known positions (specified in robot coordinates). For each of these tool positions, the location of the led blob in the image is measured for both cameras. From the sets of robot and image coordinates, two camera matrices can be obtained. The robot positions are points on a cube, typically a 3x3x3 cube (27 points for each matrix). The procedure is described in [57].

The calibration procedure is made more complicated by the fact that there are several different coordinate systems in the system. Each introduces its own errors. An analysis of this problem for the entire cell is given in [5]. As far as vision is concerned, the following systems can be

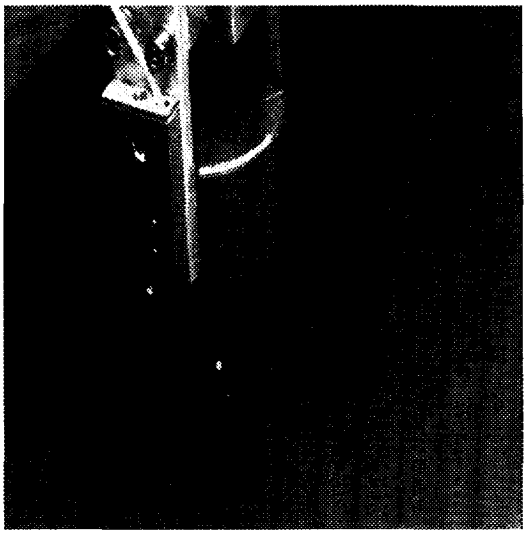


Fig. 4-24 The calibration too, held by the robot. During the actual calibration procedure, all other lights are switched off.

identified (see Fig. 4-25):

- The robot coordinate system. This is the system that the robot cell should actually use. The robot is moved to coordinates specified in this system. Robot specifications include an absolute accuracy and a reproduction accuracy (corresponding to what is generally referred to as accuracy and precision). The former, as far as it is linear over the observed volume, is taken into account by the calibration procedure. The latter necessarily influences the calibration and recognition results. For the ASEA IRB 6/2, they are specified as 0.1 mm and 0.2 mm, respectively.
- The calibration coordinate system. This differs from the robot coordinate system by a fixed translation: the vector from the robot Tool Centre Point (TCP) to the location of the LED when the calibration tool is held by the robot. This translation should result from the design of the tool. If it is not exactly known, it can be measured.
- The camera coordinate system. This is the coordinate system in which the objects are observed by the vision system. The aim of calibration is to make this system identical to the calibration coordinate system.

4.6.2 Experiments

Experiments have been performed to determine:

- the number of points necessary for calibration

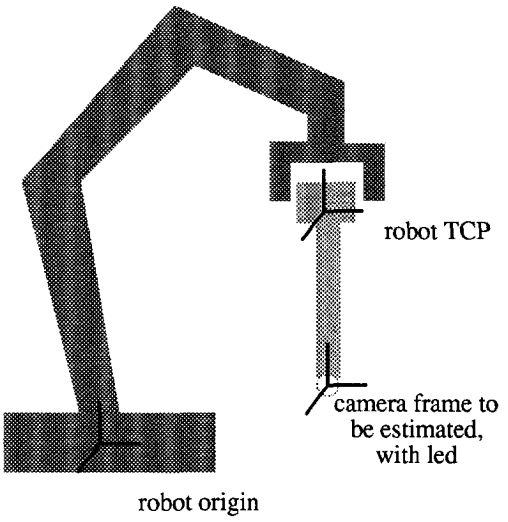


Fig. 4-25 . Different coordinate frames in the calibration setup.

- the accuracy of the calibrated system

- the influence of robot accuracy on the calibration result

The first experiment requires repetition of the calibration procedure for various numbers of points. Each time the camera matrix is calculated, the distance is calculated for all points between the measured point and the projection of the calibration tool using the camera matrix. The average distance is listed in Table 4-4

Table 4-4 Calibration result as a function of the number of points.

Number of points	Average distance left (pixels)	Average distance right (pixels)
2^3	0.61	0.61
3^3	0.42	0.49
4^3	0.43	0.43
5^3	0.44	0.43

As can be seen from this table, there is no significant improvement of the calibration if more than 3^3 points are used. Also, repeatedly measuring the same set of points (in another experiment) did not give a reduction of average distance of the standard deviation. Apparently, these

errors result from the noise in the measurements of the camera tool. It should be noted that in this case we use the same points for calibration and evaluation.

The accuracy of the calibrated system has been evaluated in the following way: after calibrating the system with a varying number of points, the calibration tool is sent to 50 points randomly chosen with a uniform distribution over the calibrated volume. Each time the stereo vision system measured the position of the tool in 3-d. The distance of the measured position of the tool to the predicted position has been measured, and the results are listed in Table 4-4 Note that in this case, different points are used for calibration and evaluation.

Table 4-5 Evaluation of calibration as a function of the number of calibration points.

Number of points	Average distance (mm)
2^3	0.56
3^3	0.47
4^3	0.38
5^3	0.44

Again, this experiment shows that extending calibration beyond 3^3 points does not gain any accuracy improvement. Given that the robot specifications suggest an accuracy of 0.1 mm and a precision of 0.2 mm, we may conclude that the contribution of the vision system to the inaccuracy is comparable to that of the robot.

4.7 Conclusions

A stereo vision system has been described using straight lines and ellipses in images, representing straight lines and circles in 3-d. First, images are acquired and edge detection is performed. This can be sped up considerably by use of edge detection hardware.

The straight line based stereo vision algorithm is straightforward, mainly caused by the use of strong constraints. A straight line fit is applied to edges and lines are checked for correspondence. The best correspondence for each is kept, if it is better than a certain threshold. Although the system is capable of finding well defined straight lines mostly with an accuracy of 2-3 mm, there are some outliers at certain orientations.

Ellipse based stereo vision is based on an algorithm to find ellipses or partial ellipses in gray value images, and an algorithm to find the circle in 3-d corresponding to two ellipses. The algorithms appears to be sufficiently fast and accurate. The performance of the stereo algorithm where ellipses are only partially visible must still be investigated quantitatively.

The calibration procedure has been evaluated. It proves not to be useful to apply the calibration procedure at more than 27 points.

5 Matching wireframes

In this chapter, a system is described for the recognition of objects using the output of the stereo vision system. It uses the geometric primitives that are present in the objects considered: straight lines and circular arcs. The system is designed to be robust against the kind of errors introduced by a feature-based stereo vision system.

The operation of the system is illustrated with examples. An evaluation of the system is impossible without evaluating the stereo vision system as well, therefore this is not included in this chapter. This will be done in chapter 6.

This chapter only describes the implementation of the recognition system, for a summary of popular recognition paradigms and some similar systems the reader is referred to chapter 2.

5.1 Introduction

The recognition system described in this thesis takes the wireframes output by the stereo vision system, and compares them with a set of model wireframes derived from models, see Fig. 5-1. Some specifications of the recognition system are imposed by the application, others by the stereo vision system. These specifications include:

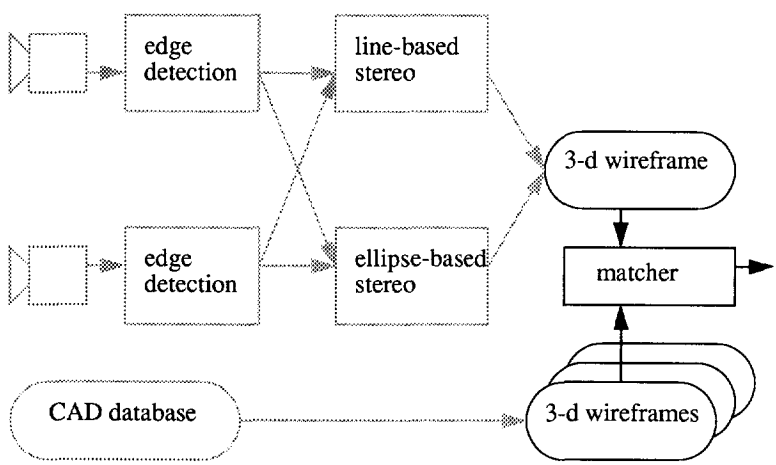


Fig. 5-1 Overview of the recognition system.

- The possibility to handle objects that consist of straight lines and/ or circular arcs.

- The possibility to handle scenes with several, or no objects.
- The use of models derived from the CAD database (see chapter 7).
- The use of the assumption that a known flat surface (a table or a pallet) is present.

The recognition algorithm has been designed to meet those specifications. It has been outlined in [20] and [17], but will be described in more detail here. Like a number of other recognition algorithms[33], it uses the well-known paradigm of hypothesis generation and testing. There are two main steps involved: for each model a set of hypotheses is found first, then for each of these hypotheses it is evaluated how well it corresponds to the observed scene. The initial hypotheses are based on a few selected features in the observed scene and some similar features in each model, whereas the evaluation is based on the entire observation and model. The final result consists of a set of hypotheses that are not conflicting and for which there is sufficient support. This set may be empty, or may contain one or more hypotheses, because it is not known in advance whether there are one or more objects in the scene. The whole process is illustrated in Fig. 5-2.

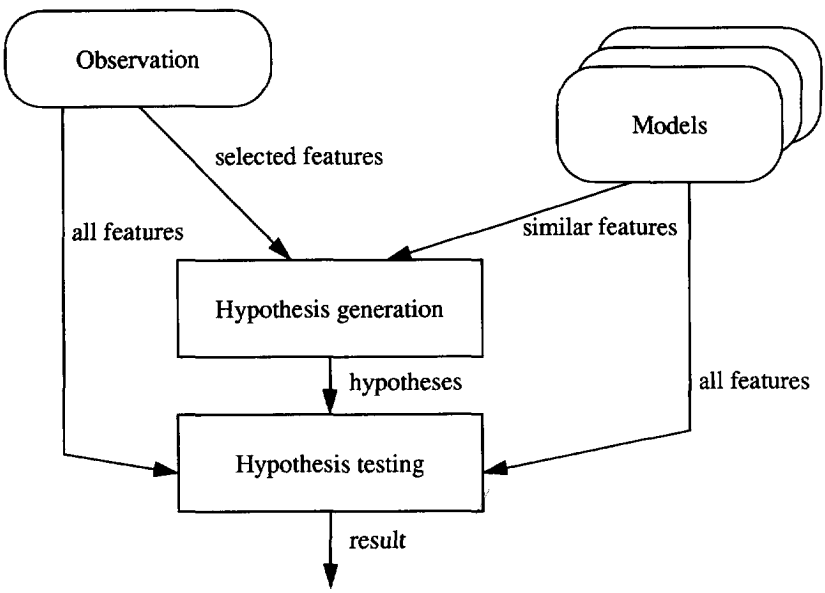


Fig. 5-2 Hypothesis generation and testing. From a set of selected features in the observation and a set of similar features in the models, a list of hypotheses is generated. These are then tested using all observed and model features.

There are two important issues in hypothesis testing. Both the definition of conflicting and a means of expressing support (and determining whether it is sufficient) must be chosen. The latter usually involves a distance or similarity measure. The choices made here are to a certain extent arbitrary, as it is very difficult to draw conclusions that generalise different applications.

This is unsatisfactory but common and accepted in this field.

The recognition system described here follows the paradigm, but adds to it a number of additional steps that are specific for this context. These are:

- Removal of hypotheses corresponding to symmetries of the models. These occur rather frequently in man-made objects, such as the parts considered in this theses.
- Application of checks for bad matches at various steps, in order to reduce the computational cost of following steps.
- Application of information about the objects (which objects are possible?) as well as about the scene (where is the table or pallet?).

The procedure is described in some more detail in Fig. 5-3. The selected observation features mentioned earlier are called the *start features*, since they are used as starting points for a database search. Whenever possibly corresponding model features are found, a *feature pair* is constructed, which references both the start feature and the corresponding model feature. A *hypothesis* is defined as a tuple of a model name and a geometrical mapping from model to observation. It can be considered as a possible identity and pose of an object in the scene. A feature pair can be used to calculate the mapping. Once a mapping has been generated, it is checked against all previous hypotheses using the same model, to see whether the two refer to a rotation symmetry of the object. If this is true, the last hypothesis is deleted. These steps together result in a list of hypotheses, and constitute the hypothesis generation stage.

Hypothesis testing is divided into several steps too. The most important step is the matching step, where all features in model and observation are found that correspond according to the mapping in a hypothesis. If the result is bad, or conflicting with a previous result, it is removed. For the remaining hypotheses (these may be zero or more since there may be zero or more objects in the scene), the mapping is improved upon using the extra information available at this stage. Finally, any remaining bad matches are removed.

To most of the separate steps, sections of this chapter are dedicated. In this chapter, this is illustrated with an example. The full evaluation of the total system is postponed to the next chapter.

5.2 Model and Observation Representation

The stereo vision system provides two different data structures: a graph representing the straight lines present in the scene, and a set of the circles that are also present. Although it detects the end points of elliptical arcs, these are difficult to find accurately. Therefore no end points are included in the final set of circles. The observation can therefore be described by a tuple

$$O = (G_o , C_o). \tag{36}$$

where G_o is the graph and C_o a set of circles, represented by their centre and a normal vector with the radius as length. The graph is given by

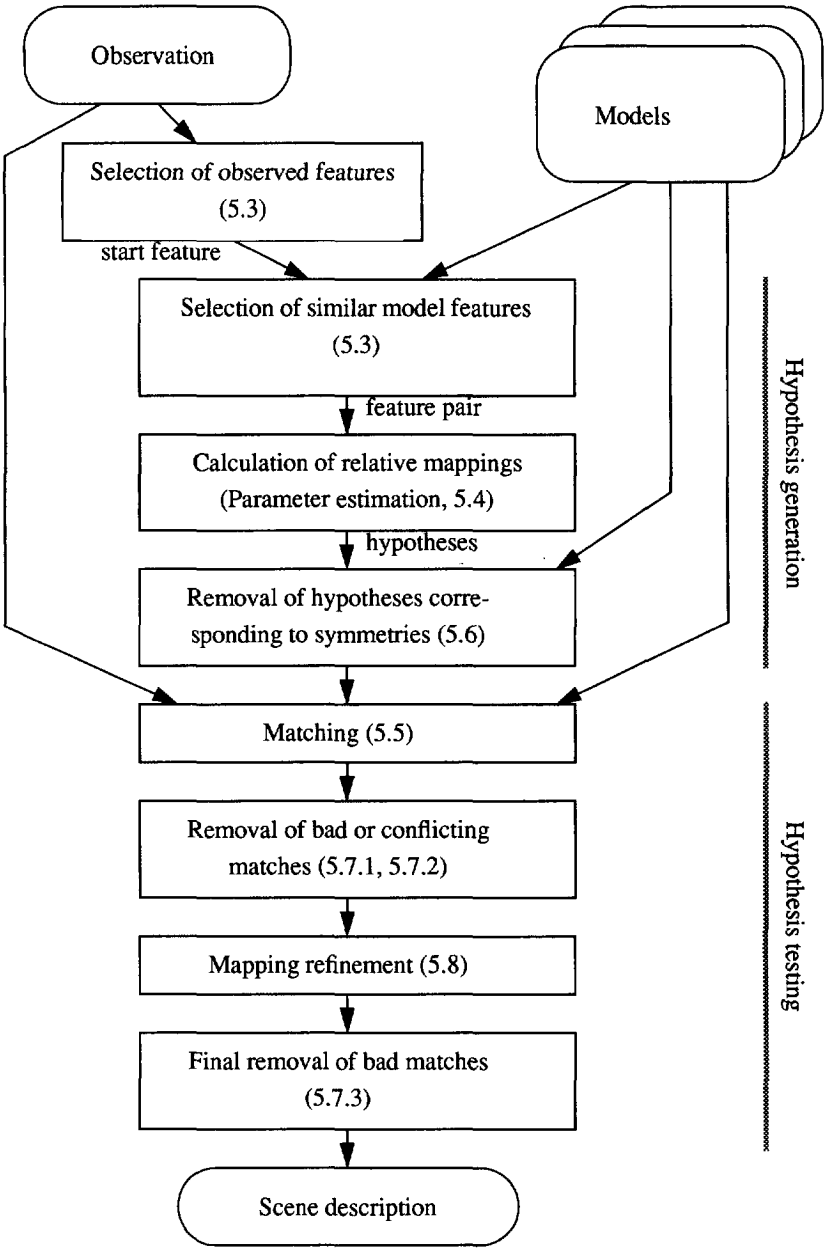


Fig. 5-3 Detailed plot of the recognition procedure, with numbers referring to the (sub-)section where each stage is described.

$$G_o = (V_o, E_o). \tag{37}$$

where V_o is a set of vertices (3-d points) and E_o is a set of edges (referring to two vertices wherever a straight line connected them). By a feature, we will generally mean either one or more edges or circles.

It is convenient to have a similar representation for the model. However, the model contains more information than the observation:

- The model contains information about all the views that can be obtained from a single object.
- A new possibility, introduced in this work, is the use of information about rotation symmetries of the object in order to reduce the computational complexity.

The first point can be realised by computing all the necessary views in advance and storing them as separate models, but this is a complex procedure (see chapter 8). Another possibility is to include all the views in one model by treating the object as if it were transparent. This can be done whenever the absence of features is not used as extra information, as is the case here. Therefore, the graph and circle set in the models contain all vertices, edges and circles present in the object.

The second point is realised by defining the model as a 3-tuple. Given a set of model graphs $\{M_i\}$, each model is defined as

$$M_i = (G_{mi}, C_{mi}, S_{mi}). \tag{38}$$

where G_{mi} and C_{mi} are a graph and a set of circles, defined as in the case of the observation. S_{mi} is a set of symmetry mappings as described in section 5.6. Implicit with each model or observation is a coordinate system, which allows us to describe positions of elements of the graph.

5.3 Finding suitable feature pairs

5.3.1 Introduction

The first two steps in hypothesis generation are finding start features and feature pairs. As the two issues are closely connected, they are both described in this section.

Feature pairs are pairs of observed and model features that allow a mapping to be estimated. As a mapping is a combination of a 3-d rotation and a 3-d translation, it has six degrees of freedom. It can therefore be estimated if three point correspondences are known. Both the start feature and the corresponding similar model feature should contain at least three different points. There are two conflicting requirements here:

- In order to reduce the computational complexity, the list of start features should typically be short, and the corresponding model features should be easy to find.
- Accuracy of the mapping estimate depends on the fact that the points are at some distance from each other, and not on one line (if they are exactly on one line, the mapping

could not be estimated).

Suitability of a feature for use as a start feature is indicated by a so-called *preference function*, that can be defined for each kind of feature. A feature with a high preference value is more suitable than a feature with a low preference value. Since a preference is usually based on two lengths, a preference function has the dimension length squared. A number of different features are discussed in the next subsections, each time defining a preference function and criteria for selecting corresponding model features.

5.3.2 Preference functions for straight lines

Two straight lines provide sufficient information for a parameter estimate. Above all, in order to prune the number of possibilities, it is required that corresponding straight lines (in observation and model) do not differ more than a factor of Δ_1 in length. However, another constraint can be imposed. In order to keep the number of features low, a requirement can be that the two observed edges have at least one end point in common. In that case the preference function for two straight lines A and B with end vectors $\underline{a}_1$, $\underline{a}_2$, $\underline{b}_1$ and $\underline{b}_2$ can be defined as:

$$s_1(A,B) = \begin{cases} \| (\underline{a}_1 - \underline{a}_2) \times (\underline{b}_1 - \underline{b}_2) \| & \text{if } (\underline{a}_1 \equiv \underline{b}_1 \vee \underline{a}_1 \equiv \underline{b}_2 \vee \underline{a}_2 \equiv \underline{b}_1 \vee \underline{a}_2 \equiv \underline{b}_2) \\ 0 & \text{otherwise} \end{cases} \quad (39)$$

The conditional part of (39) provides that the observed edges have a point in common. For simple or badly observed objects, this may not result in any features found. In this case a second preference function can be applied, to edges that specifically do not have end points in common:

$$s_2(A,B) = \begin{cases} 0 & \text{if } (\underline{a}_1 \equiv \underline{b}_1 \vee \underline{a}_1 \equiv \underline{b}_2 \vee \underline{a}_2 \equiv \underline{b}_1 \vee \underline{a}_2 \equiv \underline{b}_2) \\ \|A\|\|B\| C(\underline{a}_1) C(\underline{a}_2) C(\underline{b}_1) C(\underline{b}_2) & \text{otherwise} \end{cases} \quad (40)$$

where $C(\underline{a})$ is the number of edges that are connected to vertex a. This increases the preference for edges that are connected to others, which is motivated by the fact that they are usually more accurate (they support each other).

A disadvantage of s_2 is that the angle between the edges make is no longer taken into account, as was the case with s_1. A solution for that is to use a second constraint: the requirement that the angle between the edges (or rather, the cosine of the angle) in start feature and model feature does not differ more than a value Δ_2, where the cosine of the angle of two edges A and B is calculated as

$$\cos(A,B) = \frac{(\underline{a}_1 - \underline{a}_2) \cdot (\underline{b}_1 - \underline{b}_2)}{\|\underline{a}_1 - \underline{a}_2\| \|\underline{b}_1 - \underline{b}_2\|}. \quad (41)$$

5.3.3 Preference functions for circles

One circle almost determines a mapping up to one rotation, however for objects that are solids

of revolution (or cannot be distinguished from that by the stereo vision system), this is the best that can be obtained. Therefore, there is a preference function for a single circle, which is given by

$$s_3(C) = \|\underline{n}_C\|^2, \tag{42}$$

where $\underline{n}_C$ is the vector perpendicular to the circle, with as its length the radius of the circle. If $\underline{c}$ is the vector of the centre of the circle, the three points used are $\underline{c} + \underline{n}_C$, $\underline{c} + \underline{n}_1$ and $\underline{c} + \underline{n}_2$, where $\underline{n}_1$ and $\underline{n}_2$ are vectors with the same length as $\underline{n}_C$, but are perpendicular to it. One of the coordinates of $\underline{n}_1$ can be chosen, e.g. set to 0.

The corresponding model feature should of course be another circle, with a radius that is relatively not more than Δ_1 different from the observed circle. However, these circles almost determine the mapping, there are still two possibilities which both must be used as hypotheses. Both hypotheses contain a flag that indicates the fact that a rotation symmetry could not be resolved.

Should two circles be available that are not on the same axis (there is no line through both centres, perpendicular to both), the mapping can be determined uniquely. The preference function could then be defined as

$$s_4(C_1, C_2) = \|\underline{n}_{C_1} \times \underline{n}_{C_2}\|, \tag{43}$$

however as no such objects are present in the DIAC database, this has never been implemented. Two circles that are on the same axis but have a different radius could solve the case above, where one circle still led to two hypotheses, however, the advantage of this is so minor that it is neglected.

5.3.4 Preference functions for straight line and circle

If a straight line and a circle can be found, where the line is not on the line perpendicular through the circle and through the circle centre, the mapping can be resolved uniquely. A preference function for this case could be defined as

$$s_5(C, A) = |\underline{n}_C \cdot (\underline{a}_1 - \underline{a}_2)|, \tag{44}$$

although this disregards the rare case where an edge can be found, perpendicular to the circle but not on the axis (see Fig. 5-4). A corresponding model feature should consist of one line and one circle, with length and radius differing no more than Δ_1 from the observed length and radius, respectively.

For the DIAC objects, in the objects where this preference function could be helpful the circles are much larger than the straight lines, so the value of the preference function for circles only would be much higher. Therefore, this feature would not be used and as an elegant solution for this could not be found, this has not been implemented.

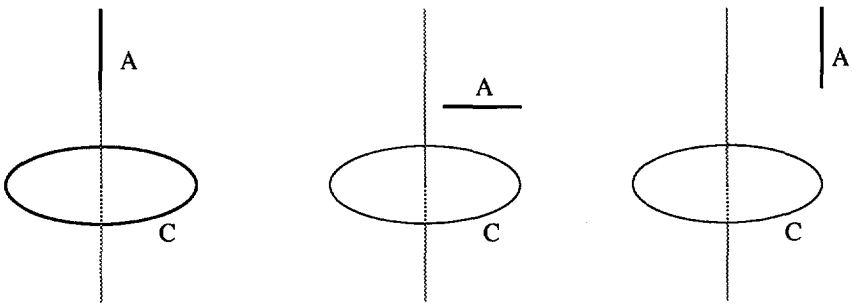


Fig. 5-4 Using a combination of a circle and a straight line as a start feature. In the leftmost case, the rotation cannot be resolved. For the other two cases, it is possible to resolve the rotation. The rightmost case is the exception not covered by the preference function in equation (44).

5.4 Parameter estimation and transformation representation

5.4.1 Introduction

Geometrical transformations in 3-d can be represented in several ways with varying compactness and ease of estimation[33]. Related to the compactness is the issue of degrees of freedom: do we allow scaling, skewing and other distortions of objects? In this thesis, we have rigid objects of fixed scale (see section 3.2). As a result of this, the maximum number of degrees of freedom to be represented is 6: three rotations and three translations.

In this section, a number of methods of describing these mappings will be described, and some estimation methods described. For this system, one of each is chosen based on the following criteria:

- Ease of estimation.

- Ease of use computationally.

- Ease of interpretation for humans.

5.4.2 Euler angles and translation

A transformation can be represented by rotations around the three major axes of the coordinate system, followed by a translation. The angles of the three rotations are generally referred to as the Euler angles. It is usually convenient to rewrite this transformation as

$$\underline{x}_2 = R\underline{x}_1 + \underline{t} \tag{45}$$

where the matrix R is of a special form (because it represents a pure rotation). In fact, if α, β and γ are the angles corresponding to rotations around the x-, y- and z-axes, which are applied in that order, R can be written as

$$R = \begin{bmatrix} \cos\beta\cos\gamma & \cos\gamma\sin\alpha\sin\beta + \cos\alpha\sin\gamma & \sin\alpha\sin\gamma - \cos\gamma\sin\alpha\sin\beta \\ -\cos\beta\sin\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & \cos\gamma\sin\alpha + \cos\alpha\sin\beta\sin\gamma \\ \sin\beta & -\cos\beta\sin\alpha & \cos\alpha\cos\beta \end{bmatrix}. \quad (46)$$

It is not trivial to estimate α, β and γ. Even estimating the sines and cosines (6 parameters with 3 constraints) rather than the actual angles is difficult. In [2], Arun et. al. describe a method for estimating the angles using a singular value decomposition. In [58], Rijnierse applies a least squares procedure to solve the general form of (45), where one element of the matrix R is set to unity and the other eight are estimated as independent variables.This implies that skewing and scaling of objects are allowed. The degree to which the matrix found differs from a strict rotation is proposed as an indication of the validity of the transformation found. This seems an insufficient argument for estimating 11 parameters (including the translation) instead of 6.

5.4.3 Quaternion and translation

This transformation, proposed by e.g. Horn[37], Faugeras and Hebert [29], allows a compact representation of rotations. *Quaternions* are defined in such a way that they can represent rotations and translations, the former by multiplication, the latter by addition. In the case of rotations, a quaternion can be interpreted as a representation of an axis around which the rotation takes place, and an angle of rotation. In the case of translations, a quaternion is added in a way similar to vector addition. The formal definition follows below.

A quaternion can be defined as a tuple $q = (\underline{w}, s)$, where $\underline{w}$ is a 3-vector and s is a scalar. Addition and multiplication are defined for quaternions:

$$(\underline{w}_1, s_1) + (\underline{w}_2, s_2) = (\underline{w}_1 + \underline{w}_2, s_1 + s_2) \quad (47)$$

$$(\underline{w}_1, s_1)\,(\underline{w}_2, s_2) = (s_1\underline{w}_2 + s_2\underline{w}_1 + \underline{w}_1 \times \underline{w}_2, s_1 s_2 - \underline{w}_1 \cdot \underline{w}_2), \quad (48)$$

where $\cdot$ and $\times$ denote the vector dot and cross products, respectively. Quaternion multiplication is associative, but not commutative[37]. A conjugate is defined as

$$(\underline{w}, s)^* = (-\underline{w}, s) \quad (49)$$

and the dot product as

$$(\underline{w}_1, s) \cdot (\underline{w}_2, s) = \underline{w}_1 \cdot \underline{w}_2 + s_1 s_2 \quad (50)$$

so that the norm of a quaternion can be defined as

$$\| (\underline{w}, s) \| = (\underline{w}, s) \cdot (\underline{w}, s). \quad (51)$$

If a vector $\underline{x}$ in 3-d space is mapped onto a quaternion using

$$x = (\underline{x}, 0),$$

it can be proven that a rotation over an angle θ around an axis through the origin and unit vector $\underline{v}$ can be written as

$$x' = qxq^* = (\underline{w}, s)\ (\underline{x}, 0)\ (-\underline{w}, s)\,, \tag{52}$$

where

$$w = \sin(\frac{\theta}{2})\, v \tag{53}$$

and

$$s = \cos(\frac{\theta}{2})\,. \tag{54}$$

This provides a powerful way to describe rotations. One of the advantages is that rotations can easily be combined, for instance, a rotation first using quaternion q_1 and then q_2 of a point described by quaternion x, can be written as

$$x'' = q_2 q_1 x q_1^* q_2^*, \tag{55}$$

furthermore, translations can be described by an addition of quaternions. Faugeras and Hebert[29] describe a vision system where the mapping from model to object space is expressed using quaternions, they also describe a method of estimating the mapping between two sets of other primitives than points (lines and quadrics). These are least squares methods, which means that if $\{x_i\}$ and $\{x_i'\}$ are sets of corresponding points, the mapping found minimises:

$$\sum_{i=1}^{n} \| q x_i q^* + t - x'_i \| \tag{56}$$

where the rotation is described by q and the translation by t. The method is described in detail in [29]. Using this method, the transformation is described by four parameters for the rotation (with the constraint that the norm of the quaternion is 1), and three parameters for the translation (the translation is used as a quaternion with s=0).

5.4.4 Matrix exponentials

Other useful representations have been published, most notably matrix exponentials, described by Ayache and Faugeras[3]. They use the fact that every orthogonal matrix R can be written as

$$R = e^H, \tag{57}$$

where matrix exponentials are defined as

$$e^H = I + \frac{H}{1!} + \frac{H^2}{2!} + \frac{H^3}{3!} + \ldots, \tag{58}$$

and the matrix H can be written as

$$H = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}. \tag{59}$$

They show that the vector $\begin{bmatrix} a & b & c \end{bmatrix}^T$ has the orientation of the axis of rotation, where its norm is equal to the rotation angle squared. By estimating the rotation represented with this vector, the rotation is estimated using only three parameters without a constraint. Although their representation seems to have computational advantages, it is hard to interpret.

5.4.5 Conclusion

For the recognition system, the quaternion notation has been chosen, for the following reasons:

- It is very easy to invert the rotation, or combine several rotations and translations. Therefore using the representation is easy.
- For humans, it is relatively easy to interpret (using axis and angle).
- Estimating the quaternion is comparable in computational cost to estimating the Euler angles[2].

5.5 Hypothesis testing

5.5.1 Introduction

Recognition of objects in the scene amounts to finding mappings from subsets of the observed data to one or more models. This is done by finding similar (though not necessarily identical) edges and arcs in each set. The matching process should be able to deal with incomplete data, due to occlusions and noise. It also deals with the fact that only a certain view of the object is observed. In the algorithm described here, no attempt is made to distinguish different views. Instead, we are satisfied if the object is recognized from a significant observed part.

Let M_i, be the model, which contains a set of features of an object i from our database. Let O be the set of observed features, consisting of edges and arcs obtained using vision. Furthermore, let $\{ F_j \}$ be a set of potential mappings from the model's coordinate system to the observation's coordinate system, in other words F_j is an hypothesis about the presence of object i at a certain position and a certain orientation. Now consider each of the feature sets

$$H_j = (O \cap F_j(M_i)), \tag{60}$$

where $\cap$ is an inexact section operator yielding a feature set H_j containing corresponding edges and arcs in two feature sets O and F_j. Then H_j represents the data found supporting the presence of object i at the location indicated by F_j. We assume that no observed edge or arc can be caused by two objects. This means that two hypotheses are conflicting if they are based on the same evidence, in other words if

$$// H_j \cap H_k // \gg 0, j \neq k, \tag{61}$$

where $//G//$ is an operator taking the size of a feature set G, and $\gg$ is used to denote a signifi-

cant difference. If two hypotheses are conflicting, we prefer the one with the most supporting data. To interpret a scene (which may contain several objects), we are looking for the sets of mappings $\{F_j \mid j = 1..N\}$ and associated hypotheses $\{H_j \mid j = 1..N\}$ which have a sufficient amount of supporting data and are not conflicting. This means that

$$// H_j // \gg 0, \tag{62}$$

and

$$// H_j \cap H_k // \cong 0, j, k = 1..N, j \neq k. \tag{63}$$

After having found H_j, in other words the match, we perform another estimate of the rotation and translation parameters. This time we use all matched elements of O, to get as accurate an estimate as possible. If several H_j remain, we have identified several objects in the scene. As this system does not make the assumption that only one object is present, all of these are part of the output result.

In the matching process, the same edges are used both in the straight line stereo vision algorithm and in the circle finding algorithm. Therefore, if possible a detected primitive is added to both sets. Therefore, in the matching process, circles are only matched with circles and straight lines are only matched with straight lines. The matching process then amounts to the following steps:

- Determine whether two features match.
- Calculate a quantity indicating how well they match. This will be proportional to the length of the corresponding parts of the edges.
- If this stage can be followed by a refinement step, calculate the corresponding points which may improve upon the parameter estimate.

For both straight lines and circles, a separate subsection explains each step below.

5.5.2 Matching straight lines

The section operator ($\cap$) for model and object feature sets mentioned above is implemented for straight lines in the following way: for each of the edges in O, calculate its mapping in model space. Then, project each mapped observed edge onto each edge in the model (see Fig. 5-5). If the distance of the edge to its projection, and the displacement of the projected edge along the model edge are not too large, we accept this as a possible match. For each observed edge, the best match to the model holds and the projection is made part of the section. The operator is implemented this way, because corner points and end points of edges can not be measured very accurately. Also, loss of information due to occlusions, lighting conditions or noise can be overcome in this way.

The projection can be determined as follows (see Fig. 5-6):

Let ab be the model edge, determined by its end vectors $\underline{a}$ and $\underline{b}$. Let $\underline{v}$ be the direction vector of unit length of the line through ab:

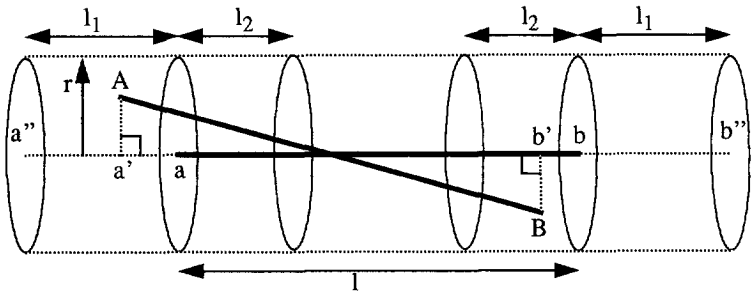


Fig. 5-5 If the observed edge AB lies completely within the cylinder with axis a"b" and radius r , and its endpoints are within the subcylinders delimited by l_1 and l_2 around the endpoints a and b, it is considered to match with model edge ab. The matching length is determined by the projection a'b' of AB onto the axis a"b". The radius r and the relative cylinder lengths l_1 / l and l_2 / l are parameters of the matching routine.

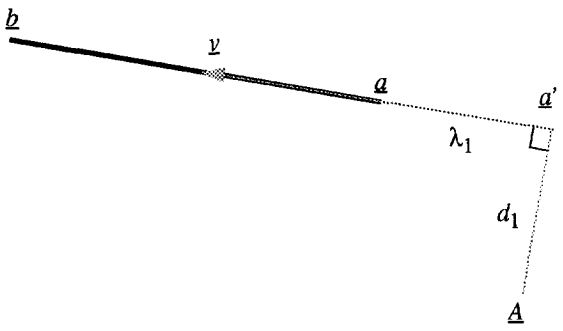


Fig. 5-6 Calculating the projection of a point in the transformed observation onto a model edge. Obtained are: the displacement along the edge λ_1, the distance d_1 of the point to the model edge, and the projection a'.

$$\underline{v} = \frac{\underline{b} - \underline{a}}{\|\underline{b} - \underline{a}\|} \tag{64}$$

Let $\underline{A}$ be the first end vector of the transformed observed edge. Without loss of generality, we can assume that $\underline{A}$ corresponds to $\underline{a}$, and the other observed vector $\underline{B}$ corresponds to $\underline{b}$. If this turns out not to be the case, $\underline{A}$ and $\underline{B}$ can be exchanged at the end of the calculation. Then, the displacement λ_1 of the projection of $\underline{o}_1$ onto the line through *ab* is given by

$$\lambda_1 = (\underline{A} - \underline{a}) \cdot \underline{v}, \tag{65}$$

whereas the projection itself, $\underline{a}'$ is given by

$$\underline{a}' = \underline{a} + \lambda_1 \underline{v}. \tag{66}$$

The distance d_1 of the point a' to the original endpoint of the transformed observation then is given by

$$d = \|\underline{A} - \underline{a}'\|. \tag{67}$$

Similarly, the values λ_2 and d_2 and the projection $\underline{b}_2$ for the second observation vector $\underline{B}$ can be found. There are three constraints that determine whether the two edges are considered matching: the endpoints of the transformed, observed cylinder must be inside the cylinder:

$$d_1 < r \wedge d_2 < r, \tag{68}$$

and they must lie within subcylinders around the end points of the model vector:

$$-\frac{l_1}{l} < \lambda_1 < \frac{l_2}{l}, \tag{69}$$

and

$$1 - \frac{l_2}{l} < \lambda_2 < 1 + \frac{l_1}{l}. \tag{70}$$

If the two edges match (that is, all criteria (68)...(70) are met), the amount of correspondence between them is indicated by the length of the overlapping part:

$$\tau = (\lambda_2 - \lambda_1) \cdot \|\underline{b} - \underline{a}\| \tag{71}$$

For further refinement, the correspondence between the points A and a', and B and b' can be stored. The parameters used in this straight line matching procedure are r, l_1/l and l_2/l.

5.5.3 Matching circles

Circles match when, after transformation, one circle refers approximately to the same set of points in space as the other circle. As in the case of straight lines, it must be realised which part of the information is most reliable. The stereo vision detects the fact that only a part of a circle is present (and indicates which part), but once again these end points prove to be very noise sensitive. Also, the part of the circle that is detected may differ between the left and right image. Therefore, the matching program uses only a part of the information: the location, orientation and radius of the whole detected circle are used, even if only part is detected.

Analogous to the definition for straight lines, the matching parts of two circles can be defined by calculating a torus of thickness r around the one circle, and looking at the section of that torus and the other circle, see Fig. 5-7. There are two differences though. First of all not the whole circle is required to be inside the torus. This allows for the case where only a part of a circle is visible, and the remaining part is estimated inaccurately.

Second, rather than analytically computing the section, which would be complicated even if possible, the section can be approximated by approximating the second circle with n_c straight

lines, and computing which of these are inside the torus. For large values of n_c, this converges to the correct value (as the set of straight lines converges towards the curve of the circle), while for small values this approximation gets worse, but its computation less demanding (the algorithm complexity is of the order n_c).

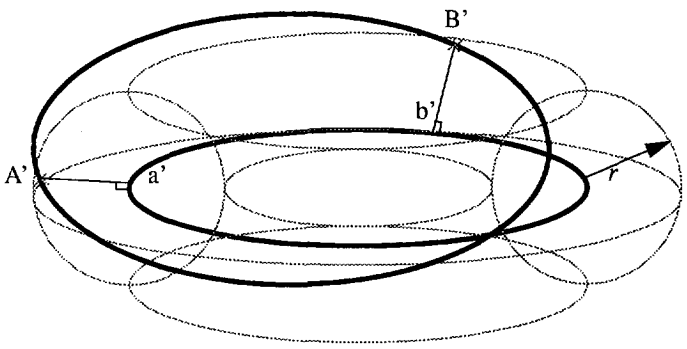


Fig. 5-7 The matching part of two circles can be defined using a torus of thickness r around one. The part of the other circle inside the torus (here between the points A' and B') and the corresponding part of the original one (here, a' b') are the matching parts.

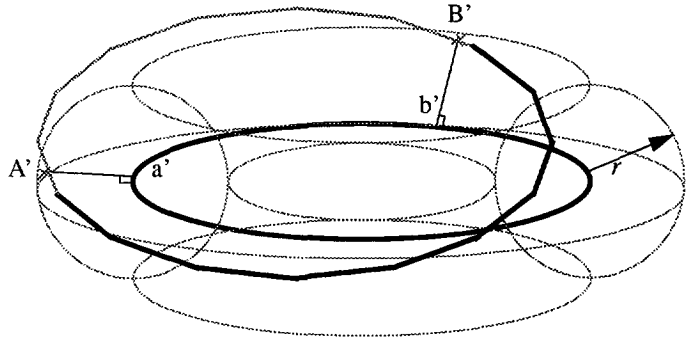


Fig. 5-8 If the second torus is approximated by straight lines, the section becomes computationally easy.

A straight line is considered to be in the section if both its end points are in the section. In order to find out whether a point p is inside the torus with radius r around the circle with centre vector $\underline{c}$ and normal vector $\underline{n}$ (where the length of this vector is the radius of the circle), its perpendicular distance to the circle must be calculated, see Fig. 5-9. In this figure, the signed distance d_1 of the point to the plane through the circle is

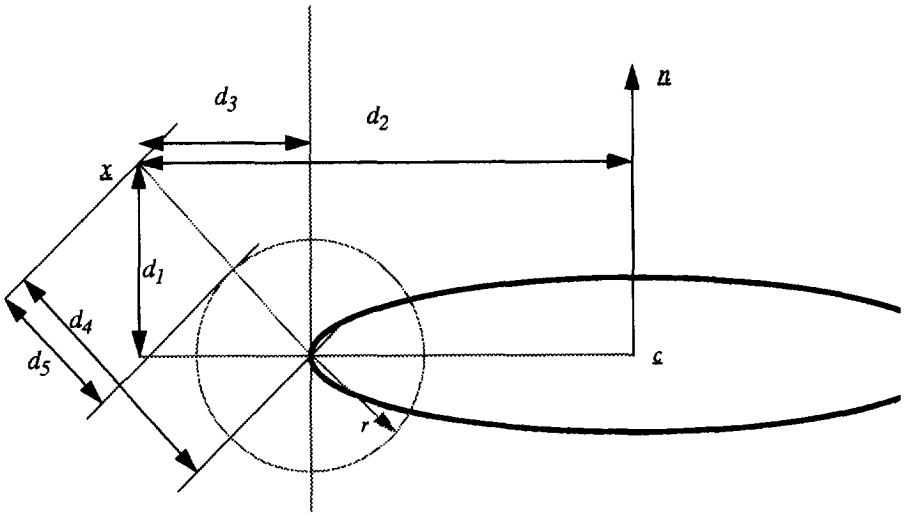


Fig. 5-9 Distance d_5 of the point $\underline{x}$ to the torus around $\underline{c}$ with radius and orientation indicated by $\underline{n}$ and thickness r, with intermediate steps d_1 (signed distance to the plane), d_2 (distance to the axis of the circle), d_3 (signed distance to a cylinder through the circle), d_4 (distance to the circle).

$$d_1 = (\underline{x} \cdot \underline{n}) - (\underline{x} \cdot \underline{c}), \tag{72}$$

where the sign indicates whether the point is above or below the plane. The distance of the point to the circle's axis, d_2, is given by

$$d_2 = \left\| \underline{x} - \underline{c} - \frac{d_1}{\|\underline{n}\|} \underline{n} \right\|, \tag{73}$$

and the signed distance d_3 of the point to a cylinder through the circle then is

$$d_3 = d_2 - \|\underline{n}\|, \tag{74}$$

where a negative sign indicates that the point is inside the cylinder. The distance of the point to the circle itself, d_4 follows from

$$d_4 = \sqrt{d_1^2 + d_3^2} \tag{75}$$

so that the signed distance d_5 of the point to the torus is

$$d_5 = d_4 - r, \tag{76}$$

where a negative sign indicates that the point is inside the torus. So a line segment is inside the torus when for both endpoints, $d_5 < 0$.

Two circles are considered to match if their radii are within a certain ratio, and any line segment of the approximation of the observed circle is within the torus around the model circle. If r_m and r_o are the radii of model and observed circle, respectively, this requirement can be written as:

$$\frac{1}{\zeta} < \frac{r_m}{r_o} < \zeta, \tag{77}$$

where ζ is a parameter of the system.

In that case, the amount of correspondence between them is given by the sum of the lengths of the matching segments:

$$\tau_c = \sum_{\substack{\text{segments} \\ \text{inside torus}}} \| \text{segment} \| . \tag{78}$$

This amount of correspondence is of dimension length (measured in robot units, where 1 unit in the prototype is $^1/_{16}$ mm), hence it is comparable to that given in (71). For later refinement, the end points of each matching segment and their projections onto the circle are stored. The parameters used in the circle matching algorithm are the radius of the torus r and the limit to the ratio of the radii, ζ.

5.6 Removal of symmetries

5.6.1 Use of symmetries

The number of hypotheses to be verified can be reduced by exploiting knowledge about known symmetries in the object. A mapping S, such that

$$S (M_i) = M_i, \tag{79}$$

if S is not the identity mapping, indicates that the object i is symmetrical. Of course, mirrorwise symmetry is not important here, since only rotations and translations can occur.

If we have the set of symmetries $\{ S_l \}$ and two hypotheses with associated mappings F_j and F_k respectively, and we find that for some $S \in \{ S_l \}$

$$F_k \circ S \equiv F_j, \tag{80}$$

then, after verifying F_j there is no need to verify F_k, and F_k can be disregarded. This comparison is performed immediately after the parameters associated with a hypothesis have been estimated. Two mappings with normalised axes directions $\underline{a}_1$ and $\underline{a}_2$, angles α_1 and α_2 and translations $\underline{t}_1$ and $\underline{t}_2$ are considered equal if translations, axes and angles each do not differ too much:

$$\begin{aligned}&(\|t_1 - t_2\| < \sigma_1 \wedge |\alpha_1 - \alpha_2| < \sigma_2 \wedge \|a_1 - a_2\| \sin\frac{\alpha_1 - \alpha_2}{2} < \sigma_3) \vee \\ &(\|t_1 - t_2\| < \sigma_1 \wedge \left||\alpha_1 - \alpha_2| - \frac{\pi}{2}\right| < \sigma_2 \wedge \|a_1 + a_2\| \sin\frac{\alpha_1 - \alpha_2}{2} < \sigma_3)\end{aligned} \quad , \tag{81}$$

where the second line corresponds to the case that the second axis is oriented inversely. σ_1, σ_2 and σ_3 are thresholds on the differences in translation, angle and axis, respectively, and thus parameters of the matching procedure. Note that the difference between the axes is weighed with the difference between the angles, if the rotations are over small angles the estimate of the axis orientations becomes unreliable.

5.6.2 Finding symmetries

The set of symmetries $\{ S_l \}$ of a model can be computed by applying the matching procedure to the model and itself. However, a different set of parameters is used in order to find only real symmetries and not approximate ones. Most notably, it is required that nearly all features in both feature sets match (typically the threshold is put at 99.5 % to allow for some computational errors), and the parameter Δ_1, indicating the maximum distance between two matching edges, is set to a very small value (a fraction of a mm).

Cylinders themselves have an infinite number of symmetries. This number is reduced artificially by the fact that the CAD system used (Medusa) inserts false edges in order to produce simple wireframes. By maintaining these in our models, the number of symmetries is reduced, see Fig. 5-10.

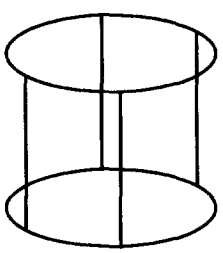


Fig. 5-10 Introduction of false edges by the CAD system. Only the two ellipses are real edges, but they would not constitute a simple wireframe. For that reason four straight edges are introduced. These can be used to reduce the number of symmetries from infinite to four.

5.7 Removal of bad matches

5.7.1 Overlapping or bad matches

Removal of bad and conflicting matches can be done based on the information acquired at the time the amount of supporting edge length for a hypothesis was computed. If the amount of supporting data is below a certain fraction δ_1 of the total edge length of the model, a hypothesis is immediately rejected:

$$\| O \cap F_i(M) \| > \delta_1 \| M \| \Rightarrow \text{reject}. \tag{82}$$

After all hypotheses have been evaluated, possible conflicts are considered, as described by equation (61). The amount of data supporting two hypotheses simultaneously should actually

be computed by matching the two results, which is an operation as complicated as evaluating a hypothesis. This is made more unattractive by the fact that the number of times this calculation has to be performed is in the order of the square of the number of hypotheses.

The section can be approximated however by simply adding the lengths of all the model features for which a correspondence was found in both hypotheses. Although this is an overestimate, it turns out to work satisfactory and the computational cost is much lower. Because it is an overestimate, it could lead to more rejections, but no false detections are introduced. This overlapping edge length is computed, and if it exceeds a threshold, one of the hypotheses involved will be deleted from the list. If the threshold is δ_2 times the total length of the observed edges, this deletion happens when

$$\|H_i \cap H_j\| > \delta_2 \|O\| \Rightarrow \text{reject}. \tag{83}$$

Which hypothesis is deleted, depends on two factors: if one hypothesis is supported much better than the other, the weaker one is deleted. However if the amounts of overlap are similar, other information is used as described in the next subsection.

$$\begin{aligned} \frac{\|H_i\|}{\|M_i\|} - \frac{\|H_j\|}{\|M_j\|} > \delta_3 &\quad \Rightarrow \text{reject } j \\ \frac{\|H_i\|}{\|M_i\|} - \frac{\|H_j\|}{\|M_j\|} < -\delta_3 &\quad \Rightarrow \text{reject } i \\ -\delta_3 < \frac{\|H_i\|}{\|M_i\|} - \frac{\|H_j\|}{\|M_j\|} < \delta_3 &\quad \Rightarrow \text{use additional information} \end{aligned} \tag{84}$$

5.7.2 Use of world information

Use of additional information of the recognition environment can make a significant improvement on the matching performance. There are two such facts used in the recognition system: the rigidity of both the objects and the table. Both will be described in this section.

The first fact indicates that no two objects can occupy the same position in space, and that therefore two hypotheses that put objects at overlapping positions are conflicting. This is a stronger requirement than the fact that no observed edge can belong to two objects as used above. Evaluating it however requires matching to each other, all hypotheses that were initially accepted. Therefore, this requirement is more computationally expensive to verify then the previous one.

Based on these considerations, the recognition algorithm uses this constraint only at its very end, when all remaining hypotheses are matched against each other. If the relative overlap between two hypotheses is more than δ_2, only the one with the most support is maintained in a manner similar to the one described in 5.7.1.

The second additional piece of information about the environment is the rigidity of the table. This makes it possible to rule out any hypotheses that suggest points below the table surface. Furthermore, a hypothesis where some object points are at the table surface is considered

more plausible than a hypothesis with the object hanging in the air. The system makes use of this information as follows:

During the evaluation of a hypothesis, each model point is transformed to the observation space, where its relative position to the table surface is calculated. This results in a height h for that point. For the whole object, the lowest value of h, h_{min} is calculated. If h_{min} is negative, then the hypothesis is impossible. Taking a small margin for measurement errors, this requirement becomes

$$-\mu_{below} r < h_{min}, \tag{85}$$

where r and μ_{below} are parameters of the system. If the lowest point is too high above the table, the hypothesis is rejected too, because it is very unlikely that the object will be on top of another large object (we allow for small parts below the objects). This requirement can be written as

$$h_{min} < \mu_{above} r, \tag{86}$$

where μ_{above} is another parameter.

The information about the table position is also used to calculate another score for the hypothesis, which can be used if the scores of two conflicting hypotheses are too close for a decision. This score is defined as:

$$\tau_{table} = \sum_{\text{allpoints}} \begin{cases} \alpha h^2 \text{ if } (h \le 0) \\ \beta h^2 \text{ if } (h > 0) \end{cases}, \tag{87}$$

where α and β are parameters of the system, used to account for the difference in the situation when the object is above the table (less likely) or below the table (impossible). As implemented, this score is only compared to other table scores, so that only the ratio of the two parameters matters.

5.7.3 Final removal of bad matches

After the initial removal of bad hypotheses, the pose estimations for the remaining hypotheses are refined as described in the next section. This is done iteratively, after each iteration bad and conflicting hypotheses can again be rejected. After this refinement, the final evaluation of conflicts using the rigidity constraint takes place. The hypotheses that remain then, are output as accepted hypotheses.

5.8 Pose refinement

The initial hypotheses were based on four points, as shown in 5.3. During the matching process, more information has become available from the matching features that were found for a number of model features. For straight lines, this information consists of two points per straight line found (see 5.5.2), whereas for circles the number of points found depends on the approximation of the circle by straight lines (see 5.5.3). All these point pairs are collected, and

using these the mapping is reestimated a few times, each time followed again by the removal of bad or overlapping cases.

Although it cannot be proven that this procedure always converges towards a valid solution, during the experiments described in this thesis no case has been observed where the procedure did not converge. Occasionally, a wrong hypothesis, after the first iteration, will not yield the four points necessary to continue, but otherwise the parameter estimate seems to converge. After three iterations, the parameter estimate changes between two subsequent iterations usually becomes smaller than the accuracy, so the number of iterations is fixed at three (this behaviour was observed consistently throughout the experiment described in chapter 6).

5.9 Parameters

5.9.1 Introduction

Systems of this complexity invariably have a large number of parameters, most of which fortunately are not very influential. In this section, most of those that influence the result are listed, with a discussion of their effect. Common values are also supplied. Not listed in this section are the parameters with the greatest influence on the speed of the system. These will be discussed in section 5.11.

5.9.2 Scale

The most important parameter in the recognition system, and the only one involving scale, is the radius r, used in nearly all evaluations. It should be set to a value corresponding to the size of the errors that the recognitions system can be expected to produce. For the DIAC system, a value of 4 mm could be chosen.

5.9.3 Start features

Three choices have to be made when choosing start features. The first is which start preference functions are to be used. For straight lines, the choice is between s_1 and s_2. A reasonable choice is to use s_1, which is most restrictive, and only if that fails, to resort to s_2. The choice can also be made to include other features. For the DIAC problem, only single circles are used (s_3). Combinations of circles (s_4), or circle and straight line (s_5) are not used.

The second choice is the maximum relative difference in length of two edges that are assumed similar, Δ_1. A common choice here is 0.3. The third choice is only made when preference function s_2 is used: the maximum difference in angle between the two edges, Δ_2. This threshold can be set at the arccosine of 0.2.

All these parameters in fact achieve a balance between speed and quality: if they are chosen too restrictive, the result will be fast, but some objects will not be recognised. Trivially, with the thresholds set to zero, no start features will be used, and with these thresholds set to a very high value (>> 1), n (s_1) or n^2 (s_2) will be used, where n is the number of observed straight lines. Generally, if these thresholds are set too liberal, too many hypotheses will be generated. However, within an interval around the values listed here, they do not appear to be critical. With these values, chosen at an early stage, the system performed satisfactory throughout the

experiments described in chapter 6).

5.9.4 Matching straight lines

When matching straight lines, the following thresholds are imposed:

- The distance between straight line end points and the other straight line, in other words the width of the cylinder in Fig. 5-5, is determined by the scale parameter r. This is formulated in equation (68).
- The displacement of the end points of one edge over the other is determined by parameters λ_1 and λ_2, as defined in equations (69) and (70). These are usually set at 0.2 and 1.3, respectively. These parameters, that determine how much edges should overlap in the direction of the model edge, are chosen to allow large displacements in case broken-up edges occur. However, as these usually have a low contribution to the support of a match, these parameters are not critical.

5.9.5 Matching circles

Matching circles involves the scale parameter r again, but there also is a threshold on the relative difference of the circle radii involved. This threshold, ζ, which is not critical as it only limits the contributions of badly corresponding circles, is set at 1.3.

5.9.6 Removing bad hypotheses

There are three parameters involved here:

- One determining the relative support below which a hypothesis is immediately rejected. This parameter, δ_1, has a direct influence on the number of errors, and is the one varied in the experiment in chapter 3. A typical value is 0.01, this means that hypotheses are rarely rejected on this criterion.
- One determining the relative overlap between two hypotheses, above which they are considered to be conflicting. This parameter, δ_2, is also of great influence. A typical value is 0, which means that any overlap leads to rejection of hypotheses.
- A third parameter, δ_3, determines when the two hypotheses considered are too close to decide which one is rejected without further information. This relative parameter, which is not very critical, is set to 0.05.

5.9.7 Use of world information

The use of object rigidity requires no parameters other than the ones used above for removal of bad hypotheses. The use of information about the supporting surface does use some parameters:

- A description of the supporting surface must be given. As implemented, this means the specification of a z-value.
- Two factors, μ_{below} and μ_{above}, that describe how many times the scale factor the lowest point in an object is allowed to be below or above the supporting surface, respec-

tively. These can typically be set at 5 and 10 times the scale parameter, so that inaccurately detected positions will only rarely lead to false rejections.

- Two factors, α and β, that specify the relative weight of positions below and above the supporting surface, respectively, when using this information. These are typically set at 0.1 and 0.01, respectively.

5.9.8 Summary

A table listing all the parameters described in this chapter is given as Table 5-1. Obviously, for such a large number of parameters, it is impossible to do a simultaneous optimization. Fortunately, most parameters are not critical, and only specify another criterion for rejection. These were just set to a reasonable value. Critical parameters were set on a trial and error basis over a large number of images.

Table 5-1 Accuracy determining parameters.

Name	Description	Typical value
r	Scale	4 mm
	Set of start preference functions	s_1,s_2,s_3
Δ_1	Relative length difference of edges	0.3
Δ_2	Cosine of angle difference	0.2
l_1	Lower edge displacement limit	0.3
l_2	Upper edge displacement limit	1.2
z	Relative radius difference of circles	1.3
δ_1	Hypothesis reject threshold	0.01
δ_2	Hypothesis overlap threshold	0
δ_3	Reject criterion parameter	0.05
z_{table}	Height of supporting surface	
μ_{below}	Lowest permissible point below the surface	5
μ_{above}	Lowest permissible point above the surface	10
a	Weight of point below the surface	0.1
b	Weight of point above the surface	0.01

The scale parameter, also set in this way, is of course closely linked to the performance of the stereo vision system. This means that the recognition system cannot be evaluated in isolation. In chapter 6, a large experiment is described that allows an evaluation of the system perform-

ance as a whole.

5.10 Example

The recognition system can be illustrated with the following example. Using the image pair shown in Fig. 4-13, the recognition system was started with a set of 4 known models. Two of these were the DIAC objects 10 and 11 as described in chapter 3, the remaining two a cylinder and a block used with the robot. During 5 s of CPU time on a Sun 4/330 workstation, a total of 80 hypotheses were generated involving both DIAC objects. The edges of the highest object on the side turned to the bottom left hand corner, and of the lowest object the edges on the same side and on the bottom, were used as starting features.

In Fig. 5-11, the top row of images shows the 3-d edges output by the stereo vision system. Two of the false hypotheses involving the highest part are shown in the middle row. Based on two edges of one side, one hypothesis puts the image on the wrong side of the plane of that side, while the other hypothesis has the object (which is close to square) rotated 90 degrees. Similar wrong hypotheses were also generated involving the other part.

After the first removal of overlapping hypotheses, only two remained, which after refinement are shown in the lower row, for both images of the stereo pair. The two objects were recognised correctly.

5.11 Computational complexity/ improvements

5.11.1 Identification of bottlenecks

The speed of the recognition system depends on a few quantities that are under user control, and should be carefully controlled in order to have adequate performance. These are:

- The quality of the observation. Edges and circles that are too small to be accurately detected, and features that lie outside the area in which objects can be recognised, should be avoided. This is mainly the task of the stereo vision system. The time used to evaluate a hypothesis is on the order of the square of the number of features detected, and more hypotheses may be generated by larger observations.
- The number of models should be kept as small as reasonably possible, and no irrelevant information should be included in them (like edges that are too small to detect). The time used to evaluate a hypothesis is again on the order of the square of the number of model edges. The number of hypotheses that can be expected to be generated is also on the order of the number of models. reducing model complexity is mainly the task of the model generation system.
- When models and observation are given, the first factor to determine the performance of the recognition system is the number of start features considered. The maximum number of start features used is a parameter available to the operator. It is set to 10 for the scenes described in this thesis. For very complex scenes it should be increased.
- The number of start features for which corresponding model features can be found is

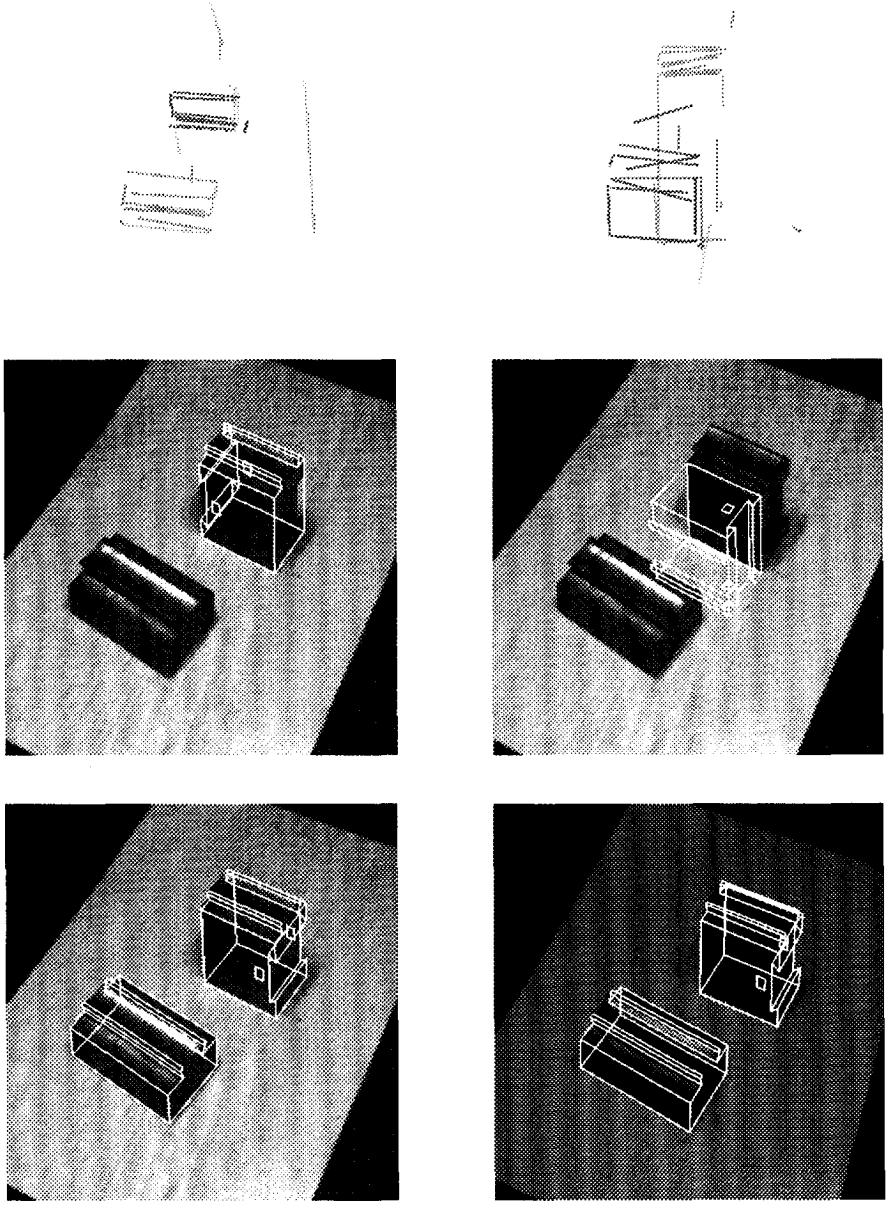


Fig. 5-11 Some results of the recognition system. Top row: the input data from the stereo vision system, seen from two different viewpoints. Middle row: two examples of wrong hypotheses that were evaluated. Bottom row: the accepted hypotheses overlayed on the original stereo vision input pair.

of course smaller than the maximum set previously. However, not all of them have to be used, as some will correspond to the same object. A parameter is the number of start features, for which corresponding features can be found, that are actually used. This number is set to three for all scenes described in this thesis. Note that this number imposes a maximum on the number of objects that can actually be found in one scene.

- Finally, the number of feature pairs, pairs of a start feature and a corresponding model feature, is also limited. As each of these pairs generates a hypothesis, this parameter imposes a hard limit on the number of hypotheses generated for each model, thus limiting memory requirements as well. For all the scenes in this thesis, it is set to 1000 per model, a value which is rarely reached (e.g. in the experiments described in chapter 6, the mean number of feature pairs per model was about five, the highest observed value was 120).
- A number of parameters that influence the qualitative performance of the system also influence the speed. For instance, the constraints on hypothesis overlap, position with respect to the table, and of course the scale parameter r have a significant impact on the amount of hypotheses evaluated. Careful tuning is required, using a debugging mode in which the system logs intermediate statistics and timing. No complete recipe can be given for this.

5.11.2 Improvements

A major improvement on the speed of the system could be obtained using indexing, see e.g.Clemens and Jacobs[23]. Here, a table is compiled beforehand (off-line) of relevant features of models, in such a way that for each start feature a list of relevant model features can be found by a simple table lookup instead. This could be implemented by using the length of two edges as well as their angle in a three dimensional table, and the radius of a circle in second, one-dimensional one. This seems to correspond to the notion of perceptual grouping as suggested by Clemens and Jacobs.

Another improvement can be obtained by exploiting the inherent parallelism in the algorithm: both the steps of hypothesis generation and of hypothesis testing can be parallelised. In the former case, this can be exploited by using a number of processors, each of which gets the task of finding the feature pairs (and hence the hypotheses) corresponding to one or more models (as illustrated in Fig. 5-12). In the latter case, each processor can get the task of evaluating one or more hypotheses. All models must be known to each processor in advance. The master processor must be able to request additional information about each hypothesis evaluation for the removal of bad and conflicting hypotheses. This is illustrated in Fig. 5-13. Only the final step of removing overlapping hypotheses cannot be parallelised completely, although it can to some extent.

5.12 Conclusion

In this chapter, the recognition system has been described qualitatively. It is based on hypothesis verification and testing, using careful pruning to reduce the computational complexity.

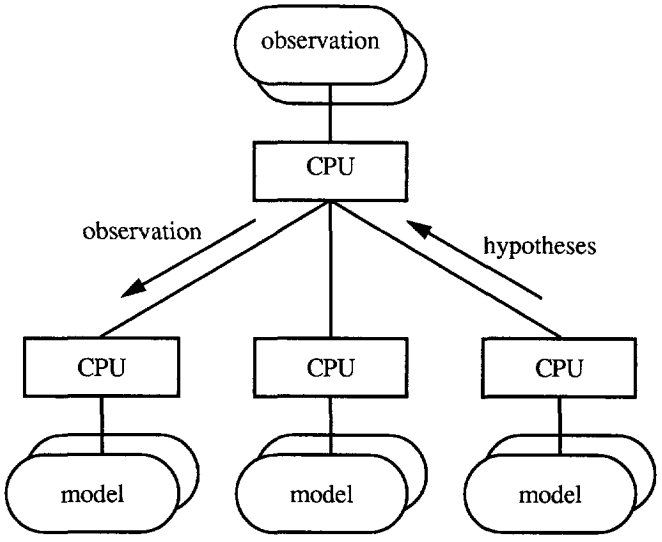


Fig. 5-12 Parallelising hypothesis generation. Each CPU deals with one or several models. The master CPU transmits the observation to each CPU and receives the corresponding hypotheses back.

Rotational symmetries, included in each model, are eliminated at an early stage of recognition. Knowledge about the rigidity of the object and the supporting surface is also exploited.

The meaning of the parameters has been explained, and their influence both on the recognition result as well as the computation time. The operation of the system has been illustrated with an example. In this case, two of the DIAC objects were recognised after five seconds of CPU time (and 32 seconds for image processing and stereo vision). It has been shown that the system lends itself well to parallelisation.

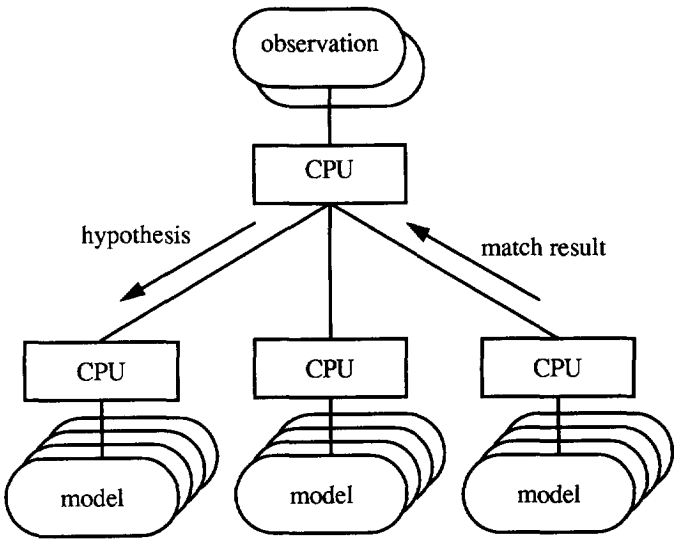


Fig. 5-13 Parallelising hypothesis testing. Each CPU has all models. The master CPU transmits the hypotheses to each CPU and receives the corresponding match result (score and, upon request, corresponding points) back.

6 An evaluation of the whole system

This chapter describes an evaluation of the complete system, the combination of the stereo vision and the recognition algorithms. The evaluation is performed using a standard data set, representative of the DIAC parts, and focuses on the correctness of the detected objects. An evaluation of the positional accuracy of the system has not been performed.

6.1 Introduction

Although the stereo vision system could exist without the recognition system, the reverse is not true: design and performance of the recognition algorithms are directly related to the algorithms used to acquire data. In this system, an evaluation of the complete system, stereo vision and recognition together, is attempted. The main question is: can correct recognition be achieved?

As all systems described in the literature are applied to different problems, it is hard to give results that are comparable to those of other researchers. In robot vision, there is no such thing as a standard data set, or even a standard problem! The best one could do is set up a reproducible experiment and make the data available to other researchers, hoping that others will apply their algorithms to it. The data set described in this chapter is preserved on magnetic tape, including the relevant geometric data. This means that such an exchange could take place.

The following properties of the system are evaluated:

- Overall performance on a large set of images of DIAC objects.
- Efficiency of the use of symmetries.
- The influence of the acceptance threshold on the performance.
- The influence of the scale parameter on the performance.

Each experiment is described in a separate section of this chapter. Finally, some conclusions and recommendations for further research are drawn.

The evaluation focuses on the correctness of the identity of the detected objects. Not evaluated is the accuracy of the position and orientation of the objects found, or even the stable position that is detected. Although some information about the actual location of the objects in the data set is present, it would still require a large effort to make this knowledge explicit.

It should be pointed out that the recognition system as tested uses no knowledge about the number of objects present in the scene. It allows for the presence of zero or several objects. In the data set presented here, each scene contains exactly one known object. An experiment to evaluate the performance for more complex scenes has not been performed.

A consequence of this property of the recognition system is that in the experiment described

in this chapter, the sum of the correct classifications, incorrect classifications and image pairs with no objects detected is not equal to the total number of image pairs. The following definitions will be used for each:

Correct - The correct object was among the accepted hypotheses. Note that the orientation and stable position are not evaluated.

Incorrect - A hypothesis that was accepted belonged to the wrong model.

No detection - No hypotheses were accepted.

From these definitions it follows that a scene could cause zero or one correct classifications, and zero or more incorrect classifications. No detection applies if neither a correct nor an incorrect classification occurs.

6.2 Description of the data set

The full data set consists of 200 stereo image pairs, obtained using the stereo vision cameras described in chapter 4. Most of the DIAC parts were used (some were not available at the time), each of which in one, two or three different stable positions. The number of stable positions used depended on the shape of the object, such that only realistic positions were used. For each object in each stable position, eight image pairs were obtained, where the object was rotated around a fixed point over 45 degrees between each image pair. These rotations were done by hand, using lines drawn on the supporting surface. The whole set is summarized in Table 6-1. The left hand image of the first pair of each series of eight is shown in Fig. 6-1.

Table 6-1 Listing of the full image set

DIAC part no.	No. of stable positions	No. of image pairs	Images	Details
7	2	16	1-16	small
8	3	24	17-40	
9	2	16	41-56	
10	3	24	57-80	
11	2	16	81-96	
14	2	16	105-112, 177-184	cylindrical, transparent
16	1	8	113-120	small, cyl.
17	2	16	185-200	cylindrical
18	1	8	121-128	small, cyl.
24	2	16	145-160	cylindrical
27	1	8	97-104	small

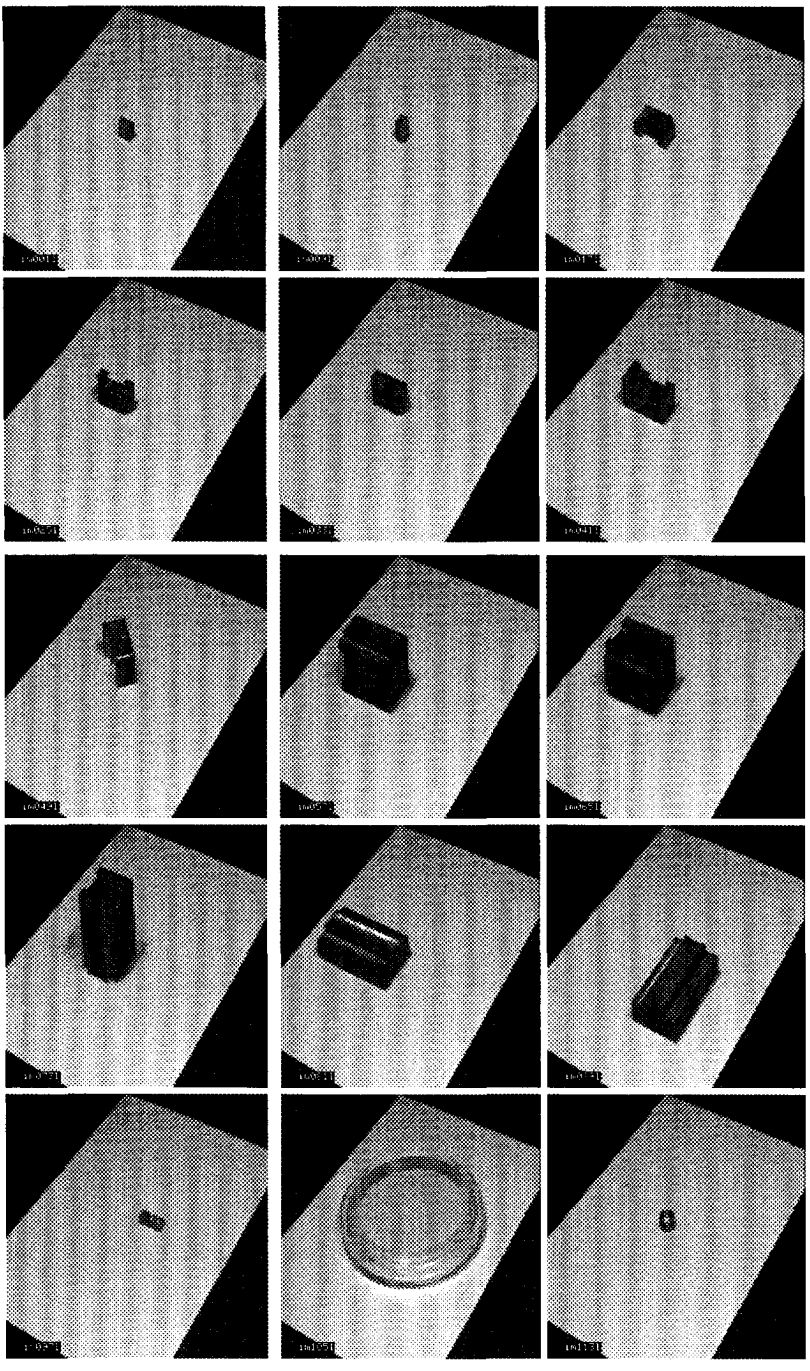
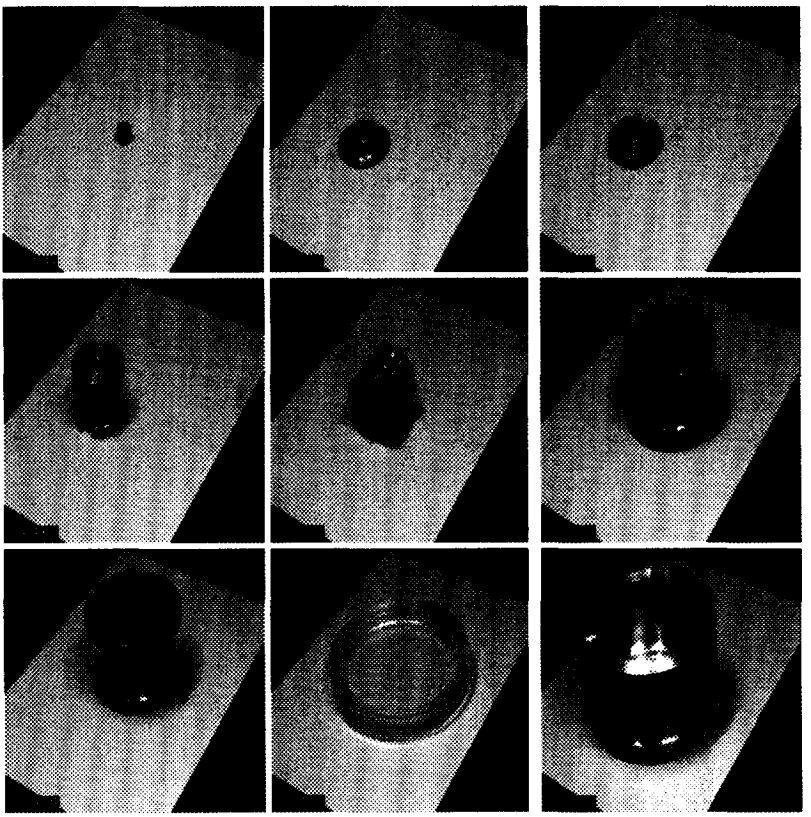


Fig. 6-1 The left image of the first pair in each series of eight, showing all objects and the stable poses considered. Shown to the right is the part number for each object. The order of the original data is used, which was motivated by practical considerations (see also next page).



18, 28, 28

24, 24, 29

29, 14, 17

17

Fig. 6-1 Continued.

Table 6-1 Listing of the full image set

DIAC part no.	No. of stable positions	No. of image pairs	Images	Details
28	2	16	129-144	cylindrical
29	2	16	161-176	cylindrical

Of these 200 image pairs, 104 are of objects that are mostly prismatic, and 96 of cylindrical objects. Forty are of objects that can be considered small. Two sets of 8 images are of part 14, which is transparent. The images have been obtained using the prototype setup described in chapter 4. Each image is 512 times 512 pixels in size, where one pixel is 8 bits (one byte). As there are 400 images, the data set is in total 102,400 kB (100 Mb) in size. This data set is representative for the total set of DIAC objects.

Some information is present about the position and orientation of the objects. All objects are positioned at a fixed point, at eight fixed orientations per stable position. As the pencilled lines used for positioning the objects show up well in the image set, their location and orientation can be determined accurately. A person looking at the images and the models should be able to determine the correct mappings for each case. However, as this is a cumbersome procedure, it has not been done as part of this research. As a result of this, the evaluation can only consider the identity of the objects found in the scene, and not the position, orientation or stable position.

6.3 Overall performance

6.3.1 Summary of the result

All image pairs in the set were subjected to the recognition system, and the results are summarized in the confusion matrix Table 6-2. This table lists for each of the models, how image pairs corresponding to an object of that type is classified. The rightmost two columns list the number of image pairs in which no object was found, and the total number of entries (correct, incorrect, no detection) for that model. Note that there is no distinction between the stable positions of each object, because the recognition system does not supply that information. As the examples in Fig. 6-2 show, a number of parts were detected correctly.

Table 6-2 Confusion matrix for the full data set.
(The number of times that in an image of object i, object j was detected).

i\j	7	8	9	10	11	14	16	17	18	24	27	28	29	none	sum
7	3	1									1			11	16
8		11	3							4	1			5	24
9			15	1											16

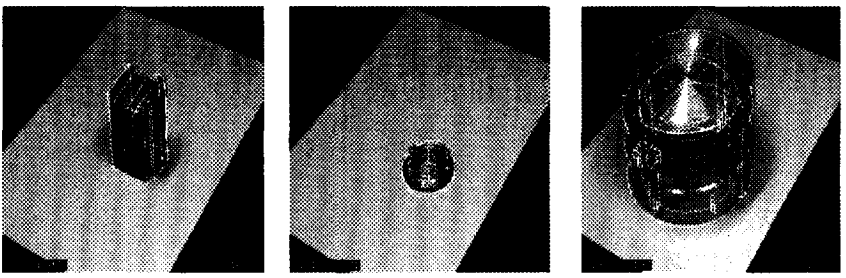


Fig. 6-2 Some examples of objects in the set that were detected correctly.

Table 6-2 Confusion matrix for the full data set.
(The number of times that in an image of object i, object j was detected).

i\j	7	8	9	10	11	14	16	17	18	24	27	28	29	none	sum
10				19	1									4	24
11					15						1				16
14	1	1		1		8		2		1		1	5	3	23
16	1													7	8
17	1		1			3		6		1			2	4	18
18						1								7	8
24	1		2		1	1		1		3		4		7	20
27											6			2	8
28										1		7	1	7	16
29				1	1	1		7					6	1	17

From the set of 200 image pairs, in 99 cases the correct object was found. (49.5%). In 58 image pairs, no object was detected, whereas 57 times the wrong object was reported. In 12 cases, two objects were detected and in one case three. All of the multiple detections occurred in images containing cylindrical objects, in half of the cases the transparent object 14 was involved. This suggests that detections of extra objects result from cylindrical edges for which the corresponding circle could not be found.

From the table it follows that objects 16 and 18 cannot be detected at all (these are small, cylindrical parts) and objects 7 and 24 badly (below 25% correct), the former a small prismatic object, the latter an elongated, medium-sized cylindrical part. A valid conclusion of this research might be that these objects cannot be detected using this system. Apparently, an object needs to be more then 40 pixels in size in order to be recognized by this system (the failure for object 24 will be discussed below). For the DIAC cell, this means that either small

parts have to be fed by special equipment or fixed on pallets (so that vision is not necessary), or that the resolution has to be increased.

To account for this, a reduced data set has been set up, excluding the parts 7, 16, 18 and 24. Using the reduced set, the results become 93 out of 152 correct (61.2%), with 43 wrong and 26 times no object found at all. Prismatic parts (75%) were detected much better than cylindrical parts (42%). As these results are still unsatisfactory, a discussion on the causes of errors is required. This is done in the next section.

6.3.2 Causes of errors

The following major causes of errors can be indicated:

- *The view is too difficult.* An example is shown in Fig. 6-3. The object shown (part 10) has its most difficult side, the swallow's tail (difficult because of the large number of real edges as well as the shading) pointing towards the camera. Furthermore, the contour of the object has corners that are very blunt, causing errors in the straight line detection algorithm. As a result of this, no good hypotheses could be generated.

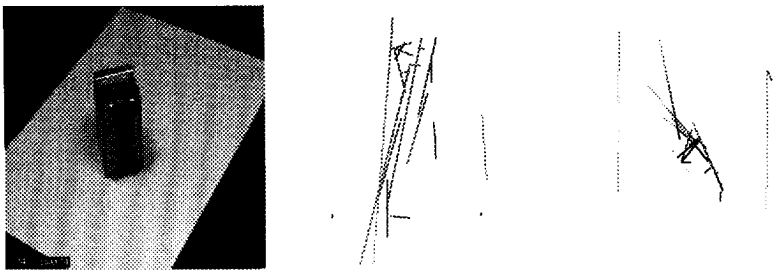


Fig. 6-3 An example of a view that prooved to difficult, with side and top views of the straight lines found. The diagonal lines are false detections.

  Solutions for this problem would be to change the position of the lighting and/ or the camera. The former could make other edges show up in the image contour, which could generate the right hypothesis. The latter, more expensive solution would allow an active search for the best view.

- *Another part fits the observed part.* A good example is Fig. 6-4, where the cylindrical part in the scene (part 29) is mistaken for the hollow cylinder in which it has to be assembled (part 17). This may occur if the stereo vision system found few edges apart from those on the fitting surfaces, and therefore only occurs if only a small number of features can be detected (such as for some cylindrical parts, where only one circle may be detected). This is one of the more common errors with part 29 (7 out of 16 image pairs), as suggested by Table 6-2, but is more rare the other way around (twice in 16 image pairs). As fitting parts are (hopefully!) common in an assembly cell, this is a serious problem, caused by the fact that the model does not have information about the possible views of an object, and thus a circle on the inside is as likely to match as a cir-

cle on the outside.

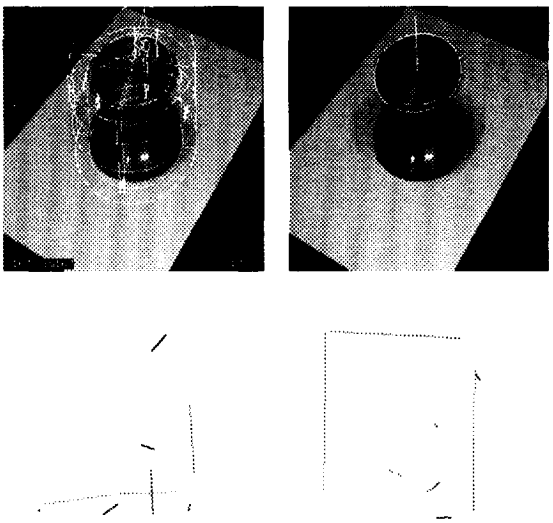


Fig. 6-4 An example of a case where the hypothesised object fits the object that is actually visible. Top left: left image (and detected model), top right: image with circle found superimposed. Bottom row: side and top view of the straight lines found.

There are a number of solutions for this problem. First, in this particular case, the hollow cylinder has a bottom that, in the hypothesized position, must be slightly below the supporting surface. Strengthening the constraint that aims to prevent this might solve this case. A more general solution would be to include in the model information about the likelihood of the visibility of an edge, in which case the model in which the edge is exterior would be preferred over one in which it were interior. The most drastic step would be to model all the different views of an object. This would be a fundamental difference, and is discussed in the final section of this chapter.

- *Circles were not detected correctly.* A good example is shown in Fig. 6-5 (part 14). As the performance on cylindrical parts (42% correct in the reduced set) shows, the ellipse based stereo procedure in its present form is still too crude for reliable detection of the diac parts. In Fig. 6-5, the problem is particularly difficult as the transparent part contains a large number of similar circles. In this case, no circles were detected correctly (one falsely), but some straight lines fitted to edge fragments actually caused a completely different object to be detected.

Related to this last point is the problem with part 24. This part (see Fig. 6-6) is cylindrical, yet its largest dimension consists of straight lines. The most visible ones are not real edges, but are caused by the round surface. As a result of this, straight lines are hard to detect. Furthermore,

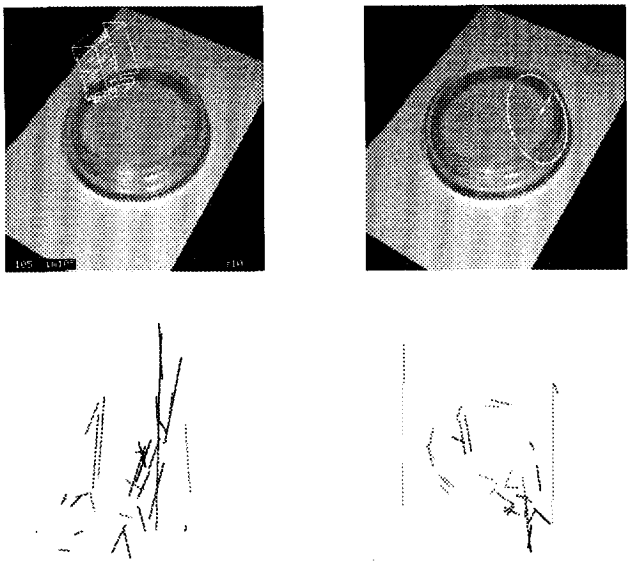


Fig. 6-5 One example where circles were not detected. Top left: left image of part 14, top right: circle detected (a bad detection). Bottom row: side and top views of straight lines detected. As the bottom row shows, the edges belonging to undetected ellipses result in a lot of spuriously (and erroneously) detected straight lines.

the shape of the circles that constitute the ends of the cylinder is broken up in a number of places, with only a few pixels distance. This means that detection of the circles is a difficult problem too.

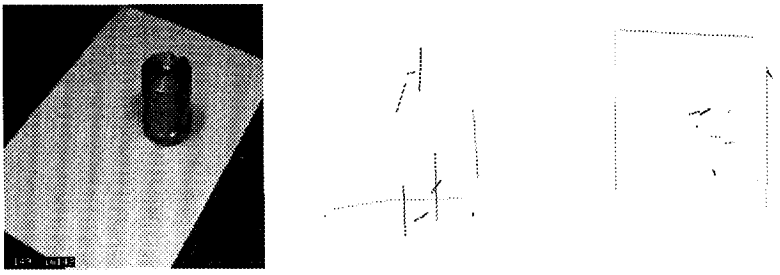


Fig. 6-6 Another example of a case where circles were not detected. From left to right: image of part 24, side view and top view of straight lines found. No circles were detected..

As already remarked in chapter 4, ellipse fitting is certainly not the best way of detecting ellipses, although it is a fast one. As the present solution is not sufficiently robust, a better method should be implemented. This would improve the performance on cylindrical objects.

Objects like part 24 can also be recognised easier by means of an active search for more edges, as described above.

6.3.3 Execution times

On a Sun IPX workstation, the execution times were measured over the whole set for stereo vision and recognition separately. Image acquisition was from disk and was not included, nor was the display of the results. Statistics about these times are listed in Table 6-3.

Table 6-3 Execution times in seconds for the full data set

N = 200 objects	Stereo vision	Recognition
Minimum	16.0	0.06
Maximum	20.4	23.7
Total	3383	697
Mean	16.9	3.5
Standard deviation	1.2	4.3

As the table shows, the execution time for stereo vision was nearly constant over the entire set, with the highest times occurring in the images where most ellipses were present, such as for parts 14 and 17. The mean time was 16.9 s for the complete stereo vision process. It should be pointed out that this execution time applies to the version of the system using interpreted TCL-Image command files. A recent port to compiled C code should be faster, but no times have been reported yet[68].

The execution time for recognition shows a much more random behaviour, depending not just on the contents of the image but much more on the number of hypotheses that could be generated. Although the mean time was only 3.5 s, in some cases almost seven times as long was required.

The different behaviour of the execution times of stereo vision and recognition is illustrated in Fig. 6-7. The distribution for stereo vision actually shows two peaks, one for objects with and one for objects without large circles. Yet both peaks are narrow. The distribution for recognition shows only one peak as expected, but the curve only becomes flat for very high times This behaviour is typical for recognition systems based on combinatorics.

6.4 Use of symmetries

The use of symmetries has been evaluated in a very simple way: by processing the full data set with and without the use of symmetries. The execution times of the recognition system were integrated and the results are shown in Fig. 6-8. Note that the execution times here are higher than those mentioned earlier, as this experiment took place on a Sun Sparcstation 1+. In the figure, the distance between the two curves gradually increases, suggesting that using symmetries offers an advantage for most of the objects concerned. In a few isolated cases the use

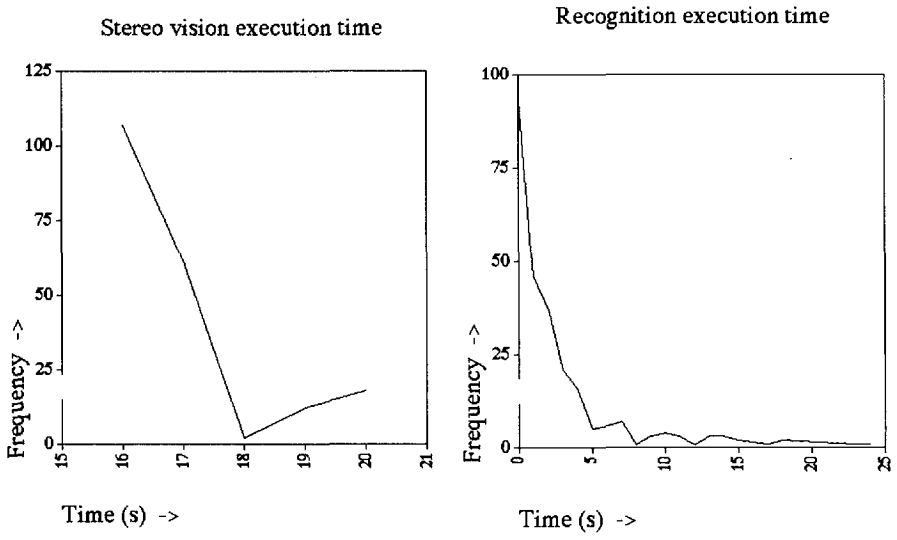


Fig. 6-7 The execution times of the stereo vision (left) and recognition (right) algorithms over the full data set.

of symmetries caused a slight decrease in speed, however the overall gain was 17%.

6.5 The reject threshold

The reject threshold δ_1 has a direct influence on the numbers of correctly and incorrectly recognized images. This has been evaluated by applying the recognition system a number of times to the full and reduced data set for different values of the threshold in the range from 0 to 0.2. The resulting numbers of correctly detected objects, incorrectly detected objects and images without any objects detected (note that the sum of the first two is not equal to the third, as explained earlier) are shown in Fig. 6-9.

These graphs show that as the threshold increases, the number of correct detections decreases monotonically as could be expected. However, the expectation that the number of false detections should also decrease monotonically does not come true. Instead, there is a slight increase in the number of incorrect detections, caused by extra detections in images where some objects are already detected at threshold 0. This phenomenon can be explained as follows:

These new extra detections are probably caused by the order in which the constraints are applied. Assume we have three hypotheses A, B and C, of which A and B are conflicting; A is a smaller object than B. Hypothesis A has less support then B in an absolute sense, but a larger support relative to its own size. Furthermore, hypothesis B violates the rigidity constraints with respect to hypothesis C, a constraint which is evaluated at the end. C is the best supported hypothesis.

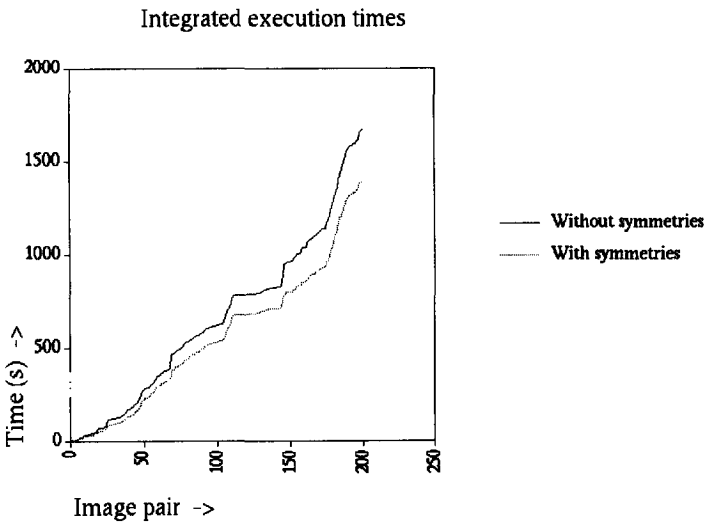


Fig. 6-8 Integrated execution times over the entire set for recognition with and without use of symmetries. Use of symmetries offers an improvement of about 17 %.

If the reject threshold is low, then all hypotheses are accepted at first. Hypothesis B will suppress A, only to be removed afterwards by the rigidity constraint (as a result of C). In this case no incorrect detection takes place. If the threshold is higher however, hypothesis B will be removed and A will remain. In this case, an incorrect detection (A) is found. For an even higher value of the reject threshold, A and B are both rejected and only C will remain.

This behaviour results from the order in which the constraints are applied, which was amongst others based on efficiency considerations. After all, the rigidity constraint is expensive to compute. Although this seems rather counter-intuitive behaviour, it results from mechanisms in recognition that usually work well, and it has only a minor impact on the result. Therefore, this is not considered a problem. However, as the number of correct classifications does not change for low values of the threshold, depending on the exact weights assigned to incorrect classifications and no classifications, the choice of 0 for the threshold may be the best one.

6.6 The scale parameter

The scale parameter, r, is the only parameter in the recognition algorithm that has a dimension. It affects all geometrical thresholds in the recognition system. The performance of the system for different values of r has been calculated for the full and reduced data sets and is illustrated in Fig. 6-10. It follows from this figure that the value of 4 mm for r, used by default (see Table 5-1) and found by tuning the system, is the value for which the system performs best on these data set.

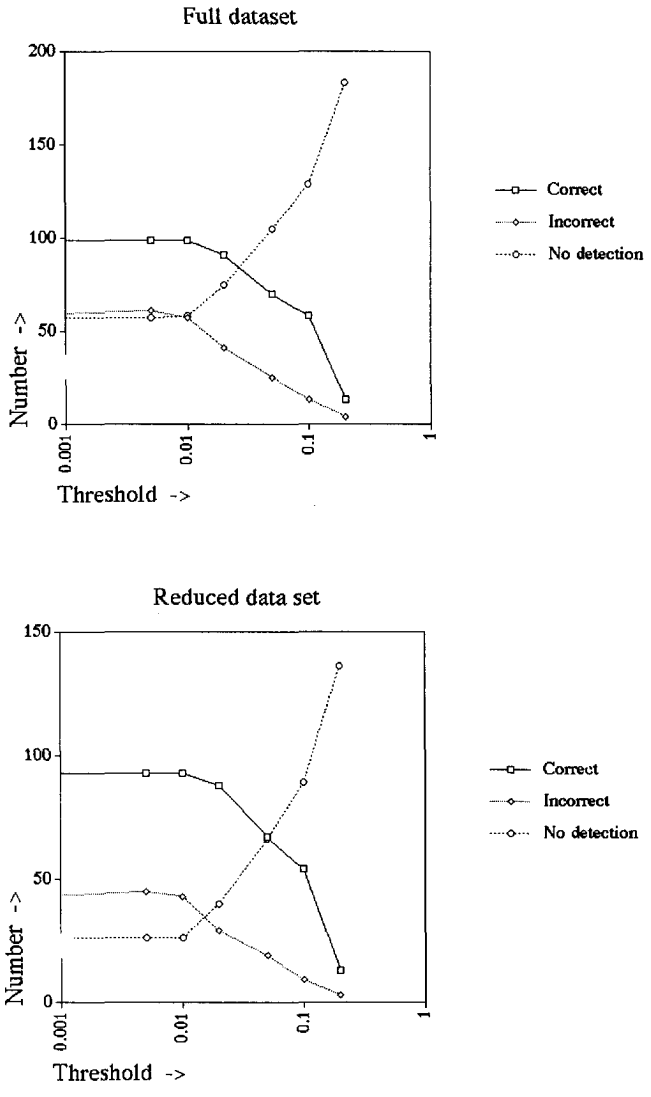


Fig. 6-9 The numbers of correct detections, false detections and images where no objects were detected, as a function of the acceptence threshold, for the full data set (top) and the reduced data set (bottom).

The expected behaviour is that for low values of r, few features will be found similar and thus, only a few correct and incorrect classifications will be found. For values of r that are too high, the system will not be able to distinguish right hypotheses from wrong, and the number of correct classifications will drop.

Although these trends are confirmed by Fig. 6-10, the question remains why the number of incorrect classifications also drops with increasing r. Apparently, besides the shift from correct to incorrect classification there is another, larger shift from incorrect to no classification. This may be explained by the fact that more rejects are based on the rigidity constraints.

6.7 Conclusions

In brief, the conclusions of this chapter are the following:

- The whole system was able to recognize the prismatic objects in the reduced set with 75% success, and the cylindrical parts in the reduced set with 42% success. Objects that are too small (less than 40 pixels) cannot be detected at all. Objects with certain shapes can be difficult even when they are larger.
- Some views of objects are hard to recognize. Changes in lighting and/ or camera viewpoint may improve this.
- Sometimes the inside of a model fits the outside of an observed part. This can be resolved by extending the model with information about insides and outsets, or by exploiting constraints to a larger extent.
- Ellipse-based stereo is not yet robust enough for this application. However, a better ellipse detection algorithm is likely to solve that.
- The use of model symmetries, which is original in this work, offers a 17% speed gain over the full set.
- The order in which constraints are applied may lead to acceptance of bad hypotheses in some cases. Depending on the cost assigned to incorrect detections and no detections, a different value for the reject threshold may have to be chosen. In particular, 0 may be a good choice.

In one case though (a badly observed object is recognised as another, fitting object), the failure of the vision system is caused by a fundamental design choice: viewpoint dependent information is not included. If this information were computed during recognition, the recognition process would slow down considerably. If it were computed beforehand, the size of the model would grow considerably. Also, the problem of generating all possible views (or all relevant views) of an object is a complex one, and general solutions do not yet exist [21]. As a result of this, it cannot be concluded that simply using viewpoint dependent information would be better than not using it.

The question remains: are these results satisfactory? In an absolute sense: No, they are not. Even a 75% success ratio is not enough to justify a vision system as complex as this. Oppo-

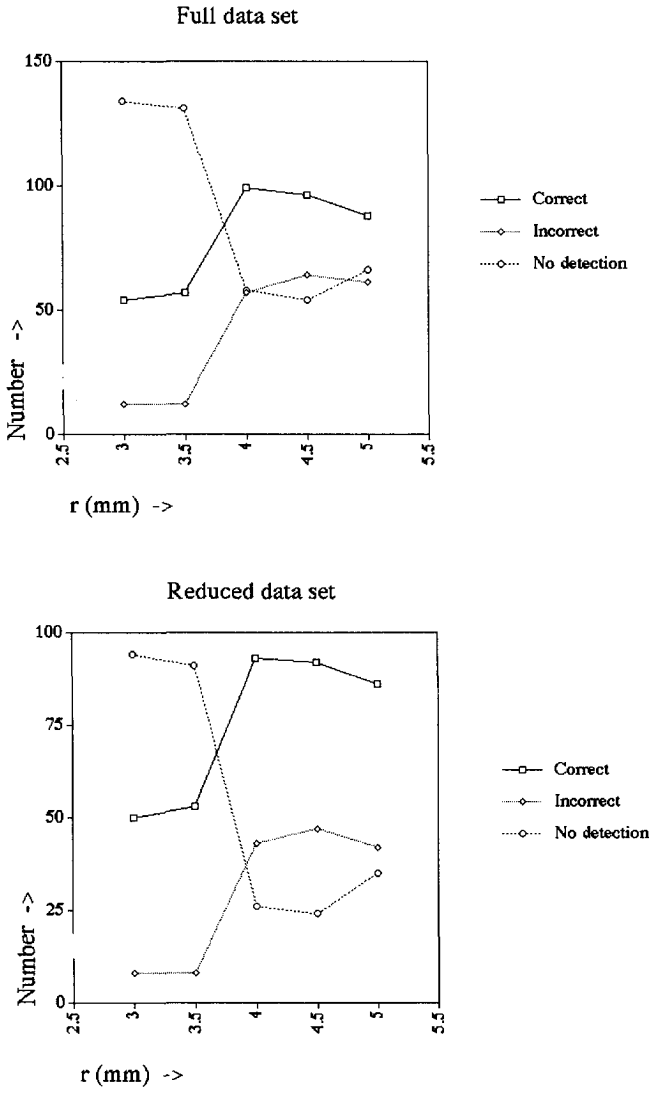


Fig. 6-10 The numbers of correct detections, false detections and images where no objects were detected, as a function of the scale parameter r, for the full data set (top) and the reduced data set (bottom).

nents of the use of complex vision systems (such as Miller[54]) will immediately argue that mechanical solutions (like feeding all parts in fixed positions) can achieve much higher success rates. However, there are still possibilities for improvements of this vision system, such as those mentioned above. These should be exploited before discarding the idea of vision.

In a relative sense, it is unknown whether these results are satisfactory compared to those of others in the field. Large-scale evaluations are not common in literature, and they are even less comparable. The usual report in papers: "The correct operation of the algorithm is illustrated with the following example...", does not exclude the possibility that an algorithm has a success rate much lower than those found in this chapter.

A third point is, are these results satisfactory for DIAC? In the DIAC cell, this system will be used complemented by a fast verification system, see chapter 3. The recognition system will only be applied to those cases that verification could not handle, and as the performance of the verification system has not been evaluated yet[11], it is not known what kind of cases these will be. It may be that the data sets used in this chapter are not representative for these cases.

Also of importance for DIAC is the flexibility of the vision system, which has only partially been exploited so far. The possibility to recover from unexpected situations using vision may outweigh the cost of a vision system for unmanned operation. Perhaps this system's possibility to handle scenes with an unknown number of objects is of specific value here.

A possibility hitherto unmentioned is that in the DIAC application, the verification system can be used to check the output of recognition for any incorrect detections, thus improving upon the recognition results.

7 Model generation

This chapter addresses the problem of generating the models for recognition of a set of objects. The chapter has two main parts, describing different sources to obtain the models from. Each of the sources has been studied.

In the case of DIAC, the recognition models should be derived from CAD models, a field in which much research is being done currently. This procedure is described in the first part of this chapter†.

The second, and largest part of this chapter is dedicated to the problem of generating these models from observations‡. Parts of this work were reported in [42] and [43].

7.1 Introduction

Model generation can be defined as the process of gathering explicit knowledge about objects, to an amount that is at least sufficient for object recognition using a vision system. The complexity of this problem varies with the application. In DIAC, where the objects have been designed using a known CAD system, the models have to be extracted from that system. This is considered an important problem in the literature[61]. However, for DIAC the problem of model generation is reduced to that of selecting the relevant features from the CAD database, with only slight additional processing. This is caused by two facts:

- The information on the parts to be recognised is complete with respect to the geometry of the objects, because we have all the information the designer supplied in order to manufacture the parts. Additional information, about lighting conditions, reflectivity, or camera viewpoints, does play a part in the stereo vision process described in chapter 4, but not in the recognition process described in chapter 5. Therefore this additional information is not needed for model generation.

- The recognition process, as described in chapter 5, does not make use of viewpoint dependent information, for reasons explained in section 5.1. Therefore there is no need to extract the different views from the models, as opposed to for instance the approach described in [12].

As a result of this, for the DIAC recognition system model generation is fairly straightforward, and will be dealt with in section 7.2. There are however cases where model generation is more complex then that. Two such cases are dealt with in this thesis:

- If a CAD database is not available, or the CAD database does not supply sufficient informa-

†. The interface with the PDM described in this chapter was defined during discussions with DIAC-colleagues Peter Martens and Dave Bierhuizen.

‡. The author would like to acknowledge the important work done by Johan de Jong, and the use of his Soma-puzzle.

tion, models cannot be obtained from it. In such a case, it is possible to learn models using the vision system. Sections 7.3 to 7.6 of this chapter are dedicated to this approach.

- If viewpoint dependent information is needed for the recognition algorithm, a set of different views must be generated whenever new objects are introduced to the system. The problem of which and how many views should be used is a common problem in Computer Vision.

7.2 Interface to the DIAC CAD database

7.2.1 The Product Data Model

For the standard DIAC application, model generation can be described using Fig. 7-1. The models (even those of the nearly cylindrical parts) are represented as wireframes. The matching algorithm is capable of comparing an observed wireframe with a set of model wireframes and selecting the best match. It is the task of model generation to generate each of these wireframes. The CAD database contains a description of the objects using a large number of geometric primitives, including parametric curves and surfaces. Although the description of the geometry is complete, only little information about other optical properties is present.

The database connects to the object recognition system through the Product Data Model, exhaustively described by Martens[51]. This PDM, which is also used for the generation of assembly plans, contains information about the geometry of parts, materials used etc., using a restricted set of primitives. In particular, parametrisation of curves and surfaces is replaced by an approximation with straight lines and planes, using a resolution specified by the operator. This geometrical information is made available to other processes, like the vision systems.

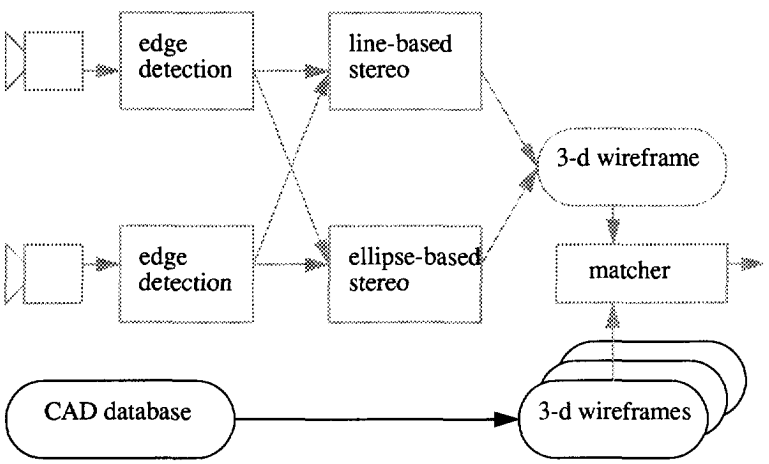


Fig. 7-1 Model generation in the recognition system.

The PDM data structure is defined in [51]. It includes the following geometrical entities:

- *shapes*. A shape describes an entire part. A shape contains lists of surfaces, edges and

points.

- *surfaces*. A surface describes one of the faces of a part as a list of profiles, the first of which represents the boundary of the surface, and any others represent cut-outs.
- *profiles*. A profile is a list of edges, with such an ordering that the edges form a closed chain.
- *edges*. An edge could be a straight line or a curve approximated by straight lines, and is defined by a set of 3-d points.
- *points*. Points are specified in 3-d, with respect to a common origin for the part.

All these entities were extracted from the CAD system. The PDM extends this with a set of circles present in the list of edges, by trying to fit a circle to each edge. As a result of this, each circular edge is present twice in the PDM: once as an edge, using a large number of points to approximate the curve by a polyline, and once as a circle (orientation and radius).

7.2.2 A model for recognition

At this point the model to be used for recognition must be defined. Some of the design decisions of the recognition system are discussed later (in 5.1), so at this point is suffices to summarise them:

- No viewpoint information is required to be available, instead we use a wireframe representation as if the objects were transparent.
- Circle detection is a complex problem. Although circles themselves can be detected well, endpoints of circular arcs are less easily detected. Therefore the recognition system only uses entire circles, even where only partial edges are present.
- The system uses known rotation symmetries of objects to reduce the number of hypotheses to be evaluated.

This leads to the following definition of a recognition model:

A recognition model is a triple consisting of a graph representing polyhedral features, a (possibly empty) set of circles and a (possibly empty) set of symmetries.

The graph representing polyhedral features is a tuple consisting of a list of nodes representing 3-d points and a list of arcs representing lines connecting those 3-d points.

The set of circles is a list of tuples, each consisting of a (3-d) location vector and an (3-d) orientation vector. The latter vector is perpendicular to the plane of the circle and has as its length the radius of the circle. Each tuple represents a 3-d circle, at least parts of which are present in the object.

The set of symmetries is a list of transformations in a representation, described in 5.4.

7.2.3 Generating a recognition model from the PDM

For model generation, the following operations are necessary to transform each part description into a wire frame:

- Extraction of the relevant shape description. Functions to extract the geometrical shape of parts from the PDM were made available by Martens[51] and used by Cruz Picao[24].
- Detection of circles. This step was incorporated in the PDM by Martens.
- Removal of polylines. Polylines are redundant information as they are used to described curved features. As the only occurrences of curved features in the DIAC objects are circles, the circle detection module allows for deletion of the remaining polylines.
- Removal of features that are too small to be detected. The features only contribute unnecessarily to the complexity of the recognition problem and hence to the execution time.
- Detection of symmetries. A symmetry was defined in section 5.6.1 as a mapping of the model onto itself, which is not the identity mapping. The set of all symmetries can be found by applying the recognition program to the model and itself. In this case, strict parameter settings (thresholds very low etc.) are used in order to obtain only exact matches.
- Transformation of the remaining data into a relevant data file.

A set of programs that realise the remaining steps as mentioned above has been implemented, using the recognition program to find the symmetries and the Unix 'awk' tool to do most of the conversions.

The output of these programs is a set of models, suitable for the recognition program of chapter 5. The remainder of this chapter describes an alternative approach: learning the models.

7.3 Learning object descriptions from a set of observations

7.3.1 Introduction

In 3-D computer vision, objects are generally recognized using a model that has been obtained either from an operator or directly from a CAD database. In the first approach, measurements have to be taken and entered into the system by hand. This can be a labour-intensive procedure requiring a knowledgeable operator. In the second approach, the objects have to be designed using CAD, and an interface to the CAD system has to be available. This is not always the case. Therefore, there are situations where neither approach is feasible. In this case, models of the objects can be acquired through learning, by putting the objects in front of the camera and combining the features found in individual images.

However, there is another reason for learning models as well: CAD and operator models are not sufficient to characterize the images obtained from these objects. The imaging process is also influenced by lighting conditions and noise. Some attempts have been made to model lighting conditions (for instance Sato et. al. [61]). Noise however has only been described at the pixel level, the noise in the various structural descriptions of objects generally remains undescribed. Instead, a distance measure is usually applied to allow for small distortions of the

structural description. Models obtained through learning however can include statistical data obtained during the learning process, and can also model the conditions under which learning took place. If these conditions are kept identical between the learning stage and the recognition stage, learned models may provide more information than models provided by CAD or operators.

In the remainder of this chapter, a system is described for learning models of polyhedral objects. Unlike the systems described by Lu and Wong [49], it does not require exact knowledge about object poses. Instead, only some coarse requirements exist. The system obtains a 3-D wireframe model of the object and also acquires statistical data. This is illustrated with a set of objects taken from a Soma-puzzle[26]. The algorithm is described using the L-shaped object shown in Fig. 7-2, while other parts of the puzzle are used in the experiment described in 7.6.

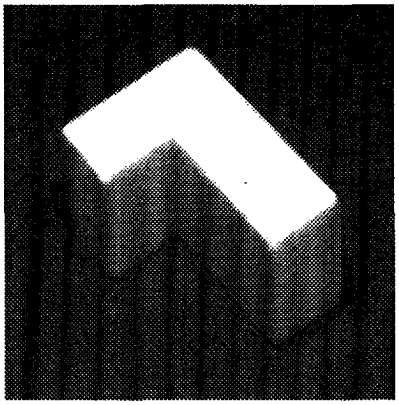


Fig. 7-2 The L-object, used in most of the examples in this chapter. The dimensions of the object are 85 x 57 x 29 mm.

The architecture of the learning system is based on that of the recognition system described in the other chapters of this thesis, and can be explained using Fig. 7-1. The stereo vision system is identical to that described in chapter 4, only it does not use the ellipse-based part in order to find circles in 3-d. Instead, it uses straight lines only. The CAD-based models have been replaced by learned models, that are learned from observations of the stereo vision system. Once models have been learned, they can be used to recognise observed objects using the matcher described in chapter 5.

7.3.2 Outline of the method

The learning procedure described here allows an operator to let the system learn the model of an object. We have used the stereo vision system described in this thesis and in [17], which acquires a 3-D wireframe of the object, which is restricted to straight lines. Each wireframe is represented as a graph, with nodes representing points in 3-dimensional space and arcs representing straight lines.

The procedure is specifically designed so that exact knowledge is not necessary about how the

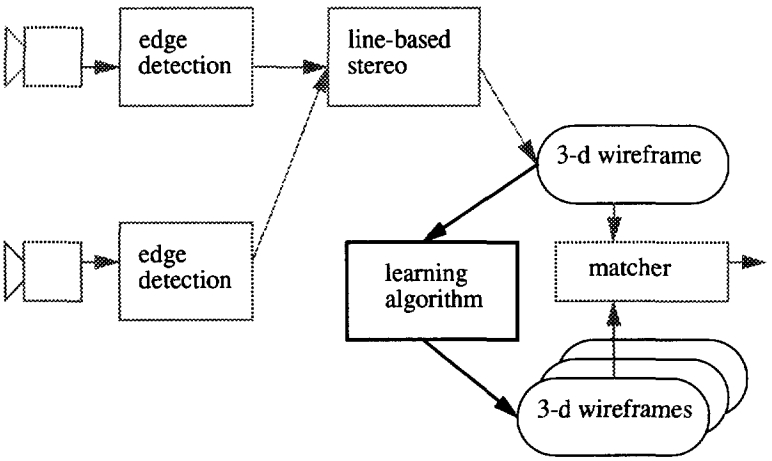


Fig. 7-3 Model generation by learning.

object is placed in front of the cameras, however, in order to reduce the complexity of the procedure some restrictions are imposed. This leads to a two-step procedure: in the first step a number of models is acquired of the object in certain stable positions, and in the second stage these models are combined in order to get a model of the whole object.

In both steps, graphs are merged creating larger graphs, starting with observed graphs at the beginning of the first step, leading to graphs representing stable positions after the first step, and finally to graphs representing the whole object. Both steps consist of a number of occasions where two graphs are merged. Each merger follows the same paradigm as the recognition system: that of hypothesis generation and testing. A set of hypotheses is formed about how the two graphs can be related using only a few features of each graph, then each hypothesis is tested based on the two complete graphs. Finally, the best hypothesis is kept and refined using the information obtained while testing

The procedure involves a few instructions for the operator. For a number of stable positions of the object, the operator must:

- put the object in front of the cameras of the stereo vision system, and acquire a 3-D wireframe;
- rotate the object in the clockwise direction;
- acquire another wireframe;
- repeat the last two steps until the object has rotated a full turn.

The rotation in the second step does not need to be exact, the only requirement is that it does not exceed 180 degrees. If it would exceed a half-turn, some features of the object may not be observed. Usually, smaller angles will be used in order to get better descriptions, a typical

value is 45 degrees. Small translations at this stage are allowed and can be dealt with.

During this first stage, the learning system repeatedly tries to combine two observations of an object in a stable position on a flat horizontal surface. This includes estimating the transformation between the two observations. The observations are made by moving the object and keeping the cameras in the same position, so the transformation can be described by a rotation around the vertical axis followed by a horizontal translation. This procedure consists of four steps, which are described separately in subsections of 7.4:

- forming hypotheses about possible transformations (using the stable pose constraint),
- testing the hypotheses to get a score for every hypothesis, and selecting the hypothesis with the best score,
- optimizing the best hypothesis,
- merging all the observations using the best hypothesis found.

In the second stage, graphs obtained from different stable positions without known relation are combined. Once again, the steps of hypothesis generation, hypothesis testing, best hypothesis selection and hypothesis optimization are followed. This time the stable pose constraint is not available, however, the combined information of each stable position should provide sufficient information for solving the correspondence problem. The second stage is also finished by a merging procedure.

The next two sections are dedicated to the learning procedure, the first to the stage of learning one stable position, the second to the stage of combining several stable positions into a single model. The transformation estimation procedure is described, where it differs from the general procedure of section 5.4. Also, for each stage, each of the steps of hypothesis forming, hypothesis testing and hypothesis optimization are described. The step of best hypothesis selection is trivial after the testing procedure has been described. A section is then dedicated to the merging step. Finally, an experiment is shown, illustrating the different steps.

7.4 Learning stable positions

7.4.1 Obtaining observations

As a first step in the procedure, some observations of stable positions of the object must be obtained. For the L-object, used as an example throughout this chapter, some observations may look like Fig. 7-4. The object has been put on one side, and has been observed four times, while being rotated approximately a quarter turn in the clockwise direction between the observations. Shown in this figure, as well as in all similar figures in this chapter, is a 2-dimensional parallel projection of the actual 3-d graph that has been obtained using the stereo vision system. It can been seen that some edges are missing, while others have been observed with various errors.

As clearly not all object edges are present at least a few times in the set of observations, the procedure has to be repeated for a few other stable positions. The observations for those are shown in Fig. 7-5 and Fig. 7-6. In Fig. 7-5, the object has been placed on four points at the far

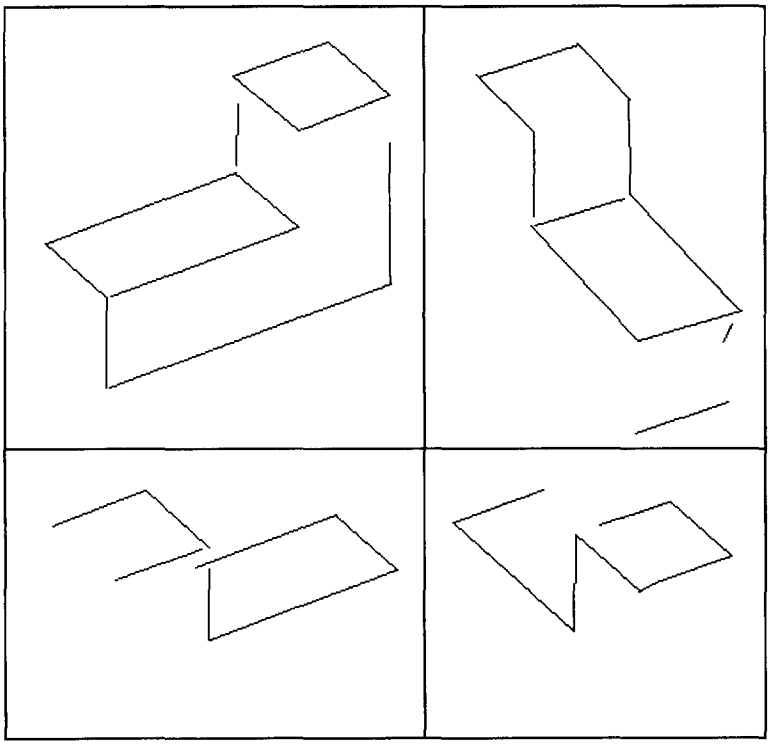


Fig. 7-4 Different observations of one stable position of the L-object (projections of 3-d data).

ends of the L, resulting in observations that are very incomplete. In Fig. 7-6, the object is put on one side. Although the top side has been observed well, the sides and bottom are not present everywhere.

In all these observations, it can be noted that:

- edges are often completely missing.
- edges are sometimes partially found.
- some small edges are found, probably false detections.
- broken edges occur only rarely.

These errors are introduced by the stereo vision system, and explained in chapter 4.

7.4.2 Estimating transformations

As part of hypothesis generation as well as during the refinement step, the transformation between two observations has to be estimated. In the usual 3-d case a transformation has six degrees of freedom (three translations, three rotations), and an estimate can be based on three point correspondences (which is achieved by two edge correspondences). However, in this case there is a constraint: the observations are of the same stable position, and the transforma-

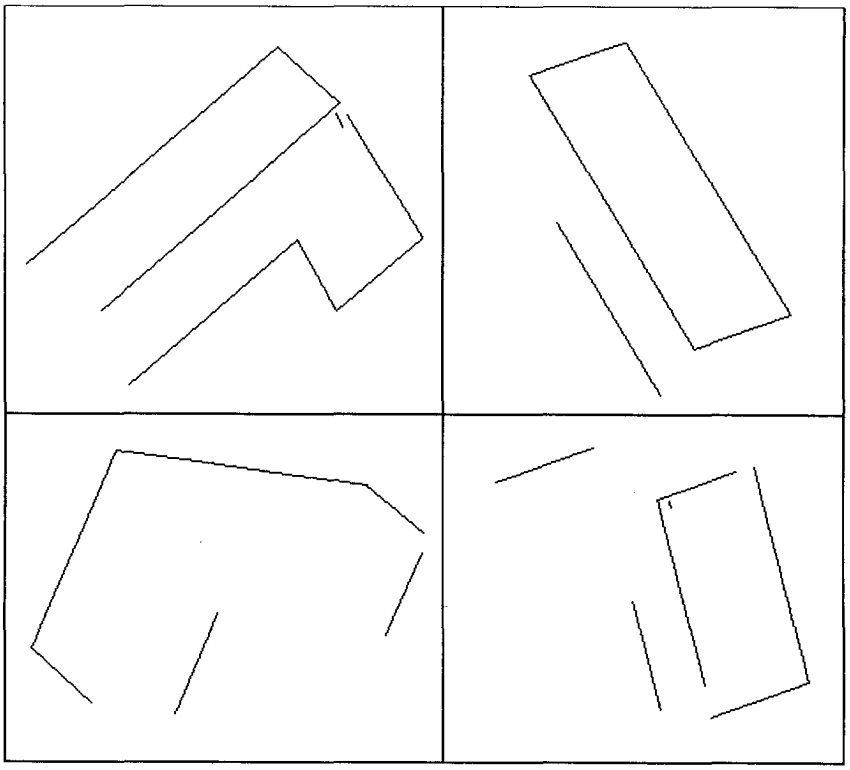


Fig. 7-5 Different observations of another stable position of the L-object.

tion between them is only a rotation around the z-axis, followed by a translation in the x-y plane. As a result of this, there are only three degrees of freedom and the transformation can be estimated using only two point correspondences, or a single edge correspondence. (Use of the z-axis and x-y plane assumes a perfectly horizontal table. However, as long as the table position and orientation are known, there are only three degrees of freedom).

The latter transformation can be estimated using a modified version of the method described by Faugeras and Hebert [29] (which is also used elsewhere in this thesis, see 5.4). This method minimizes the sum of the squared distances between corresponding points in the two observations.The estimated transformation consists in general of a rotation around an axis through the origin followed by a translation. In this method the rotation is described by a quaternion $q = (w, s)$, where

$$w = \sin(\frac{\theta}{2})\, v, \tag{88}$$

$$s = \cos(\frac{\theta}{2}) \tag{89}$$

v being the direction vector of the rotation axis, and θ the angle of rotation. The result of the method are a rotation and a translation which minimize

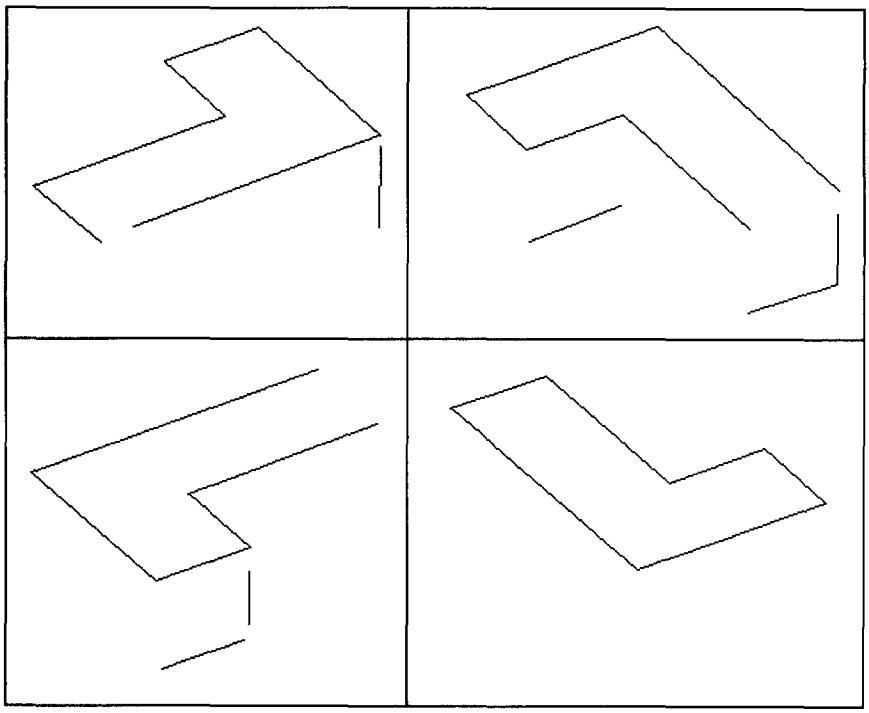


Fig. 7-6 Different observations of the third stable position of the L-object.

$$\sum_{i=1}^{n} |qx_i\bar{q} + t - x'_i| \tag{90}$$

where x_i and x'_i are corresponding points from the two observations.

For estimating a horizontal transformation (between two observations of the same stable position), a modified version of the method was used. This is obtained by substituting the vertical axis $[\,0,0,1]^T$ for v in (88), so that θ is the only unknown variable in the rotation. Similarly, the z-component of the translation is put to 0. This allows the method to be simplified[42]. The resulting method reliably estimates the three parameters.

7.4.3 Generating hypotheses

In case a horizontal transformation has to be estimated between two observed graphs, it is sufficient to find edge pairs, consisting of an edge in one graph and a corresponding edge in the other graph.The hypothesis generation step involves finding all possible edge pairs, and estimating the transformation associated with each. If the transformation differs too much from either a horizontal translation or a rotation around the z-axis, edge pairs are rejected. The transformation is derived from the pairs of corresponding endpoints of the edges using the method indicated in the previous section.

7.4.4 Testing hypotheses

To test the quality of an hypothesis, the transformed observation and the other observation are compared. The assumption made is that if two observed edges (one in each graph) correspond, they were both generated by a real edge in the object. For every edge pair an estimate of the real object edge is calculated (see Fig. 7-7). The estimate could be interpreted to lie on a line between the average end points, and has a length corresponding to the part that the two edges have in common.

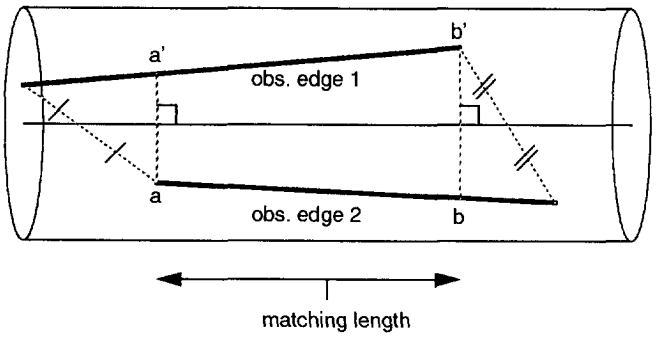


Fig. 7-7 Observation-observation matching using an estimate of the real object edge of two observed edges.

If the edges are within a certain distance of the estimated object edge, the matching length is calculated as shown in Fig. 7-7. The sum of the matching lengths of all matching edge pairs for a certain hypothesis is used as the score for that hypothesis. Selecting the best hypothesis then amounts to picking the one with the highest score.

7.4.5 Optimizing a hypothesis

Each original hypothesis was based on only one edge correspondence. However, during hypothesis testing many more edge correspondences may be found. These can be used to re-estimate the transformation between the graphs, leading to a more accurate estimate. This requires corresponding points in the graphs to be indicated. For this the pairs of points (a,a') and (b,b') as indicated in Fig. 7-7 are used. These are already determined during the test of the hypothesis, when the matching length is calculated. The contribution of these points is weighted by the matching length of the connected edge to give a more accurate result.

7.4.6 Merging the Observations into one Graph

Using the estimated transformations between the observations, all observations are transformed back to one position. All edges in the observations which result from the same object edge are then in approximately the same position and orientation. Two approaches have been tested to merge the set of positioned observation graphs into one graph. The first one is based on clustering the endpoints of the edges, the second one on clustering the edges themselves.

The first approach (based of the clustering of the endpoints) is fairly straightforward. Points

within a certain distance of each other are merged, until this is no longer possible. The results for the three stable positions of the L-object are shown in Fig. 7-8. The point clusters are indicated by numbers. The method immediately results in connected graphs. Although this approach works, it seems to rely on the endpoints of edges, which are less reliable than the edges themselves in the stereo vision process. Therefore another approach was developed.

The second approach (edge-based clustering) is based on the fact that the stereo vision system detects edges. In the process, errors occur as observed in 7.4.1. As a result of this, an object edge will show up in the set of observations as a cloud of edges, some of which only represent part of the object edge. The learning process should try to find an "average" edge in the cloud. where with the following properties:

- The distance of the observed edges to the average edge, perpendicular to it, should be small.
- The average edge should have its end points at the extreme ends of the cloud.

Of course, it is not known in advance which observed edges belong to the same model edge. Therefore, the clustering procedure should try to find clusters and average edges at the same time.

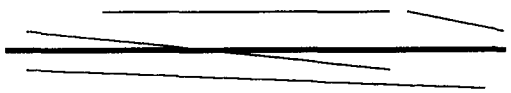


Fig. 7-9 An object edge (fat) will show up in the set of observed edges as a cloud of edges. These will have a small distance perpendicular to the observed edge due to noise in the observations. However, they may not represent the full length of the model edge because part of the edge may not be detected. The notion of an "average edge" is based on this.

The average of a set of edges is found using eigenvector analysis. All points of all k edges connecting points $\underline{a}_k$ and $\underline{b}_k$ in the set of edges are used to construct a matrix R, the covariance of all the points $\underline{x}$ in the cluster. This matrix can be found by summing over all edges and integrating over all points on each edge:

$$\begin{aligned} R = & \quad E\{(\underline{x}-\underline{\mu})(\underline{x}-\underline{\mu})^T\} \\ & = \frac{1}{l}\sum_{1}^{N}\int_{\underline{a}_k}^{\underline{b}_k}(\underline{x}-\underline{\mu})(\underline{x}-\underline{\mu})^T d\underline{x} \end{aligned} \tag{91}$$

where l is the total edge length,

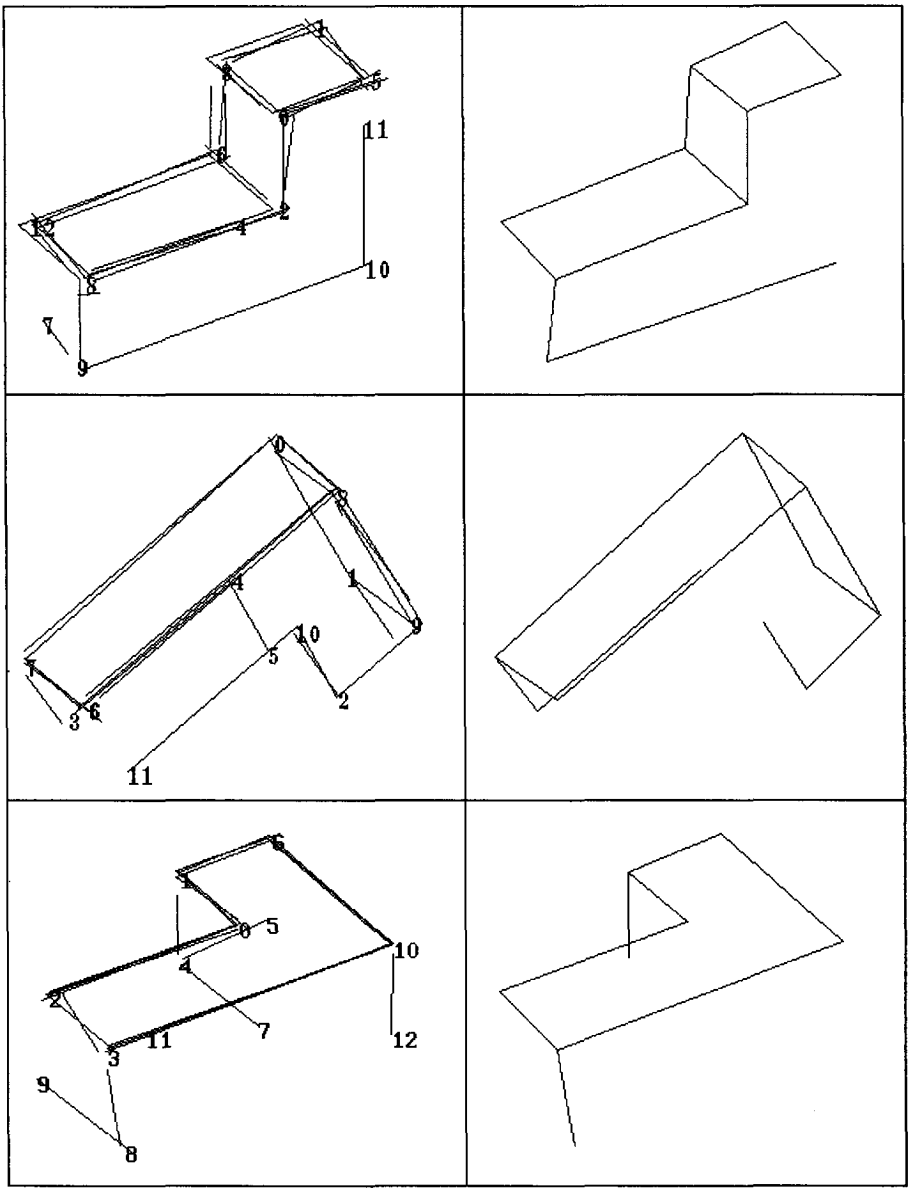


Fig. 7-8 Creation of the stable position graphs using the point-based clustering method, for the three stable positions of the L-object.

$$l = \sum_{1}^{N} \|\underline{a}_k - \underline{b}_k\| \tag{92}$$

and μ is the mean of the cluster

$$\underline{\mu} = \frac{1}{l} \sum_{k=1}^{N} \|\underline{a}_k - \underline{b}_k\| \frac{1}{2} (\underline{a}_k + \underline{b}_k) \tag{93}$$

The integrals over edges which occur in the construction of the covariance matrix can be eliminated. The eigenvector belonging to the largest eigenvalue of R is used as the direction of the average line through the mean of the set of edges. By doing so the sum of the squared distances of all points of all edges to this line is minimised. The average edge is the smallest edge containing all the projections of the edges in the set on the average line.

Using this average edge of a cluster, two constraints can be given to indicate whether a cluster is good:

- The edges are close to the average edge, in a direction perpendicular to it.
- The gaps within the cluster are small in the direction of the average edge.

The implications of these two constraints are shown in Fig. 7-10. Using the average edge and the two constraints, the edges can be clustered as follows:

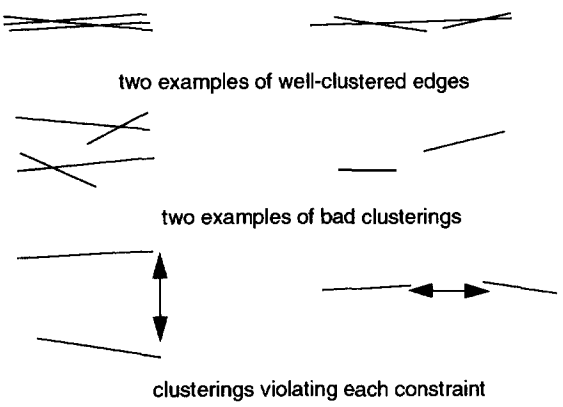


Fig. 7-10 Examples of good clustering, bad clustering and illegal clustering of edges.

The purpose of the clustering procedure is to find the groups of edges which result from the same object edge. For this an agglomerative hierarchical clustering method (see Jain and Dubes [40]) is used. The distance between two clusters is expressed in the quality of the merged

clusters, which is the weighted sum of two parts. The first part is the largest distance of the endpoints of the edges in the merged cluster to the average edge of the merged cluster. This is the radius of the cluster. The other part is the length of the average edge divided by the total length of all the edges in the merged cluster, the reciprocal density of the cluster. Furthermore two constraints are used during the clustering process. The first one constrains the radius of the cluster to a maximum radius which is determined by the errors introduced by the stereovision system. Secondly the clusters should not contain a significant gap in the direction of the average edge. The size of the allowed gap is object dependent. The two constraints serve as an end criterion, and force the clustering process to the desired result.

The result of this edge-based clustering algorithm is shown in Fig. 7-11. Again, numbers are used to indicate the clusters found. The resulting average edges are shown on the right. It can be seen that the procedure does not yield a connected graph. In fact, because the average edge has its end points at the extreme ends of a cluster rather than at some "mean" end point, the end points of each edge are somewhat outside of the object.

The final step in the merging procedure is the restoration of the connectivity between the edges in the new graph, in case the edge-based clustering method is used. For each couple of edges, the shortest line connecting them is calculated. If this line is shorter than a threshold, the two edges are connected at a point in the middle of the connecting line. This method is insensitive to the fact that the end point is on the outside of the object, as can be seen in Fig. 7-12. The full procedure is described in [42].

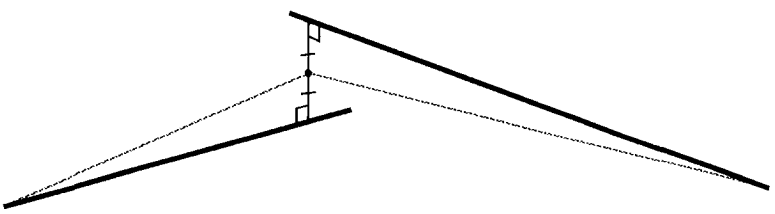


Fig. 7-12 Restoration of the connectivity in a merged graph. The new end point lies in the middle of the shortest line connecting the two edges. The grey lines indicate the new positions of the edges.

The result of this step can be seen in Fig. 7-13. It is at this point possible to remove clusters smaller than a certain size, as shown in this figure, in order to remove false detections. However, with small sets of observations (as in this example), this also removes a number of correct edges. Therefore, this has not been done in later experiments.

The above discussion suggests that edge-based clustering has advantages over point-based clustering. However, the lack of speed of the implementation (see section 7.6.1) made a large-

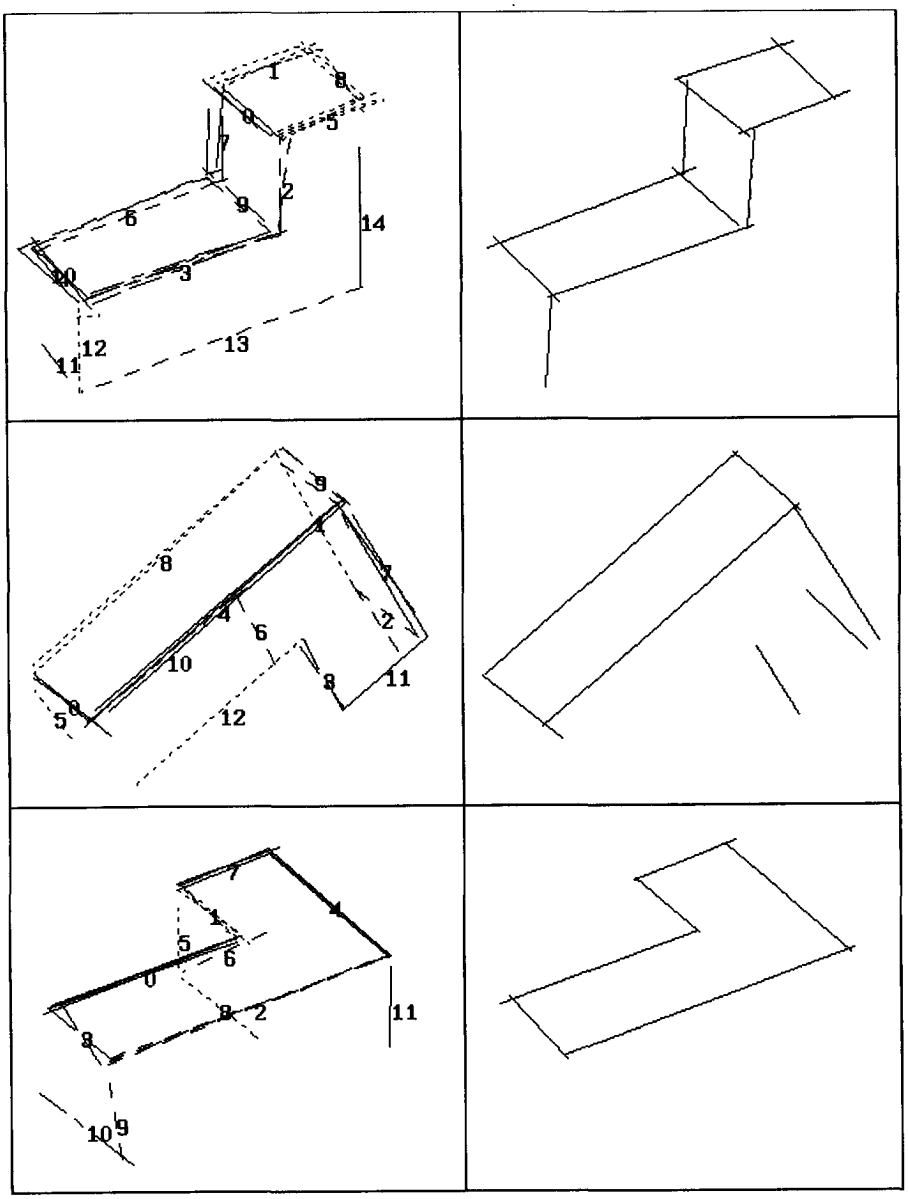


Fig. 7-11 Creation of stable position graphs using the line-based clustering method, before endpoint connection, for the three different stable positions of the L-object.

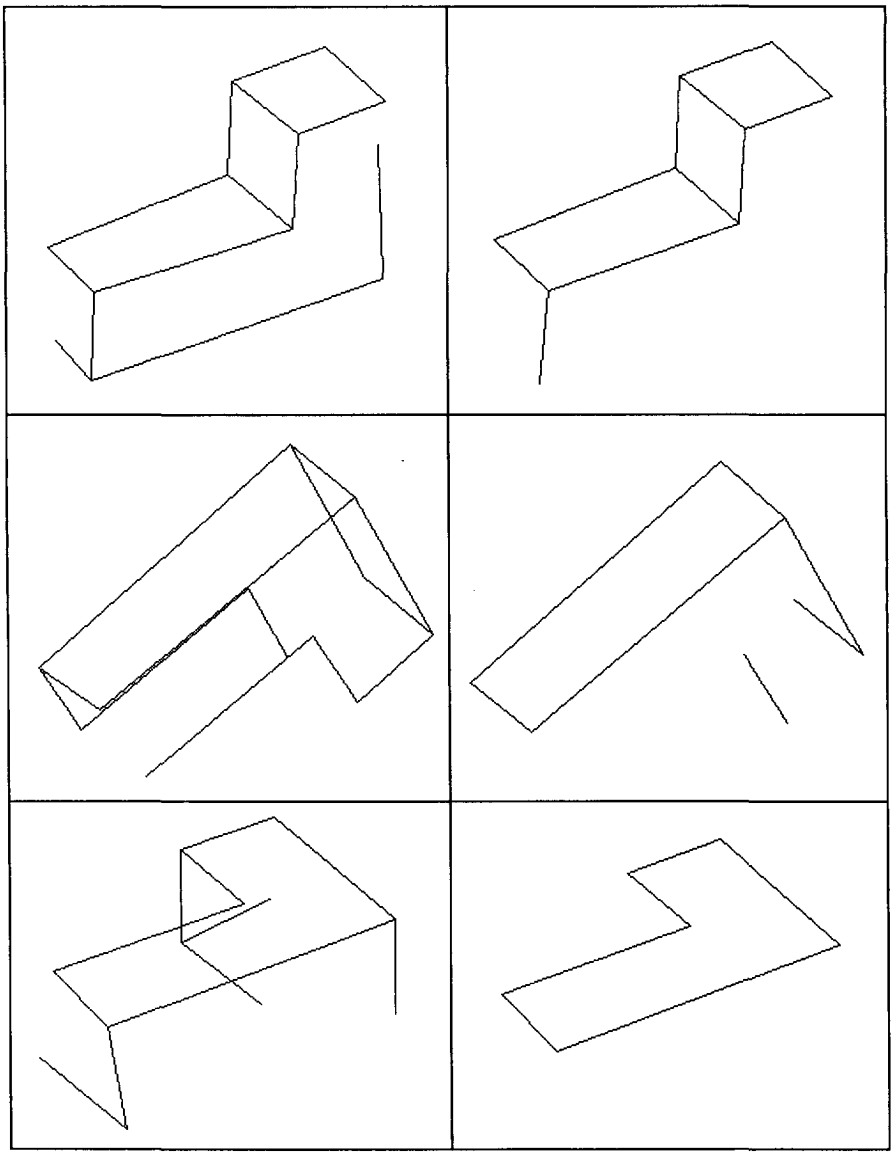


Fig. 7-13 Stable position graphs for the three stable postions of the L-object using the line-based clustering method, after endpoint connection. In the left column, the graphs are shown with single occurrences included, in the right column the same graphs after single occurrences have been removed.

scale experiment infeasible. Therefore, this issue has not been fully investigated.

7.5 Learning a full model

7.5.1 Hypothesis generation

A full model has to be generated out of the graphs for individual stable positions described in the previous section. Merging these requires hypotheses to be generated about their relations. The transformation between two stable positions is a general 3-dimensional transformation, that can be estimated using the method described in 5.4. An estimate of such a transformation is based on at least three pairs of corresponding points. For this the four pairs of end points of two edge pairs can be used. An edge pair is defined as a triple *(A,B,P)*, where *A* is an edge from one observation, *B* is an edge from the other observation, and *P* is the parity of the edge pair indicating which endpoints correspond to each other. All combinations of edge pairs containing edges of similar length can be used for a hypothesis.

7.5.2 Hypothesis testing

Testing these hypotheses can be done by transforming one of the observations, using the transformation estimated. Then, for all pairs of edges an estimate of the actual object edge is again calculated. It can be decided, based on this estimate, if the two edges can be accepted as corresponding to the same object edge. The sum of the lengths of all matching edges is again used as the score for the hypothesis, and the hypothesis with the highest score is accepted.

7.5.3 Merging of stable positions (clustering)

If best hypotheses can be found for all stable position graphs, a clustering procedure can once again be applied to all. Both the point-based and the edge-based methods can again be applied. In this case, only the edge-based method has been used. The result of applying this method to the three stable positions of the L-object is shown in Fig. 7-14. Also shown there is the final result, after restoring the connectivity. In this case, all the edges in the object can be found in the model, although small errors in the point coordinates occur.

7.6 Experiments

7.6.1 Implementation

The algorithms described in the preceding chapters have been implemented using the Poplog package, using the languages Prolog and Pop-11, and some external procedures in C. The search algorithms were implemented in Prolog, because this language lends itself very well for that. Pop-11 is the general purpose language in the Poplog package, that offers the similar functionality to most popular languages. It also interfaces well to Prolog and C. The whole system was running on a Sun 3/60. A consequence of the use of a slow language (Prolog) and a slow computer was that the whole system ran very slowly. Typically, learning of the models shown in this chapter required several hours. It could be expected though, that the use of a modern computer and reimplementation of the system would each increase the speed by an order of magnitude.

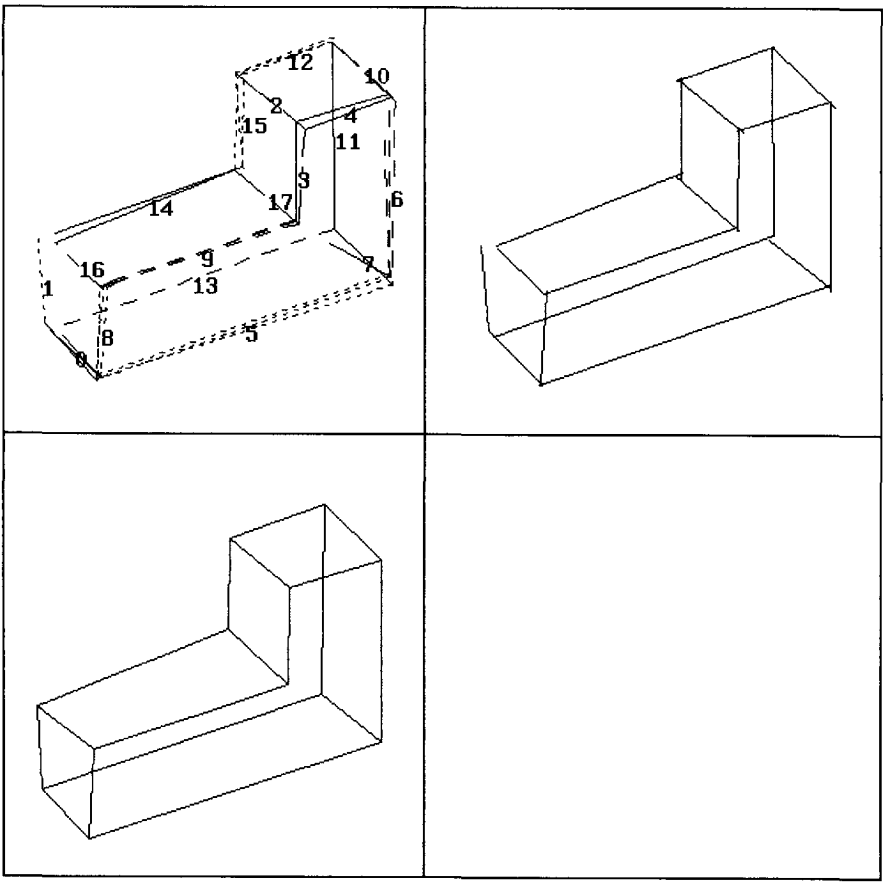


Fig. 7-14 The result of line-based clustering of the L-object, for all stable positions together. Shown are: the clusters found, the resulting edges and the result after restoring the connectivity.

A number of objects have been submitted to the learning system described in this chapter. First experiments showed that it was impossible to learn the objects that were present in the DIAC database. Only objects with a much coarser structure could be learned. This is due to the limited resources available to the program, both CPU time and memory, and the limited resolution of the stereo vision system. Using these resources, only a few observations could be merged, requiring every observation to be a good one and every cluster to be distinct from others. If more resources were available, more observations could be used and the clusters could be based more on statistics. If the resolution of the stereo vision system were higher, clustering could be better based on small numbers of edges. However as a result of these limitations, the objects used were not the Diac parts defined elsewhere in this thesis.

Instead, five parts of a Soma-puzzle (attributed to the Danish mathematician Piet Hein[26]) were used, the L-object that was introduced earlier, and four other parts shown in Fig. 7-15. The puzzle itself consists of nine parts, that can be assembled into a cube. Solving the puzzle was not part of the experiment.

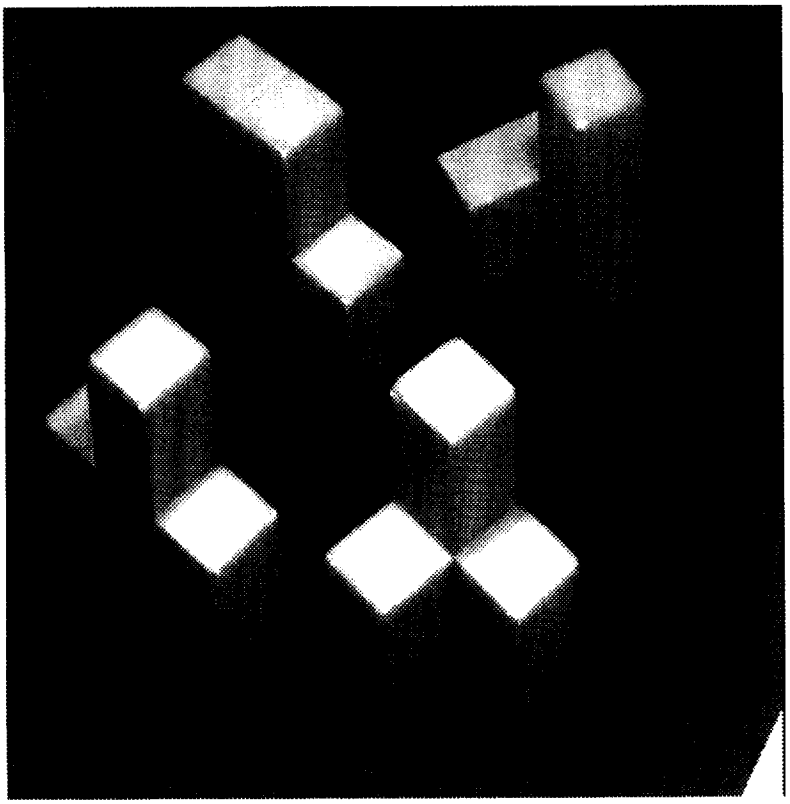


Fig. 7-15 The remaining objects in this experiment. From left to right: T, Z, X, q.

7.6.2 Learning models

Applying the learning procedure to all five objects was not completely successful: the system was not able to find a correct model for the X-object. The reason for this was the poor quality of the observations of this object. Increasing the number of observations would probably have solved this problem, but was computationally not feasible. Edge based clustering was used. The model obtained for the L-object is shown in Fig. 7-14 and those for the remaining three objects are shown in Fig. 7-16. Note that the Z-object has been found completely, with one

false edge, the T-object has one small side missing, and the q-object has only been found partially although with a shadow included (lower right hand corner). Nevertheless, all four models were used in the rest of the experiment.

7.6.3 Applying the models

After learning the models, a new set of 40 observations was obtained, ten of each object that was successfully learned. The recognition system described in chapter 5 was applied to all observations, using the learned models. The results are shown in Fig. 7-17, and a confusion matrix summarising the result in Table 7-1

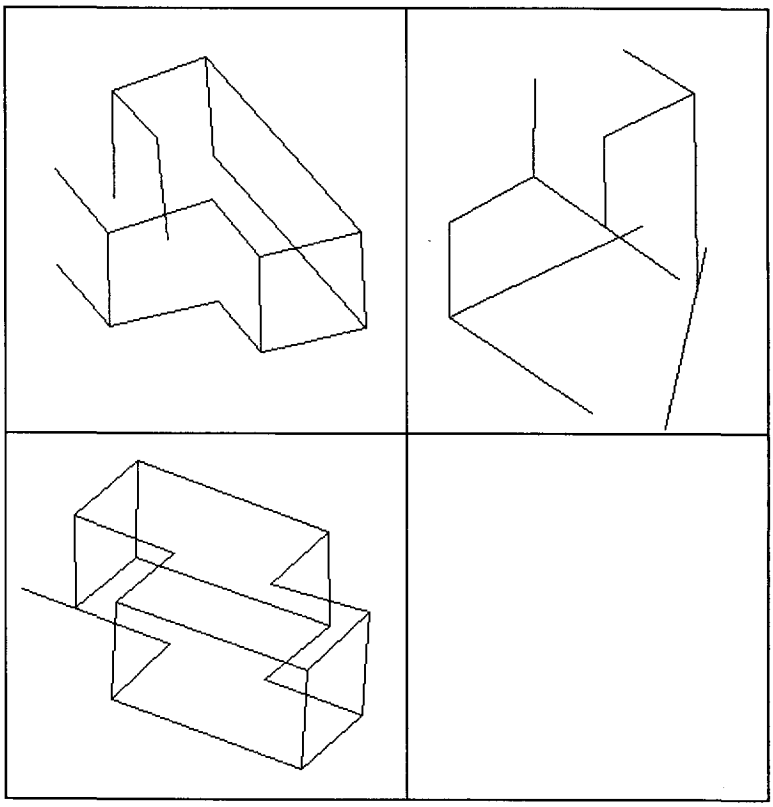


Fig. 7-16 The models found for the three other objects in the experiment for which the procedure was succesful. The T-object (top left), the q-object (top-right) and the Z-object (bottom left) are shown. For the X-object, no model could be obtained during the experiment.

From the table it may be clear that the models that are more complete are also the more successful models. The Z-object was classified correctly all of the time, while the only case where the L-object was misclassified (nr. 7), only one side of the object was clearly visible. As the T-object has an identical side, confusion is easy. The T-object, where one side is missing

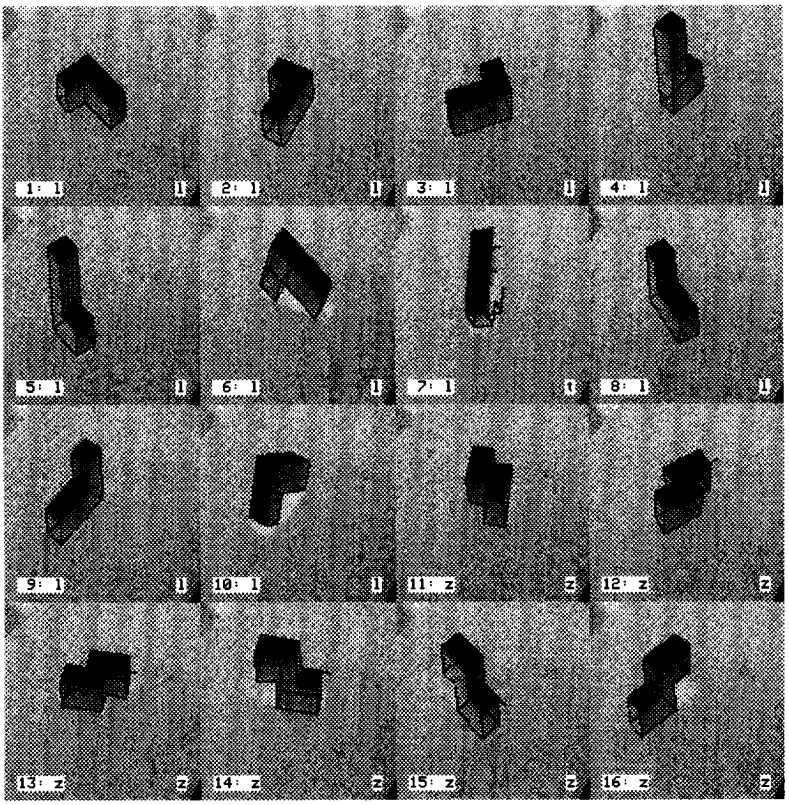


Fig. 7-17 Recognition experiment for models learned with the algorithm described in this chapter. Shown are 40 images, 10 per class, with the model recognised overlayed in black lines. The original images are shown invertedly for clarity. In the lower left hand corner of each image its sequence number and the object depicted (l,z,t or q) is shown. In the lower right hand corner the model identified is shown.

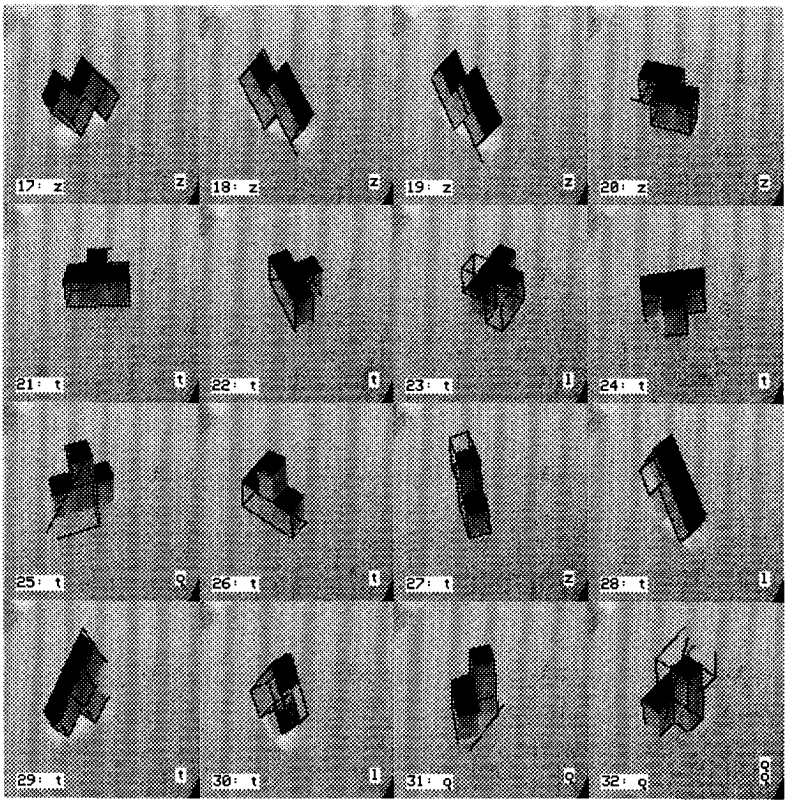


Fig. 7-17 continued. Note that in image 32, the q object has actually been found twice.

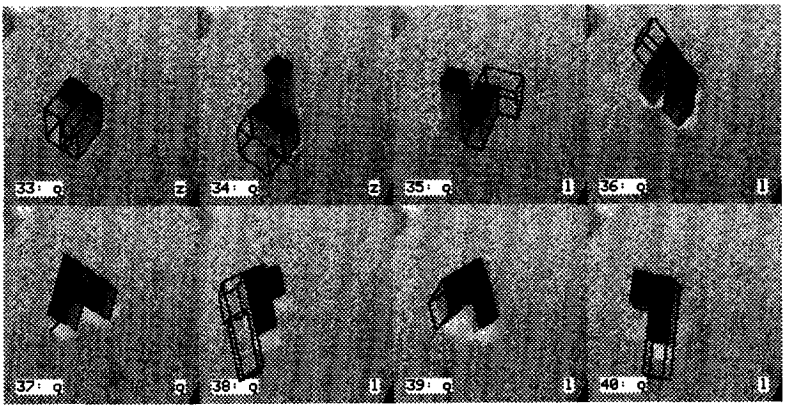


Fig. 7-17 continued.

Table 7-1 Confusion matrix using learned models.

object\ class	L	Z	T	q
L	9	0	1	0
Z	0	10	0	0
T	3	1	5	1
q	5	2	0	3

completely in the model, was only classified correctly in half the cases. Most often it was confused with the L-object, with which it has the largest side in common. The model for the q-object was obviously very bad: only three times out of ten was the q-object classified correctly.

In this experiment, the reject threshold was very low. Because all the objects consist of edges of only three different lengths, bad classifications could have scores that were in the same ranges as good classifications. As a result of this, the threshold was not very effective.

7.6.4 Evaluation of the models

To illustrate that the quality of the learning system was tested here, the same 40 observations were again classified, this time using operator-supplied models. These were obtained by measuring the objects by hand. The results of these classifications are shown in Fig. 7-18 and summarised in Table 7-2

Table 7-2 Confusion matrix for operator-supplied models

Object\class	L	Z	T	q
L	10	0	0	0
Z	0	10	0	0
T	0	0	10	0
q	0	0	0	10

From this table it is clear that all objects were classified correctly. It should be noted though, that nr. 7 (the L-object that was classified as a T) is classified correctly this time, but only because the score for L is less then 1% higher than that for T. In fact, a minor change in the parameters causes it to be classified as T. In this case, the observation is so difficult that the success of the recognition system is unpredictable. The other 39 cases though, were easily classified.

This near misclassification suggests that a second type of reject could be useful in the recognition system: one where a best hypothesis is also rejected if it is not significantly better than the second best. However, this was never implemented.

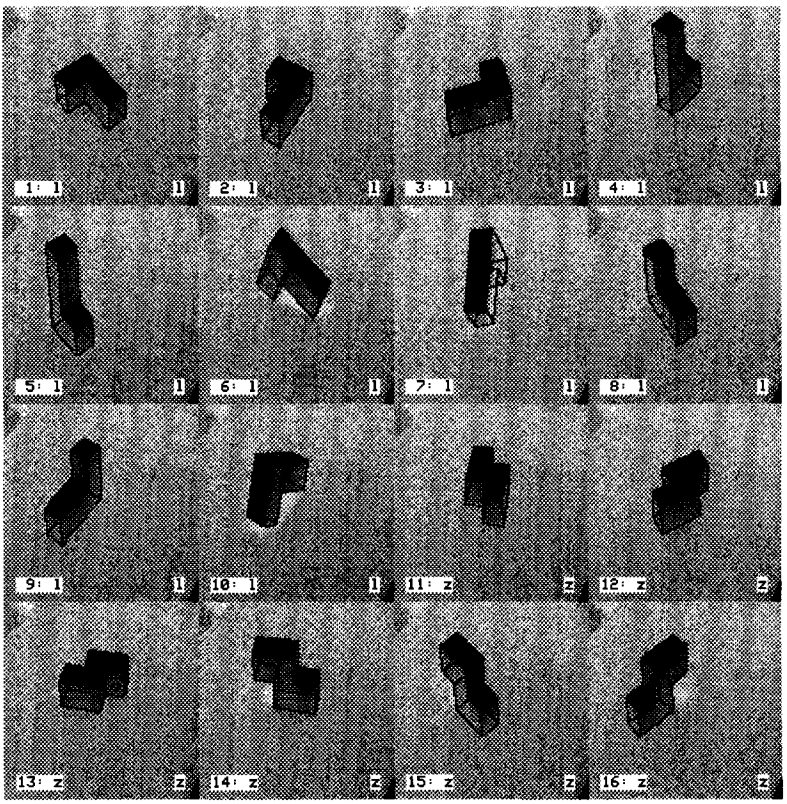


Fig. 7-18 The recognition experiment with manually generated models. All objects are recognised correctly. Shown are the same 40 images, 10 per class, again invertedly and with the recognised model overlayed in black. In the lower left hand corner of each image its sequence number and the object depicted (L, Z, T or q) is shown. In the lower right hand corner the model identified is shown.

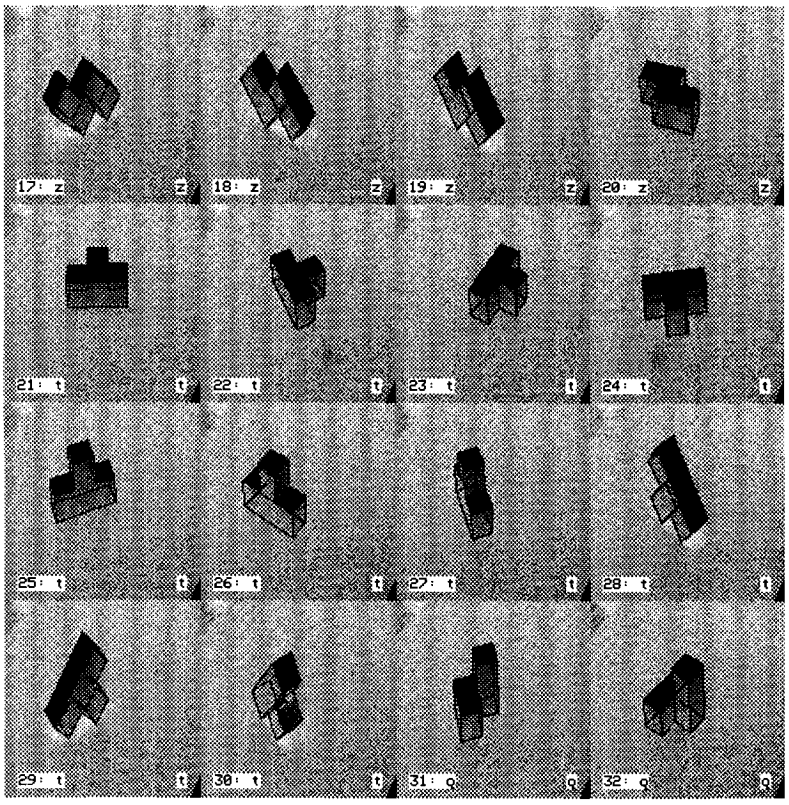


Fig. 7-18 continued.

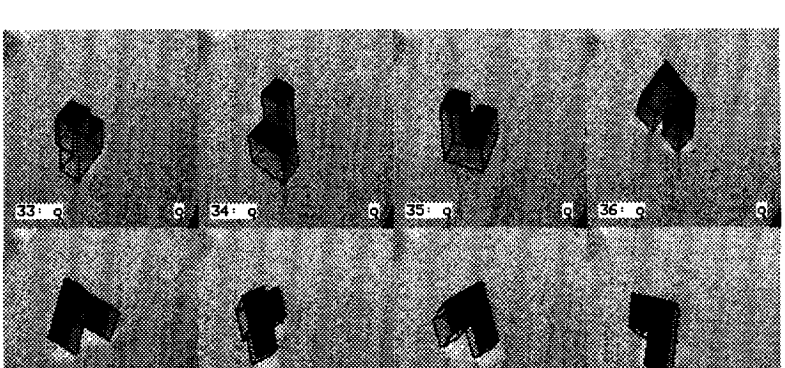


Fig. 7-18 continued.

7.6.5 Estimating the vision system's precision.

In another experiment, the precision of the stereo vision system was estimated in a way similar to that in section 4.6. Of two different stable positions of the L-object, six and eight observations respectively were obtained. These were subsequently clustered using the edge-based clustering method, and the average edges in each cluster were taken as estimates of object edges. Fig. 7-19 shows histograms of the minimum and maximum distances of all edges to the average edge in each cluster. As could be expected for most common distributions, the maximum value does not lie at radius 0. After all, these are distances perpendicular to lines, so they are the square root of the sum of two squared stochastic variables. If these were for instance normally distributed and independent, the resulting distribution would be χ^2_2.

These histograms suggest that 95% of all edges in this object can be observed within a minimum (maximum) distance of 3.2 mm (5.6 mm) of their actual location. The latter figure is slightly higher than the precision of 1 mm found in section 4.6, which could be explained by the fact that the experiment described there took place with a later version of the stereo vision system.

If such a procedure could be applied to all objects for which the vision system is to be used, and for all stable positions, definitive statistics could be obtained about the stereo vision system for this application in a semi-automatic way: the only requirement would be that an operator would go through the whole learning procedure. However, a representative subset of all possibilities would be sufficient. An important condition would be that the circumstances at the time of learning and at the time of actual use of the stereo vision system would be identical.

The performance of the current implementation restricted the experiment to the size indicated above.

7.7 Conclusions

In this chapter, the origin of the models used for recognition has been addressed. Models could originate from two sources: a model database, possibly derived from a CAD procedure, or a learning procedure. For the DIAC problem, the first approach is used. An interface has been defined between the Product Data Model and the recognition system. This allows automatic generation of models when new parts are introduced in the system.

The second approach, learning the models, is a more complicated one. A procedure has been described for learning models using the stereo vision system with only a few instructions for the operator. The learning algorithm uses a set of observations from a few stable positions of the object, first to create a model of each stable position and then to derive a model of the object as a whole. The algorithm uses hypothesis generation and testing procedures similar to the recognition system described in chapter 5.

A learning system, that was actually built, was able to acquire simple models that could be used for recognition. However, the implementation limited its use to very simple problems. In particular, the number of observations allowed was rather limited. In cases where the small

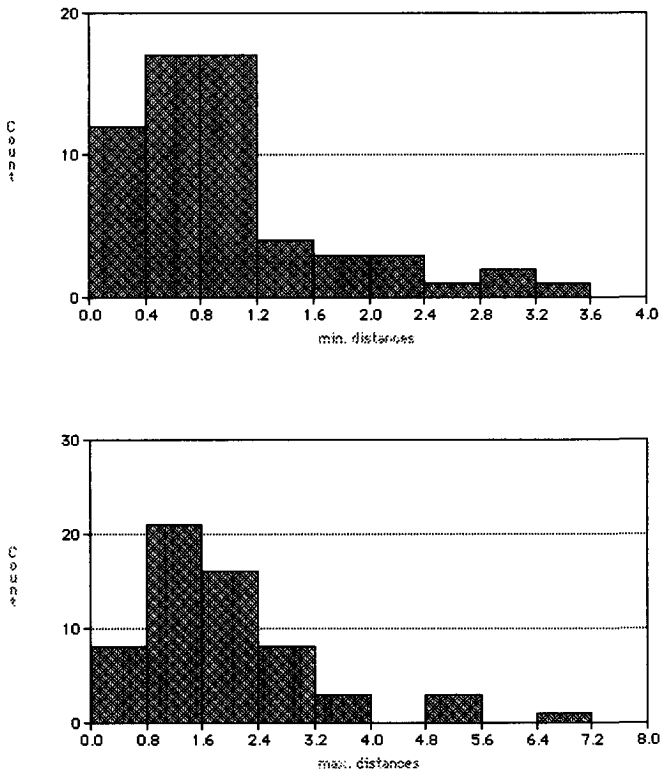


Fig. 7-19 Distribution of the minimum and maximum distance (in mm) of edges to the resulting edge of each cluster. The total number of edges is 60.

number of observations lead to a poor model, the results in recognition using the model were also poor. In one case, no model could be found at all.

Another application of the learning procedure is the semi-automatic estimation of the precision of the stereo-vision system. A number of observations of a set of test objects can be obtained, and the learning procedure can automatically find clusters of corresponding edges and an average edge for each cluster. The deviations of the edges from the average edge can be used to estimate the precision of the system. Such a procedure for an early version of the DIAC stereo vision system showed results slightly worse than the ones described elsewhere in this thesis.

Reimplementation of the learning system using faster technology would make it feasible to apply it to larger learning sets, thus enabling the learning of more complex objects. Further-

more, it could allow the inclusion of other primitives than straight lines, such as the circular arcs already provided by the stereo vision system. Also, better statistics about the stereo vision system could be obtained (for instance, precision as a function of size).

8 Conclusions

This chapter summarises earlier conclusions, and relates them to the goals outlined in the introduction.

8.1 Conclusions of earlier chapters.

- In chapter 3, a two stage system for object identification in DIAC was proposed. The first stage (the verification system) tries to verify a hypothesis based on outside information and common object positions. In case of insufficient result, the more general recognition system is activated. Criteria have been found for switching from verification to recognition. Minimization of an overall cost function leads to optimal thresholds for both verification and recognition.

The cooperation between the verification system and the recognition system as described in this chapter has not been fully tested. However, its use was illustrated by an experimental example which showed the feasibility of the concept. Using this two stage approach, it will be possible to build a system that is capable of a fast performance (using verification), yet that is sufficiently powerful to deal with a number of unexpected cases (using recognition).

- In chapter 4, a stereo vision system has been described using straight lines and ellipses in images, representing straight lines and circles in 3-d. First, images are acquired and edge detection is performed. Then, two different stereo vision algorithms are invoked.

The straight line based stereo vision algorithm is straightforward, mainly caused by the use of strong constraints. A straight line fit is applied to edges and lines are checked for correspondence. The best correspondence for each is kept, if it is better than a certain threshold. Although the system is capable of finding well defined straight lines mostly with an accuracy of 2-3 mm, there are some outliers at certain orientations.

Ellipse based stereo vision is based on an algorithm to find ellipses or partial ellipses in gray value images, and an algorithm to find the circle in 3-d corresponding to two ellipses. The algorithms appear to be sufficiently fast and accurate for the detection of clear circles, although the practical application of the system in chapter 6 suggested that the ellipse detection stage is not yet robust enough for the recognition of cylindrical industrial parts.

In chapter 5, the recognition system was described qualitatively. It is based on hypothesis verification and testing, using careful pruning to reduce the computational complexity. Rotational symmetries, included in each model, are eliminated at an early stage of recognition. Knowledge about the rigidity of the object and the supporting surface is also exploited. These result in significant speedups of the system.

In chapter 6, the performance of the system as a whole was evaluated. It was concluded that the overall recognition accuracy was still insufficient for immediate use. However, some of the weak points could be identified (insufficient resolution for small objects, lack of robust-

ness of the ellipse detection algorithm, impossibility to identify all objects from only two views) and these can be improved upon.

In chapter 7, a description was given of two ways of obtaining the models for the recognition system. These show one of the advantages of a flexible vision system: either the models can be generated automatically from a CAD database of the parts to be recognised, or the models could be learned by having an operator place the object in front of the cameras using a simple procedure that does not require accurate positioning. The latter procedure however is only suitable for models that are of a much coarser structure than those that can be obtained from a CAD database.

8.2 Methodology

The stereo vision system reflects what in the introduction was called the engineering approach to vision: stereo vision is done at a level that offers a large decrease in complexity (compared to conventional stereo at the pixel level), by using adequate primitives for the recognition system. As a result of this, an efficient vision system was realised without reference to a specific perception model.

Some effort was put into realising a reproducible and comparable evaluation. As no standard dataset was available, the dataset that was used to evaluate the system is still available along with the relevant parameters. Researchers working on similar problems are invited to use the data set for a comparable evaluation of their own systems.

8.3 Recommendations for further research

Further research is recommended on the following topics:

- As indicated in section 6.3.2, the system still has a few weak spots that should be improved upon, in particular, the robustness of the ellipse detection algorithm. Although the ellipse-based stereo vision system was shown to work for images of clear circles, it proved to be insufficiently robust for the detection of the circles on cylindrical parts. As more robust algorithms for ellipse detection exist (see section 4.3.3), it should be investigated whether these would improve the performance.
- Once the implementation of the verification system has been completed, the behaviour of the combined systems should be evaluated in a real environment.
- A comparison to similar systems should be performed. A complete dataset of 200 image pairs is available for evaluation of similar systems developed by other researchers. It is suggested to apply other vision systems to the same dataset in order to evaluate the differences in performance.

References

[1]C.S. Andersen, J.J. Sørensen, H.I. Christensen, *An analysis of three depth recovery techniques*, proceedings 6th SCIA, Aalborg 1991.

[2]K.S. Arun, T.S. Huang, S.D. Blostein, *Least-Squares Fitting of Two 3-D Point Sets*, IEEE PAMI, vol. 9, no. 5, pp. 698-700, September 1987.

[3]N. Ayache, O. D. Faugeras, *Building, registrating, and Fusing Noisy Visual Maps*, Proceedings 1st ICCV, London, 1987.

[4]N. Ayache, B. Faverjon, *Efficient registration of stereo images by matching graph descriptions of edge segments*, International Journal of Computer Vision, vol. 1, 1987, 107-131.

[5]J.P. Baartman, C.J.M. Heemskerk, *On Process Planning with Spatial Uncertainties in Assembly Environments*, CIRP Proceedings - Manufacturing Systems, Vol 20 1991, ed. J. Peklenik.

[6]E. Backer, R.P.W. Duin, *Statistische patroonherkenning*, D.U.M., Delft 1990.

[7]E. Backer, J.J. Gerbrands, *Inexact graph matching used in machine vision;* in: P.A. Devijver, J. Kittler, eds., Pattern recognition theory and applications, NATO ASI Series, vol. F30, Springer-Verlag, Berlin Heidelberg, 1987.

[8]D. H. Ballard, C. M. Brown, *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[9]M. Bart, J. Buurman, R.P.W. Duin, *A learning procedure for the recognition of 3D objects from 2D images*, SPIE conference on Intelligent Robots and Computer Vision IX: Algorithms and Techniques (Vol. 1381), Boston 1991.

[10]B. Bhanu, *CAD-based robot vision*, IEEE Computer, August 1987, pp. 13-16.

[11]D. Bierhuizen, *Aspects of object verification*, Ph. D. thesis, Delft University of Technology; Delft University Press, 1993

[12]D.J.W. Bierhuizen, J.J. Gerbrands, E.Backer, *An object identification system based on multiple image analysis and task dependent operation*, Proceedings of the international conference, Florence, Italy, September,1991, Elsevier Science Publishers.

[13]K. Bowyer, ed., Proceedings IEEE workshop on Directions in Automated CAD-based vision, Maui, Hawaii, June 1991.

[14]K.L. Boyer, A.C. Kak, *Structural stereopsis for 3-D vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 10, 1988, 144-166.

[15]J.P. Brady, N. Nandhakumar, J.K. Aggarwal, *Recent Progress in the recognition of Objects from Range Data*; Proc.of the 9th ICPR, Rome, 1988.

[16]H. Bunke, Allerman, *Inexact graph matching for structural pattern recognition*, Pattern Recognition Letters 1, 1983.

[17]J. Buurman, *The Diac Object Recognition System*, SPIE conference on Applications of Artificial Intelligence X: Machine Vision and Robotics (Vol. 1708), Orlando, USA, April 1992.

[18]J. Buurman, *Ellipse Based Stereo Vision*, Proceedings ECCV' 92, Santa Maguerita Ligure, May 1992.

[19]J. Buurman, D.J.W. Bierhuizen, *A Two Stage Object Identification System in the Delft Intelligent Assembly Cell*, Proceedings of the SPIE Symposium on Advances in Intelligent Systems (Vol. 1386), Boston, USA , November 1990.

[20]J. Buurman, R.P.W. Duin, *Object Recognition Using Inexact Matching of 3d Graphs*, in: V. Cantoni, L.P. Cordella, S. Levialdi, G. Sanniti di Baja (eds.), Progress in Image Analysis and Processing, World Scientific, Singapore, 1990, pp. 415-419.

[21]J. Buurman, R.P.W. Duin, *Comparing Distance Measures for Object Recognition using Minimal View Sets*, Proceedings 7th Scandinavian Conference on Image Analysis, Aalborg, August 1991.

[22]J. Canny, *A computational approach to edge detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, 1986, 679-698.

[23].D.T. Clemens, D.W. Jacobs,*Model group indexing for recognition*, SPIE conference on Intelligent Robots and Computer Vision IX: Algorithms and Techniques (Vol. 1381), Boston 1991.

[24]M. Cruz Picao, Internal Report Information Theory Group, Delft University of Technology, 1991.

[25]E.R. Davies, *Finding ellipses using the generalised Hough transform*, Pattern Recognition Letters, vol. 9, 1989, 87-96.

[26]P. van Delft, J. Botermans, *Spelen met Puzzels.*, De Bezige Bij, Amsterdam, 1978 (in Dutch).

[27]U.R. Dhond, J.K. Aggarwal, *Structure from stereo-a review*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, 1989, 1489-1510.

[28]L. Dorst, *Discrete straight line segments, parameters, primitives and properties*, Ph. D. thesis, Delft University of Technology, 1986.

[29]O.D. Faugeras., M. Hebert, *The Representation, Recognition, and Positioning of 3-D Shapes from Range Data*, in Techniques for 3-D Machine Perception, ed. A. Rosenfeld, pp.13-51, North-Holland, Amsterdam 1986.

[30]N.J. Foster, A.C. Sanderson, *Determining Object Orientation Using Ellipse Fitting*, SPIE Intelligent Robots and Computer Vision, vol. 521, pp. 34-43, 1984.

[31]R.J. Fryer, *On the Localisation and Characterisation of Pseudo-Elliptical Components of Edge Images*, Proceedings of the 6th Scandinavian conference on Image Analysis, Aalborg, Denmark, 1991.

[32]K. Fukanaga, *Statistical pattern recognition*, Academic Press, San Diego, 1990.

[33]W.E.L. Grimson, *Object Recognition by Computer, the Role of Geometric Constraints*, MIT Press, Cambridge, Massachusetts, USA, 1990.

[34]R.M. Haralick, H. Joo, C.N. Lee, X. Zhuang, V.G. Vaidya, M.B. Kim, *Pose estimation from corresponding point data*, IEEE Transactions on Systems, Man, and Cybernetics, vol. 19, 1989, 1426-1446.

[35]R.W. Haralick, L.G. Shapiro, *Structural Descriptions and inexact matching*, IEEE PAMI vol. 3 no. 5, 1985.

[36]R. Horaud, *New methods for Matching 3-D Objects with single perspective views*, in IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No.3, pp. 401-412, May 1987.

[37]B.K.P. Horn, *Robot Vision*, MIT Press, Cambridge, Massachusetts, USA, 1986.

[38]Ch.-Ch. Hsu, J. S. Huang, *Partitioned Hough transform for ellipsoid detection*, Pattern Recognition, vol. 23, no. 3/4, pp. 275 - 282, 1990.

[39]R.C. Jain, Th.O. Binford, *Ignorance, Myopia and Naiveté*, CVGIP vol. 53 no. 1, 1991.

[40]A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.

[41] J.J. Koenderink, A.J. van Doorn, *The internal representation of solid shape with respect to vision*, Biol. Cybern. 32, 1979, 211-216.

[42]J.J. de Jong, *Learning 3D Object Descriptions from a Set of Stereo Vision Observations*, M. Sc. Thesis, Pattern Recognition Group, Delft University of Technology, 1992.

[43]J.J. de Jong, J. Buurman, *Learning 3D Object Descriptions from a Set of Stereo Vision Observations*, proceedings of the 11th ICPR, the Hague, the Netherlands, 1992.

[44]P.P. Jonker, editor, *The Architecture of the Delft Intelligent Assembly Cell*, Milestone 1 report SPIN/FLAIR-2 IV-1.1, TU Delft, December 1988.

[45]P.P. Jonker, W.F. Schmidt, P.W. Verbeek, *A DSP based range sensor using time sequential binary space encoding*, in: Proceedings of the workshop on Parallel processing, Bhabha Atomic Research Center, Bombay, India, Februari 1990.

[46]A. Kyrkos, J.J. Gerbrands, E. Backer. *A real-time binary image analysis system*, in Proceedings of EUSIPCO-88, Grenoble, France, September 5-8, 1988.

[47]J.S.J. Lee, R.M. Haralick, L.G. Shapiro, *Morphologic edge detection*, Proceedings, Eighth International Conference on Pattern Recognition (Paris, France, October 27-31, 1986), IEEE Publ. 86CH2342-4, 369-373.

[48]R. Lee, P.-C. Lu, W.-H. Tsai, *Moment preserving detection of elliptical shapes in gray-scale images*, Pattern Recognition Letters, vol. 11, no. 6, pp. 405 - 414, 1990.

[49] S. W. Lu, A. K. C. Wong, *Synthesis of Attributed Hypergraphs for Knowledge Representa-*

tion of 3-D Objects, in: Pattern Recognition, ed. J. Kittler, Lecture Notes in Computer Science, vol. 301, Springer-Verlag, 1988.

[50]S.D Ma, S.H. Si, Z.Y. Chen, *Quadric curve based stereo*, Proceedings 11th ICPR, 1992.

[51]P. Martens, *CAD/CAM for assembly planning*; Ph. D. thesis, Delft University of Technology, 1992.

[52]G. Medioni, R. Neviata, *Segment-Based Stereo Matching*, Computer Vision, Graphics, and Image Processing, vol. 31, pp. 2-18, 1985.

[53]B.R. Meijer, P.P. Jonker, *The Architecture and Phylosophy of the DIAC (Delft Intelligent Assembly Cell)*, Proceedings of the 1991 IEEE International Conference on Robotics and Automation, Sacramento, California, April 9-11, 1991.

[54]D. P. Miller, *A twelve-step program to more efficient robotics*, AI magazine, spring 1993.

[55]S.B. Pollard, J. Porrill, J.E.W. Mayhew, *Recovering partial 3D wire frames descriptions from stereo data*, Image and Vision Computing, vol. 9, no. 1, pp. 58-65, 1991.

[56]J. Porrill, *Fitting ellipses and predicting confidence envelopes using a bias corrected Kalman filter*, Image and Vision Computing, vol. 8, 1990, 37-41.

[57]D. Rogers, J. Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, 1976, pp. 76-83.

[58]C.J. Rijnierse, *An edge based stereovision implementation*, master's thesis, Pattern Recognition Group, Delft University of Technology, 1988.

[59]C.J. Rijnierse, F.C.A. Groen, *Graph construction and matching for 3D object recognition*, in: Pattern recognition and artificial intelligence - towards an integration, ed. L.N. Kanal, North Holland, Amsterdam, 1988.

[60]R. Safaee-Rad, I. Tchoukanov, B. Benhabib, K.C. Smith, *3-D pose estimation from a quadratic-curved feature in two perspective views*, Proceedings 11th ICPR, 1992.

[61]K. Sato, K. Ikeuchi, T. Kanade, *Model Based Recognition of Specular Objects using Sensor Models*, Proceedings IEEE workshop on Directions in Automated CAD-based vision, Maui, Hawaii, June 1991.

[62]Y. Shirai, *Three-Dimensional Computer Vission*, Springer Verlag, Berlin Heidelberg 1987.

[63]W.H. Tsai, K.S. Fu, *Error Correcting Isomorphisms of attributed relational graphs for pattern analysis*, IEEE SMC vol. 9 nr. 6, 1979.

[64]P.W. Verbeek, J. Nobel, G.K. Steenvoorden, M. Stuivinga, " Video-speed triangulation range imaging", in: Traditional and non-traditional robotic sensors, ed. T.C. Henderson, Nato ASI series, vol. F63. Springer Verlag, Berlin 1990.

[65]P.W. Verbeek, H.A. Vrooman, L.J. van Vliet, *Low-level image processing by max-min filters*, Signal Processing, vol. 15, no. 3, pp. 249-258, 1988.

[66]P.W. Verbeek, L.J. van Vliet, *Estimators of 2D edge length and position, 3D surface area*

and position in sampled grey-valued images, BioImaging, vol. 1, no. 1, 1993.

[67]L.J. van Vliet, I.T. Young, A.L.D. Beckers, *A nonlinear Laplace operator as edge detector in noisy images*, Computer Vision, Graphics, and Image Processing, vol. 45, no. 2, pp. 167-195, 1989.

[68]C. Williams, *Omzetting van Hans Buurman's TCL-Image command-files naar C*, internal report, Pattern Recognition Group, Faculty of Applied Physics, Delft University of Technology, 1992 (in Dutch).

[69]M. Xie, M. Thonnat, *A Theory of 3D Reconstruction of Heterogeneous Edge Primitives from Two Perspective Views*, Proceedings ECCV' 92, Santa Maguerita Ligure, May 1992.

[70]H.K. Yuen, J. Illingworth, J. Kittler, *Detecting partially occluded ellipses using the Hough transform*, Image and Vision Computing, vol. 7, 1989, 31-37.

Summary

In this thesis, a system is described for recognition of parts in a flexible assembly cell. It will be used in the Delft Intelligent Assembly Cell (DIAC). The system consists of a stereo vision part and a recognition part. The main chapters of the thesis are summarised below.

In chapter 3, the DIAC project is described. The tasks to be performed by DIAC define the demands on the recognition system. Parts have to be recognized and their position and orientation determined in a data driven way. The parts consist of straight lines and circular arcs. A two stage system for object identification in DIAC is proposed, of which the recognition system described in this thesis will be the second stage. Criteria are described for switching between the stages.

Chapter 4 describes the stereo vision part. After the images are acquired, edge detection is performed. Following that, a simple stereo vision algorithm is applied for finding straight lines. A straight line fit is applied to edges and left and right lines are checked for correspondence. The best correspondence for each is kept, if it is better than a certain threshold. Ellipse based stereo vision is also used, based on an algorithm to find ellipses or partial ellipses in images, and an algorithm to find the circle in 3-d corresponding to two ellipses. The algorithms appears to be sufficiently fast and accurate. The chapter ends with an evaluation of the calibration procedure.

In chapter 5, the recognition part is described qualitatively. It is based on hypothesis generation and testing, using careful pruning to reduce the computational complexity. Hypotheses about the presence of an object in the scene are based on one or two observed features, and are tested using the full scene description. Rotational symmetries, included in each model, are eliminated at an early stage of recognition. Knowledge about the rigidity of the object and the supporting surface is also exploited.

In chapter 6, the overall system (stereo vision and recognition) is evaluated, using a set of 200 test image pairs. Some of the limitations of the system are explored, and parameter settings are validated. Although the performance of the system is not yet sufficient for direct industrial application, it could be sufficient for DIAC. Some directions for improvement are indicated.

Chapter 7 addresses the origin of the models used for recognition. Models can originate from two sources: a model database, possibly derived from Computer Aided Design (CAD), or a learning procedure. For DIAC, the first approach is used. An interface has been defined between the Product Data Model and the recognition system. This allows automatic generation of models when new parts are introduced in the system. For the second approach, a learning procedure has been implemented based on principles from earlier chapters. It learns wire frame models of objects consisting of straight lines only, using a few observations of different stable positions of the objects.

Samenvatting

In dit proefschrift wordt een systeem beschreven voor herkenning van onderdelen in een flexibele assemblage cel. Het zal worden toegepast in de Delftse Intelligente Assemblage Cel (DIAC). Het systeem bestaat uit een stereo visie deel en een herkenningsdeel. De belangrijkste hoofdstukken van het proefschrift worden hieronder samengevat.

In hoofdstuk 3 wordt het DIAC project beschreven. De taken die door de DIAC moeten worden vervuld bepalen de eisen aan het herkenningssysteem. Onderdelen moeten worden herkend en hun positie en oriëntatie moeten datagestuurd worden bepaald. De onderdelen bestaan uit rechte lijnen en cirkelbogen. Een tweetraps aanpak voor de objectherkenning in DIAC wordt voorgesteld, waarvan het herkenningssysteem, zoals beschreven in dit proefschrift, de tweede trap zal zijn. Criteria worden beschreven om over te schakelen tussen de trappen.

Hoofdstuk 4 beschrijft het stereo visie deel. Nadat de beelden zijn verkregen worden randen gedetecteerd. Daarna wordt een simpel algoritme toegepast om rechte lijnen te vinden. Rechte lijnen worden gepast aan de randen en de rechter- en linkerlijnen worden vervolgens gecontroleerd op correspondentie. De best passende combinatie voor elke lijn wordt bewaard, als hij beter is dan een bepaalde drempel. Op ellipsen gebaseerde stereo visie wordt ook toegepast, gebaseerd op een algoritme om gehele of gedeeltelijke ellipsen in te vinden, en een algoritme om de cirkel te vinden in 3-d die correspondeert met twee ellipsen. Deze algoritmen lijken voldoende snel en nauwkeurig te zijn. Het hoofdstuk eindigt met een evaluatie van de calibratie.

In hoofdstuk 5 wordt het herkenningsdeel kwalitatief beschreven. Het is gebaseerd op hypothese-generatie en -toetsing. Hypothesen over de aanwezigheid van een object in de scene worden gebaseerd op een of twee waargenomen kenmerken, en worden getoetst met de volledige beschrijving van de scene. Rotatiesymmetrieën, die aan ieder model worden toegevoegd, worden in een vroeg stadium van de herkenning geëlimineerd. Kennis over de stijfheid van het object en het ondersteunende vlak wordt ook uitgebuit.

In hoofdstuk 6 wordt het gehele systeem (stereo visie en herkenning) geëvalueerd, met gebruikmaking van 200 beeldparen. Enige beperkingen van het systeem worden verkend, en de geldigheid van de parameterinstellingen wordt gecontroleerd. Hoewel de prestaties van het systeem nog niet voldoende zijn voor een directe industriële toepassing, kunnen ze wel goed genoeg zijn voor DIAC. Enige richtingen voor verbeteringen worden aangegeven.

Hoofdstuk 7 bespreekt de oorsprong van de modellen die voor herkenning gebruikt worden. Modellen kunnen van twee bronnen afkomstig zijn: van een modelgegevensbestand, mogelijk afgeleid van een met behulp van de computer gemaakt ontwerp (CAD), of van een leerprocedure. Voor DIAC wordt de eerste aanpak gekozen. Een verbinding is gedefinieerd tussen het Produkt Data Model en het herkenningssysteem. Dit staat automatische generatie van modellen toe, wanneer nieuwe onderdelen in het systeem geïntroduceerd worden. Voor de tweede

aanpak is een leerprocedure geïmplementeerd, gebaseerd op principes uit eerdere hoofdstukken. Deze leert draadmodellen van objecten die alleen uit rechte lijnen bestaan, met gebruikmaking van een paar observaties van verschillende stabiele posities van de objecten.

Dankwoord

Hoewel een proefschrift slechts een auteur heeft, is het schrijven ervan slechts mogelijk met steun van een groot aantal mensen. Ook hier is dat het geval, en ik wil met name de volgende mensen hartelijk bedanken:

Allereerst natuurlijk Bob Duin, zonder wiens vragen ik veel minder antwoorden zou hebben gevonden. Ook Pieter Jonker, die in zijn eentje het halve project deed, en Ted Young, de wetenschappelijk manager op de achtergrond, ben ik zeer erkentelijk.

De overige leden van de vaste staf droegen mede bij tot het feit dat de vakgroep een vruchtbare plek voor wetenschappelijk onderzoek werd. Natuurlijk ook de collega-promovendi, inmiddels uitzwermend van de Kaukasus tot Californië. De sfeer was zo goed dat er vaak tot diep in de nacht nagekaart werd.

Ik heb de samenwerking met studenten in de vakgroep steeds als zeer constructief ervaren. Vooral het gepuzzel samen met Johan de Jong heeft de inhoud van dit proefschrift positief beïnvloed.

Met Dave Bierhuizen van de vakgroep Informatietheorie was de project-samenwerking het meest intensief en voorspoedig. Ik wil ook de andere DIAC-collega's bedanken voor de goede samenwerking en de interessante kennisverbreding. Misschien moet iemand nog eens een reünie plannen (en schedulen).

Het afmaken van het boekje eiste toch meer dan verwacht. Frans Gerritsen, mijn baas bij PMS, bedank ik voor het gestelde vertrouwen en zijn constructieve begrip. Damir Sudar hielp mij mijn trouwe PowerBook te verkrijgen, waarop uiteindelijk nog veel van de teksten in dit boekje verwerkt zijn.

Dan zijn er altijd nog de mensen die zich meer zorgen maken om het welzijn van de promovendus dan om dat van het proefschrift. Mijn ouders, die me altijd gesteund hebben en alle kansen hebben gegeven. Mijn zus Suzanne, die me een rustige plek bood om te schrijven. Tenslotte mijn vrouw Frederike, die me vaak heeft opgevangen en die ik hopelijk niet helemaal van het promotie-idee heb afgeholpen.

Curriculum Vitae

Johannes (Hans) Buurman was born in Rotterdam in 1964. From 1976 to 1982 he attended the Rijksscholengemeenschap in Brielle, where he obtained the Atheneum B diploma. From 1982 to 1987 he studied Applied Physics (Technische Natuurkunde) at Delft Technical University.

In 1986, he joined the Pattern Recognition Group there, where he did the final work for his Master's (Ingenieurs) thesis in the field of computer architectures for image processing using a CLIP-4 system. From 1987 to 1992 he was a Ph.D. student (A.I.O.) in the same group, doing research on object recognition for robotics. This work is reported in this thesis.

Since 1992, Hans Buurman is with Philips Medical Systems in Best, where he is working on workstations for medical image processing.

"Is that it ?" said Eeyore.
"Yes," said Christopher Robin.
"Is that what we were looking for ?"
"Yes," said Pooh.
"Oh!" said Eeyore. "Well, anyhow - it didn't rain,"
he said.

A.A. Milne, *Winnie-the-Pooh.*