UNIVERSITE DE NEUCHATEL
INSTITUT DE MICROTECHNIQUE

3D Object Recognition Based on Surface Representations

THESE

PRESENTEE A LA FACULTE DES SCIENCES
POUR OBTENIR LE GRADE DE DOCTEUR ES SCIENCES

PAR

Hans Peter Amann

IMPRIMATUR POUR LA THÈSE

Three-Dimensional Object Recognition Based on Surface Representations

de Monsieur Hans Peter Amann

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel
sur le rapport des membres du jury,

Messieurs H. Hügli, F. Pellandini et Th. Pun (Genève)

autorise l'impression de la présente thèse.

Neuchâtel, le 16 octobre 1990

Le doyen:

Cl. Mermod

To Conradin

Table of contents

Summary

In this PhD thesis, a 3D object recognition system is presented which acquires raw 3D data, extracts a dense high level representation and matches it with model representations contained in a reference library. The result is an ordered set of partially verified hypotheses on the identity of the objects in the scene, their respective orientations and, optionally, their positions.
The most important contributions to help to resolve the problem of recognizing objects in this thesis are i) the "inexact representation matching algorithm" and ii) that the particular methods have been integrated in a complete object recognition system including data acquisition, data preprocessing, high level representation matching and verification.
The system is intended to be used in automatic vision systems, essentially for part assembling systems but also in autonomous robot navigation.

Résumé

Ce travail de thèse présente un système de reconnaissance d'objets tridimensionnels qui acquiert des données brutes provenant d'une scène, applique certains prétraitements, et extrait une représentation de haut niveau. Les éléments de cette représentation sont ensuite mis en correspondance avec les éléments des représentations de référence contenus dans une bibliothèque, permettant ainsi d'établir des hypothèses sur le type d'objet, son orientation, et éventuellement sa position dans la scène.
Les contributions majeures de ce travail à la solution du problème général sont, premièrement, l'algorithme de mise en correspondance de représentations incomplètes, décrit dans le chapitre 7, et deuxièmement la réalisation d'un système complet, incluant l'acquisition de données, le prétraitement, la mise en correspondance à un haut niveau de représentation, ainsi que la vérification des résultats.
Les applications d'un système de reconnaissance d'objets tridimensionnels concernent essentiellement les robot mobiles ainsi que les chaînes d'assemblage.

Zusammenfassung

Das Thema dieser Arbeit ist das Erkennen von dreidimensionalen Objekten. Ausgehend von rohen Eingangsdaten aus einer dreidimensionalen Szene wird nach einer Vorverarbeitung eine dichte Darstellung herausgezogen. Diese wird darauf mit den Darstellungen von bekannten Modellen verglichen. Dadurch können Hypothesen bezüglich der in der Szene enthaltenen Gegenstände erstellt werden die auch eine Aussage bezüglich deren Orientierung und Position erlauben.
Die Hauptbeiträge dieser Arbeit zur Lösung des Problems der dreidimensionalen Objekterkennung liegen in zwei Bereichen. Erstens im vorgeschlagenen Algorithmus für eine nicht-exakte Zuweisung von Modellelementen zu Elementen aus der Szene und zweitens in der Tatsache dass ein komplettes System realisiert wurde in dem alle Nichtidealitäten berücksichtigt werden mussten.
Systeme zur Erkennung von dreidimensionalen Objekten finden vor allem Anwendung in industriellen Montageketten und in Navigationssystemen für mobile Roboter.

1. Introduction

1. Introduction

Vision is the most important way for humans to localize, and to recognize objects in the real world. Therefore, quite naturally, efforts have been made since a couple of years in order to integrate similar facilities in automatic vision systems.
In artificial environments and for well defined problems, rather good results have been obtained for two-dimensional objects. For three-dimensional objects in turn, the efforts made have essentially shown the high complexity of the problem.

1.1 The human way to interpret 3D scenes

A human being uses a large spectrum of information at different levels abstraction, but also a large *a priori* knowledge for the recognition of 3D objects. Due to the complexity of the subject we do not enter into details but give a short list of ideas.

2D image interpretation (single instance)

Steady objects seen from a certain distance are interpreted as a single 2D image. On the lowest level of abstraction one or more of the following features can be used:

i) shape of the basic elements forming the image
ii) neighbourhood relations between basic elements
iii) intensity
iv) color
v) reflectivity
vi) others

On a next higher level, the data can be interpreted as a projection of a 3D object to a 2D image, using the big knowledge of a human. This allows to extract the orientation and to establish by comparison with known objects hypotheses on size and location.

3D image interpretation (single instance)

The same basic features are also available for 3D views. Additionally, two types of 3D information are available:

i) the disparitiy of the projections of one and the same 3D point on both images in stereoscopic views
ii) the mutual occlusion of objects
iii) others

sequences of 2D and 3D images

A very strong point in the behaviour of a human observer is his ability to verify its hypotheses and to observe an object actively. If the information is insufficient for a reliable interpretation of a scene, it can be completed, e.g. by one of the methods listed bellow:

i) stereo by motion
ii) change of scale and perspective

iii) handling of the object
iv) behaviour of the object (e.g. cars, animals, ...)

A movement of the object respectively the observer changes the viewing angle and gives rise to a sequence of stereoscopic images. If the movement has a radial component, the change in scale and perspective view with respect to the change in distance allows also some interpretation. Next, humans like to take objects in their hands, allowing them to move and rotate them.
At a higher level, mobile objects, either human made or natural objects, have typical behaviours which can be used for their recognition.

These few informations on the behaviour of a human allow to imagine how the task of intelligent object recognition, realized almost in real time by a human observer, is complex. Nevertheless, with the arrival of fast computers, and nowadays of specialized hardware, researchers got tools which allow i) the test of 3D object recognition algorithms, ii) to realize prototypes with limited features and iii) their application to well defined, relatively simple problems.

1.2 Domain of application

The domain of application for object recognition systems is virtually unlimited. Wherever a work has to be done by a machine, and where the initial givings are either not known or unstable, such systems can be useful We discuss two typical examples:

bin picking task

Given a bin, containing in an non-ordered way pieces to machine. An intelligent system, usually called robot, has the task to pick pieces one by one and to put them in a well defined manner at another place, e.g. in a container or on a conveyor belt.
This task which seems to be relatively simple contains already a big amount of problems. Here just a limited number of them.

i) How to find an accessible piece ?
ii) How to determine the position of the piece to pick (localization and orientation) ?
iii) How to determine the type of the pieces if ever there are different types ?
iv) How to handle pieces which are partially occluded ?
v) How to handle with shadows ?
vi) Which handling tactic has to be applied ?
vii) How to take profit of a priori knowledge ?

mobile robot

A second class of applications can be illustrated by the example of the mobile robot which needs even in a well known environment some 3D object recognition facilities for

i) the localization using visual references,
ii) obstacle avoidance,
iii) seek tasks and others.

In a natural environment where the kind of objects is generally nonpredictible the task is even more complicated, including the analysis of unknown objects.

While a first kind of problems, object localization and recognition, is typical for an industrial environment (sorting, machining and assembling), the second kind of problems, scene analysis, is typical for robot guidance in a unknown environment and for the description of unknown objects.

1.3 Aim of the work, problem definition

Due to the complexity of the problems encountered in the domain, we had to delimit our work to a well defined problem.

set

Given a scene containing 3D objects and a library of reference objects.

problems to resolve

i) Acquire the data of the 3D scene by any 3D acquisition system and do the appropriate low level preprocessing.

ii) Represent the data in a high level representation which is well adapted to the subsequent representation matching task.

iii) Match the scene object's high level representation one-by-one with the reference objects representation, attribute a measure of similarity and establish one or more hypotheses on correspondences between scene elements and reference object elements.

iv) Verify and classify the resulting hypotheses.

v) Extract information on the orientation and position of the object (optional).

constraints

We decided to add two constraints:

constraint 1: limitation to planar faced objects (polyhedrons)

constraint 2: use of a surface based high level representation

While they let the nature of the most important problems unchanged, they reduce in an important way the complexity of the overall problem.

1.4 A frame for an object recognition system

With the problem definition and constraints from above, we give a frame for a 3D object recognition system we discuss in the present report (fig 1.1, next page). Other approaches to the 3D object matching problem will be mentioned in a next paragraph (§1.5).

In the following chapters, we discuss the intermediate steps of the frame one-by-one, insisting in particular on three partial problems which attracted our interest (fig 1.1, hatched blocks):

i) high level data representation,
ii) high level representation matching, and
iii) interpretation of the matching results (verification, object recognition)

Chapter 2 presents first the two 3D scene data acquisition systems developed and used in our laboratory. They are, in chronological order i) a passive stereo vision system and ii) an active laser light plane acquisition system. Then, we justify our choice of the active system for the further work and present some ideas for the low level preprocessing.

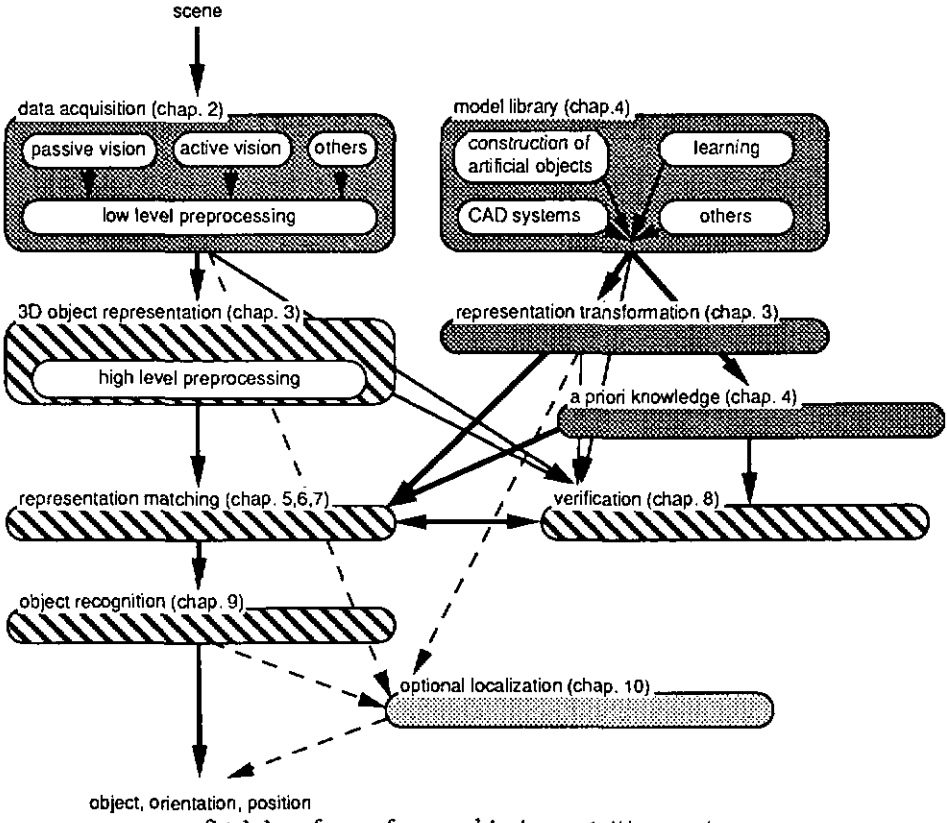


fig 1.1: a frame for an object recognition system

In the following chapter 3, the important subject of 3D object representation is treated. First, we focus on the fundamental differences between surface based "vision representations" and volume based "CAD representations". Both of them are of interest since most of the 3D-

acquisition systems furnish surface based representations while the sources for models, i.e. CAD systems, supply volume based representation. Therefore, a separate paragraph (§3.3) discusses the transformation of representations. Next, we present our choice of a high level representation, the "Planar Face Representation Graph" PFRG (§ 3.4). It contains a minimum of data necessary to represent planar faced objects for the subsequent matching task. The chapter terminates with some high level pruning methods, which reduce a PFRG to its most important faces.

The construction of a model library is the subject of chapter 4. Note that in general, this job is not time critical. It is done off-line, allowing the preparation of the data useful to speed up later on-line high level matching.

In chapter 5, the basic problem of representation matching is defined and two major approaches to solutions presented: exact matching (§ 5.3) and inexact matching (§ 5.5). The first approach using exact matching is useful as an introduction to the subject and allows to gain some experience (chapter 6), but for the natural 3D scene data we acquire, it is not sufficiently robust. The second approach, inexact matching, is more difficult to handle but has the big advantage to be more robust with respect to incomplete or error prone data. It has finally been chosen for the remaining work (chapter 7).

Chapter 8 and 9 are concerned with the interpretation of the matching results. First, the hypotheses on correspondences between 3D scene elements and reference elements have to be verified, combining eventually compatible partial solutions. Second, the object recognition task itself can be done, interpreting the resulting set of verified hypotheses. Finally, the verified correspondences can be used for the determination of the orientation and eventually the position of the object in the 3D seene space.

We terminate with the conclusions, the outview on further research topics and the references.

1.5 Related work

3D object recognition and related work has been the subject of research since the end of the 1970's. We give a short overview on work in the domain of object recognition.

1.5.1 3D data interpretation

With 3D data acquisition systems, rich information is available and features as the surface orientation and the relationships between surface patches can be exploited. Their drawback with respect to 2D systems is the higher complexity due to a much bigger number of degrees of freedom.

	translation	scale	rotation	degrees of freedom
2D	x, y	α	ψ	4
3D	x, y, z	α	ψ,θ,λ	7

fig 1.2: degrees of freedom

extended gaussian images (EGI)

Horn and al. proposed the use of a subset of this information, the extended gaussian images EGI, representing in a global way the orientation of the faces of an object [Hor84a], [Hor84b](§3) The method is limited to few applications but, for a fast determination of the orientation of a known object, they can be useful.

attributed relational graphs (ARG)

The most important kind of approaches is the one grouped around attributed relational graph representations of the 3D scene. Since the correspondences between scene elements and ARG elements can be defined rather freely, a huge number of similar implementations exist for both "boundary based methods" and "surface based methods". The most important distinctions are on the level of the representation matching algorithm and, less important, on the level of the dissimilarity measures. Table 1.3 [Ama89] gives a detailed list of ARG based 3D object recognition systems.

We will come back to the most important approaches ([Osh83], [Esh84], [Osh87], [Fau87]), in their respective chapter of concern.

recognition method	problems	references (examples)
exact topology		
graph/graph-isomorphism	searchtree size	[Ull76] (exhaustive search)
		[Esh84] (backtracking search)
pruning criterions		
a priori exclusions		[Ull76] (structure)
		[Osh87] (attribute dissimilarity)
branch exclusion		[Har79] (test strategies)
		[Osh87] (attribute dissimilarity)
		[Fau87] (rigidity constraint)
subgraph/graph-isomorphism	searchtree size	(see above)
a priori exclusions		
branch exclusion		
subgraph/subgraph-isomorphism		
exhaustive search	searchtree size	[Esh84]
reduced search		
"root" matches		
using aspect graphs		[Bow88]
using attributes		[Fau87], [Osh87]
using topology		[Fen88]
(cycles and circles)		
inexact topology		
non-attributed		
full match	number of potential solutions	
partial match		
using circles, cycles,...	criterions	[Fen88]

table 1.3 (part I): representation matching using attributed relational graphs

recognition method	problems	references (examples)
attributed		
"root" match and subsequent extension		
rigidity constraint hypothesis, prediction and verification	starting point application dependency	[Fau87] [Bol87]
exhaustive tree-search introducing "null-elements", priority to branches using no "null elements"	(see above) cost definition	[Esh84] [Bow88]
no topology		
non-attributed	impossible!	
attributed		
exhaustive combination, HT and clustering	dimensions of the Hough space	[Bow88]
"root" match and extension using rigidity constraint (searchtree-candidate prediction and verification)	starting point	[Fau87]

table 1.3 (continued): representation matching using attributed relational graphs

1.5.2 2D data interpretation

A second way to do object recognition is based on the interpretation of 3D scene data projected to a 2D image. Hypotheses on correspondences with known 3D objects of a model library are established allowing to project the model to the 2D space for verification.

[Gmu89] describes an object recognition system, [Bol87] a system destinated to the orientation of parts for machining, [Str86] ("one eyed stereo") and [Ham87] ("shape from shading") ideas for more general data acquisition systems.

1.6 Readers Guide

For the present work, all the intermediate steps mentioned in figure 1.1 above have been realized and are discussed. Readers familiar with the domain can skip a certain number of less important chapters and paragraphs. Table 1.4, next page, indicating the most interesting parts, can be used as a reader's guide.

chapter 2	Acquisition of 3D scene data
§ 2.3	Choice of the laser light plane system
§ 2.4	Low level data preprocessing
chapter 3	3D Object representations
§ 3.4	Choice of the Planar Face Representation Graph PFRG
§ 3.5	High level preprocessing
chapter 5	Representation matching
chapter 7	Inexact matching of high level representations
chapter 8	Verification
chapter 9	Object recognition

table 1.4: reader's guide

2. Acquisition of 3D Scene Data

2.1 Passive stereo vision

2.2 Laser light systems

2.3 Choice of the laser light plane system

2.4 Low level data preprocessing

2.5 Conclusions

2. Acquisition of 3D Scene Data

A first step in 3D object recognition is the acquisition and preprocessing of raw input data. A general view of acquisition methods currently used has been given in [Ama86a]. Therefore, we will limit ourselves in the present chapter to the discussion of the two methods investigated in our work.
At first, a passive stereo depth extraction system has been used. Later, a laser light plane ranging system has been built up. It will be discussed briefly, whereas a detailed description can be found in [Mai88].

2.1 Passive Stereo Vision

basic idea

Two two-dimensional images of an equal scene, called IMAGE1 and IMAGE2, are acquired. For every significant point P of the 3D scene, the projections P_1 and P_2 are to be found on both images. Their disparity, combined with the information on the acquisition device (geometry and optics) provides the information necessary for the reconstruction of the 3D position of the original point P (fig 2.1) [Ama86a].

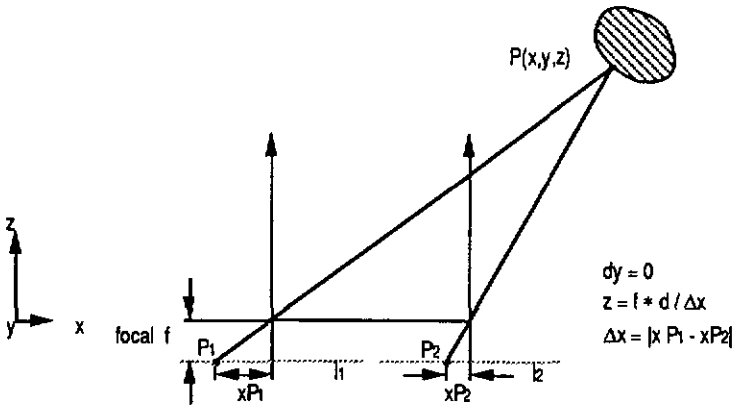


fig 2.1: stereo acquisition system

Passive stereo is of very big interest due to its universal possibilities of use. Its main advantages are that

i) it is a passive method, it can be used without artificial illumination,
ii) its conception is near to the human way to see an 3D scene and we know of its potential performances.

However, there are answers to be found to the following problems:

- the calibration of the acquisition system,
- the identification of corresponding projections (P_1, P_2) of an original 3D point P,
- the approximation of the acquired data by higher order representations (planes, ...) and
- the limited resolution in depth of such a device.

On the next pages, we comment interesting points and give some references to related work.

2.1.1 Calibration of the image acquisition system

The acquired data is completely known only if a calibration process for the acquisition system has been executed before. This process has to be done explicitly since

i) the exact position of the couple of cameras (geometry) is difficult to know.
ii) the optics as well as the light sensitive device (tube, CCD) are always fabricated and mounted with some tolerances.

Details on calibration processes can be found in [Fau84], [Fau87], [Aya87a] while the (time-variable) nonidealities of the commonly used CCD cameras and the accompanying electronics are discussed in [Bey87], [Bey89a] and [Bey89b].

2.1.2 Feature extraction

The most important problem in Stereo Vision is how to find corresponding couples of features (F_1, F_2) in both images which represent one and the same original feature F of the 3D scene. In fact, the problem can be separated into two: i) how to extract features F_i, from the acquired images and ii) how to find correspondences between them ?
In the present section, we limit ourselves to the first subject, the second being discussed in § 2.1.3.

extraction of significant features

The most common methods extract the inhomogeneities in the stereoscopic images which are due i) to edges (variation of the exposition of a surface with respect to the light source and the observer) or ii) to variations of the properties of a surface. The localization of these inhomogeneities is of utmost importance since the precise determination of disparity, and hence depth, depends directly on it.

Since edge detection for feature extraction is a very vast domain, we just indicate some references covering the above aspects. [Ros76] gives an introduction to the commonly used operators for 2D high pass filtering, [Bak81] and [Oht85] apply it to stereo vision using operators of variable size. An alternative using IIR filters has been shown in [Cas85] and [Der90].

All these algorithms take into account only the information present on one of the stereoscopic images, providing distinct results for both views since i) the angles of view, and so the directions of illumination, will be different, a fact which is natural for a Stereo Vision device and ii) the occlusion of background elements will be different in both images. Hence, an error-tolerant and flexible algorithm must be found for the subsequent correspondence search step.

Other authors use region based feature extraction [Gag87] or a combination

of both methods [Mon87], [Wro87]. [Gag87] and [Wro87] propose approaches which take both images into account for the determination of "significant points".

2.1.3 The correspondence search problem

In order to reconstruct the 3D position of a scene element, we i) must extract its projections in the stereoscopic images and ii) bring the respective elements into correspondence. The first point has already been discussed before while the second point, even more difficult, is the subject of the present section.

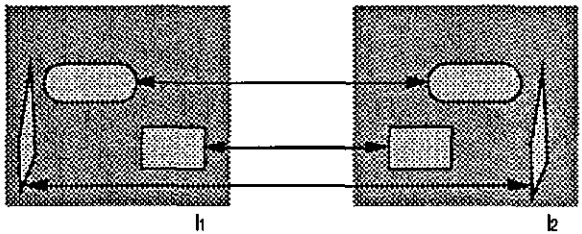


fig 2.2: corresponding elements of two stereoscopic images I_1 and I_2

The following problems are to be discussed:

i) how to establish hypotheses on correspondences between the projections P_1 in image I_1 and P_2 in image I_2 of an original 3D point P ?
ii) how to verify these correspondences ?
iii) how to profit from *a priori* knowledge about the acquisition device ?
iv) how to avoid computational complex "brute force" methods ?

a priori knowledge

The computational complexity of the correspondence search problem is so high that the inclusion of *a priori* knowledge about the acquisition system is indicated. In Stereo Vision, the basic acquisition device can be considered as two cameras of the same optical length. It is easy to see that corresponding projected points P_1 and P_2 must lie in a planar surface defined by the triplet $(P.f_1, f_2)$ where f_i is a focal point (fig 2.3). This plane intersects with the two image planes I_1 and I_2 in the straight lines s_1 resp s_2, called epipolar lines.

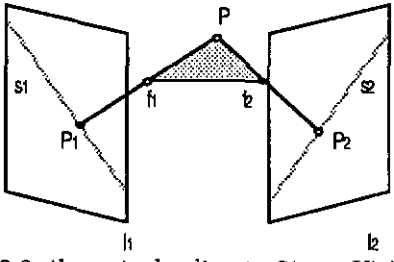


fig 2.3: the epipolar line in Stereo Vision

Turning the reasoning around, we can see that every projected point P_2 must lie on an epipolar line defined by the focal points f_1, f_2, the first projected point P_1.and the image plane l_2.

This approach has been used in many of the works in the domain of "Stereo Vision" [Bak81], [Oht85], [Gag87], [Wro87].
A second way to limit the correspondence search process is the use of geometric constraints. Generally, the 3D workspace is limited either by geometric constraints (robot's work space) or optical limitations (depth of focus). They can be transformed into limitations on the disparity allowed, reducing the search spaces on the epipolar lines. This approach has been discussed in detail first by [Arn80], other authors used it implicitly [Bak81], [Oht85].

identification problem

Until now, we supposed to look for correspondences between couples of well defined image points P_1 and P_2. In fact, the data attributed to these points P_i is very limited and usually insufficient for a reliable identification. Therefore, adjacent image data must also be considered either by a simple local correlation or by the use of informations on adjacent edges and regions. Here some examples:

- [Oht85] applies a consistency condition on neighboring edge points for the establishment of new correspondences.
- [Ama88b] and others propose the use of the edge orientation at the specific edge point as an additional feature.
- [Gag87], [Mon87] and [Wro87] make use of the attributes of the regions which are separated by the respective edge point.

The most promising approach is the third one since it uses well identifiable regions for the matching and, subsequently, the region borders (edges) for precise localization.

correspondence search by graph matching

Both stereoscopic views are preprocessed separately in order to extract the regions and their features. Then, a high level representation of the images in the form of attributed relational graphs is created. The nodes correspond to regions with their attributes, the arcs stand for neighborhood relations.
Naturally, in a general Stereo Vision application, the graphs extracted from both stereoscopic views will be nonisomorphic, this due to the acquisition and extraction nonidealities (occlusion, resolution, localization, preprocessing, ...) discussed before. Nevertheless, graph matching is possible and will be discussed in detail in chapter 5.

A good example of the application of the above ideas has been described in [Gag87]. A hierarchical region matching using the epipolar constraint is applied, based on the hypotheses that i) big 3D object surfaces are projected to big regions most probable to appear in both stereoscopic images and ii) that they are more reliably identifiable than small ones. The algorithm starts therefore with the so-called "mother regions", delivering constraints for the subsequent matches of smaller, adjacent regions.

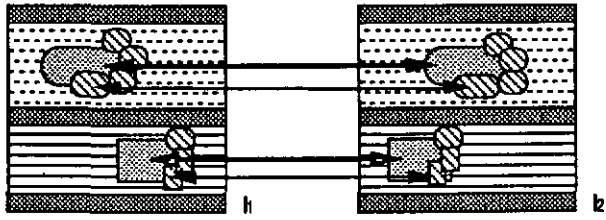


fig 2.4: hierarchical region matching using the "epipolar band" constraint

An "epipolar band" can be defined for each region, allowing the limitation of the number of candidate regions for correspondence tests.

correspondence search by correlation

A possible mean for image matching is the classical 2D correlation approach, eventually limited by additional constraints concerning the regions to process. This approach has two major disadvan-tages to be i) computational complex and ii) to detect just local maximums. The relationships with respect to other edges does not contribute to the result. Therefore, a second approach called "dynamic programming", is preferred.

correspondence search using dynamic programming

We have seen that the epipolar constraint allows to reduce the general 2D correlation to a 1D correlation on the epipolar lines of both "left" and "right" Stereo Vision images (fig 2.5).

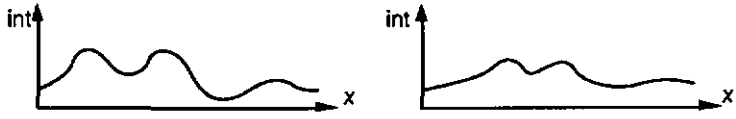


fig 2.5: intensity profiles on "left" and "right" epipolar lines in Stereo Vision

We now replace the general 1D correlation approach by the "dynamic programming" DP [Sak78]. While the correlation approach made attempts to "match" relatively small portions of the epipolar line (correlation window) with the other one, the DP approach tries to find a "best overall match" for the complete epipolar lines, taking into account already matched neighboring segments.
It is based on the hypothesis of monotony, the sequence of objects seen by the "left" and "right" camera, represented by segments on the epipolar line, must be the same. This restriction excludes very thin objects which could appear in inverted order on both epipolar lines. In turn, it does not exclude occluded objects.
DP works for discrete parts of the epipolar lines. Therefore, they must be segmented first, using usually edge extraction techniques as mentioned before.

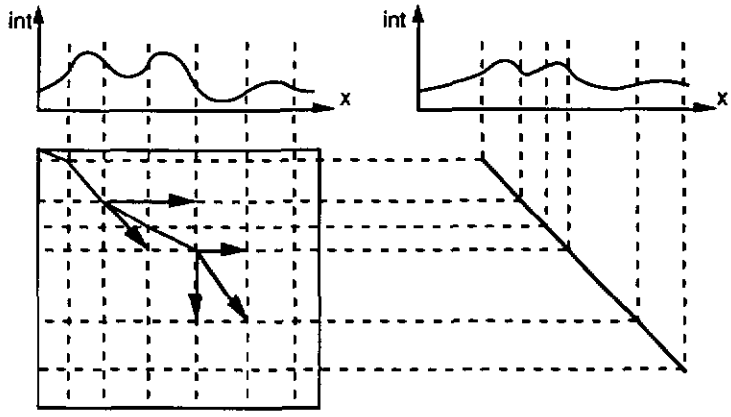


fig 2.6: segmented epipolar image lines (top) and the DP search field (bottom)

DP attempts to match the extracted segments aiming a "best overall match". This process corresponds to the task of finding an optimal path through the DP search field from the top-left to the bottom-right corner, while passing through a number of intersections.The search criterion is based on the weighted sum of the dissimilarities of the segments brought "locally" into correspondence.
The search path is allowed to be parallel to the grid, indicating the match of a segment with a nonexistent, occluded segment out of the second image.

DP matching, a promising idea, has also some drawbacks: the constraint due to the hypothesis of monotony mentioned before and the need for error-prone image segmentation.
In order to enhance the quality of the matchings, a coarse-to-fine DP procedure has been proposed by [Bak81], an extension of the 2D DP-search field to 3D, taking into account matchings done previously for neighboring epipolar lines, by [Oht85].
For more details refer to [Sak78], [Oht85], [Bak81], [Ama86a], [Ama86b] and [Ama88b].

2.1.4 Improvements and verification

In a conventional Stereo Vision approach using two cameras, a certain number of image points will be seen by one camera only, this due to occlusion, and the use of the information acquired will be very restricted. Therefore, [Ito86] proposes the use of a third camera in order to enhance the probability that a given 3D point can be seen at least from two cameras simultaneously.

If the 3D point is visible from all the three cameras at the same time, a hypothesis on its 3D coordinates can be extracted using just two stereoscopic views, using the third one for a low cost verification [Aya87b].

Further investigations have been made in the domains of subpixel edge localization and in improvements of the cameras electronics, both in order to optimize the precision of localization in the stereo images and hence in

the spatial resolution of the build-up [Bey89].

2.1.5 Conclusions on Stereo Vision

Due to its universality, Stereo Vision is of big interest. Its main advantages are:

i) Stereo Vision is a passive method, it can be used without any artificial illumination.
ii) The conception is near to the human way to see an 3D scene and we know of its potential performances.

Unfortunately, the generality of the stereo approach has to be paid heavily:

i) The computational costs for the range finding are very high.
ii) The reliability of the result is limited (region growing, edge forming, identification).
iii) A good resolution in depth is difficult to achieve: a large baseline of the cameras must be paid by more occlusions, a better spatial resolution of the acquisition devices by a higher number of pixels in the image and hence an increased computational cost.

Therefore, for each application, an extensive study on the 3D-data acquisition system to use has to be made, excluding computational overhead by the use of an appropriate technique.

2.2 Laser Light systems

A second approach for 3D depth extraction, completely different from the stereo approach, is the one using artificial scene illumination. We present one method, a laser light plane ranging system used in our laboratory [Mai88]. For additional data acquisition systems using structured light refer to [Ama86a].

basic concept

A laser light plane is projected on the 3D scene. The intersection of the laser light plane with the objects results in a set of curved and straight line segments which can be acquired by a camera whose view point and direction of view is different from the ones of the laser source.
Knowing the geometry and optics of the projector and the acquisition device, it is possible to determine by triangulation the 3D coordinates of the illuminated points of the scene. A sweep of the light plane over the scene allows to acquire a set of images, representing data from all the object points which are visible to both camera and projection system for a given build-up.

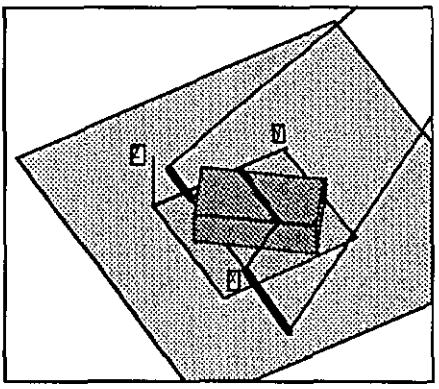


fig 2.7: Principle of the laser light plane 3D acquisition [Mai88]

The following points are interesting to note.

i) The laser approach is active and needs an artificial illumination. The application is limited to indoor scenes.
ii) All points of a surface intersecting the laser light plane are known, not just "significant points" as with the Stereo Vision approach.
iii) A calibration is necessary.

extension, acceleration

In the original system, the images for different projection angles of the light plane have been acquired subsequently. In order to accelerate the acquisition, a device has been proposed which allows to acquire all the information at once, replacing the unique laser light plane by a set of color coded light planes projected in parallel [Boy87].
It can be shown that it is possible to create N-unique color sequences which can be projected on the object [Hug88a], [Hug88b], [Hug89]. For M consecutive stripes projected to an object's surface, M≥N, the subsequences can be located in the complete sequence and, therefore, the projection angles determined. Using this system, the data acquisition system can be simplified in an important way:

- there is no more need for a sweeping laser source
- the acquisition is limited to just one (color) image, and
- the data acquisition is much faster.

Unfortunately, the applications of this approach are limited, this mainly for two reasons: the need for N-subsequences limit the resolution and the system must have color interpretation facilities, adding supplementary constraints for the interpretation of colored scene objects.

final result

Finally, the two approaches, Stereo Vision and Laser Rangefinder join at the point where sets of 3D surface points must be interpreted in order to get higher level representations of the scene data (line segments, faces, ...).

2.3 Choice of the laser light plane system

	type	measure	transformation to edge segments	transformation to surface patches	choice
stereo vision	passive	points where intensity changes	easy	difficult	-
laser light plane	active	geometry where illuminated	possible	easy	✔

table 2.8: stereo vision vs. laser light plane systems

Table 2.8 shows the most important reasons which lead us to abandon the stereo vision approach: it is not very well suited if we aim a surface based representation of objects. Additionally, there are two more important disadvantages.

- The Stereo Vision system uses already for the determination of depth a rather complex matching process which inherently introduces some incertitude on the correspondences. With the two cameras of the basic system, it is difficult, or even impossible, to verify the results.

- The basic Stereo Vision system with two cameras imposes important restrictions on the resolution which can be obtained. An extension (third camera, moving camera, ...) complicates the approach further.

Since the object recognition problem is difficult to treat, we have preferred the laser light plane acquisition system which is more restrictive in the sense that it uses artificial illumination. In turn, it delivers a huge number of directly useable 3D points with a good precision. This choice allows us to base the subsequent preprocessing and object recognition steps on reliable data.

2.4 Low level data preprocessing

We have seen before that the output of the 3D acquisition systems is limited to sets of points in the space. There are different reasons which motivate the conversion of these sets to other forms of representation:

i) the data structure will be simplified when basic elements are grouped,
ii) the visual representation will be enhanced,
iii) the object recognition task will be simplified, and
iv) the limited resolution of the input devices can be overgone.

In order to obtain a "vision representation", we attempt to approximate 3D point sets by more appropriate descriptions such as surface patches of various degrees, splines, ...
While the "region growing" approach mentioned in the context of the Stereo Vision was two-dimensional, using intensity and texture as criterions of homogeneity for an eventual merge of regions, we extend these ideas to 3 dimensions: neighboring regions will be merged if their 2D and 3D

properties are sufficiently similar [Bal82].
Since the problems have been discussed earlier, we discuss only the following three differences between the 2D and the 3D case:

i) search for neighboring points in the set of 3D points
ii) region growing, similarity measure and surface fitting

triangulation of Delaunay

The well known triangulation of Delaunay is often used to resolve this problem [Bra87]. The method fits tetrahedrons between quadruples of "neighboring points". Unfortunately, the problem is not yet resolved since the only criterion in the Delaunay algorithm to find "neighbor" points, which have to be linked to form tetrahedrons, is the geometric distance between 3D points. Therefore, it is possible that quadruples of points which are lying nearby in the 3D space form a tetrahedron while the projections of the scene to the acquisition system show clearly that behind this tetrahedron additional points have been detected. This situation, quite seldom at a first glance, can also appear when mismatched 3D points form singularities in the space, the extension of the algorithm can be used for error recovering, too. An extension, for example the one proposed in [Bra87], has to be added to the basic algorithm.
Finally, in scene analysis, the merge of the faces of all the tetrahedrons which are not occluded by other tetrahedrons, form the surface patches of the 3D scene objects and of the background. By this way, we can find some polyhedral representation of the surface of the object to recognize.

region growing, surface fitting

Once the elementary surfaces extracted, they can be grouped into bigger sets, using a 3D region growing algorithm similar to the algorithm discussed before for two dimensions. Beside the criterions of homogeneity we mentioned for 2D (e.g. intensity, texture, ...), we take advantage of the 3D features: orientation of the regions, 3D location, ... [Sha85] gives some general ideas of the formulation of homogeneity criterions.

If a representation with low order surface descriptions is desired, surface fitting methods can be applied. For details, refer to [Gri86], [Mai89] and [Mai90].

discussion

Of course, data preprocessing depends heavily on the data representation chosen for the subsequent object representation and recognition steps. Therefore, it is difficult to give detailed hints without loosing generality.
[Gri86] can be helpful for the understanding of these problems and offers some solutions to real applications. More specific, data preprocessing for the laser light plane system used in our laboratory can be found in [Mai88], [Mai89].

2.5 Conclusions

We have presented two major approaches for 3D scene data acquisition:

i) a Stereo Vision acquisition system, usable in a wide field of applications on relatively short distances and
ii) a Laser Rangefinder, usable only with special illumination and therefore limited to indoor scenes.

The advantages of the second system which are precision, resolution, amount of acquired data, ..., made us abandon the first one with which we experimented during a long period.

All the data used later on in this thesis have been acquired using the Laser Rangefinder [Mai88]. Nevertheless, the subsequent processing steps (representation, matching, ...) could also be applied to data providing from a stereo vision system, this under the condition that the data reaches a sufficient level of reliability and precision.

3. 3D Object Representations

3. 3D-Object Representation

In the present chapter, we show first the requirements for a high level representation of 3D objects, discuss a limited number of common object representations and the transformations between them. Next, we present i) the representation we have chosen for the later matching algorithms and ii) a second representation useful for visual inspection by a supervisor.
Finally, we show some high level preprocessing methods for the PFRG representation chosen aiming at pruned high level data which is less dependent on data acquisition no-idealities and low level processing.

3.1 Requirements

Once the data acquired and preprocessed, both scene object and reference (model) object should be brought into the form of a high level representation which fulfills some specific requirements.

i) rotation and translation invariance
ii) abstraction, structural representation
iii) intrinsic features of the objects

3.2 Volume and surface based representations

A large number of object representations have been investigated. We can classify them into two groups: i) volume based representations and ii) surface based representations

volume based representations

The first class is formed of representations which have their roots in CAD applications. Therefore, terms and methods are used which are near to the machinery world. Volumetric descriptions use i.e. cylinders and cubes as base elements, easy to be built on machines. These base elements can be rotated, translated and then composed to form a new object. The composition, again as in machinery world, knows two ways, the addition (for assembling, mounting) and the subtraction (for drilling, turning).The surface aspect, so important in vision, is secondary [Bha87b]. We will use the expression "CAD models" for this class of representations.

surface based representations

The second class is formed by representations where the visual aspects dominate. This kind of representations, called "Vision models" describe essentially the surfaces and their attributes.

In the following sections, we will discuss some of these representations.

3.2.1 Volume based representations

They describe the space occupied by an object. Instead of describing a huge number of spatial points, it is possible to use combinations of primitives, possibly of different sizes and shapes. Each of the primitives is thus

described by its geometric parameters. Ordered by increasing generality, the following representations can be used:

- pure primitive instancing
- space occupancy enumeration
- cell decomposition
- constructive solid geometry CSG
- others (e.g generalized cylinder, ...)

3.2.1.1 The space-occupancy representation

The "space occupancy representation" is the most simple way to describe a space:

- the workspace is divided into base elements (polyhedrons, e.g. cubes) and then,
- all the base elements which have an intersection with an object marked as occupied.

There is just one disadvantage to mention. The description of the space occupied will always be rough. The number of base elements being limited in practice, the resolution in space will be limited too. While this disadvantage makes this representation less useable for precise descriptions of objects, it can be very helpful in other domains. [Lum85] and [Shn87] describe an application in a system developed for robot guidance and collision avoidance, while a much preciser representation is used for the description of identified objects in the workspace. The space occupancy model is used to represent (approximatively) the space occupied by both identified and not identified objects.

In practice, the space representation is rarely used with base elements of equal size. In fact, it is more economic to use a kind of tree-structure, called "octrees" [Jac80]. The idea is to divide the space in a hierarchical manner (e.g. into cubes): as long as a given subspace (e.g. cube) is not homogenous and a given minimal size is not reached, the subspace is divided in smaller subspaces of the same shape.
[Hon87] as well as [Dye84] describe algorithms for the translation and rotation of objects represented by octrees.

3.2.1.2 Constructive Solid Geometry (CSG)

Constructive solid geometry CSG, used widely in CAD systems, represents objects as a binary tree where each leaf represents an instance of a primitive and each node an operation on its descendents. Primitives such as blocks, spheres, cylinders and cones are first transformed by translation, rotation, and scaling and then combined from bottom up by union, intersection or difference.

This representation is unambiguous but not unique, i.e. it can have different decompositions of an object using the same set of primitives. CSG is sufficient to cover most conventional, unsculptured objects. For these objects, similar objects give similar representations. For sculptured objets in turn, the big number of primitives to use can lead to different

representations for similar objects.
CSG, while well suited for CAD applications and quiet economic by its data structure, has its disadvantages when used as a source for "vision models" representations:

- the surface description, always needed for robot vision is relatively expensive to calculate from the CSG model,
- the way back, from the surface description to the set of primitives of the CSG model is difficult and not unique or even impossible.

3.2.2 Surface based representations

Surface based representations form the second class of object representations. It includes the representations which result from views of objects rather than volumetric constructions.

3.2.2.1 Boundary representation (B-rep)

In computer vision and graphics, the most common way to represent an object is to show the enclosing surface. A solid is represented by segmenting its boundary into finite number of bounded subsets, usually faces or patches.
If we use planar patches only, each face is represented by its edges and vertices, resulting in polyhedral or polyhedron-approximated objects. In order to describe curves surfaces efficiently, splines are used in many CAD systems, an interesting way which needs, however, a lot of computation time [Coh80]. Therefore, a big number of systems is still limited to planar patches.
Boundary representation (B-rep) can be derived from 3D range data directly. But there is still the problem that different tesselations of the surface can stand for the same object.

3.2.2.2 Surface Curvature Representation

At each surface point, the curvature can be measured in order to characterize it. Particular values can be extracted and used in similarity measurements:

- the mean curvature and
- the product of both minimum and maximum curvature, called "Gaussian Curvature".

Consult [Bes86] for more details, [Bha87b] for some examples and [Ric87].

3.2.2.3 Gaussian sphere and EGI's for convex polyhedra

A first theoretical base for the representation of convex polyhedra is already very old. Minkowski showed in 1897 that a convex polyhedron is fully specified, up to translation, by area and orientation of its faces [Min97]. We can represent this information by point masses on a sphere.

Gaussian sphere

The gaussian sphere representation of an object is obtained by moving the unit surface normals on the faces of a polyhedron in a way such that the tails of the vectors lie on the origin. The head lies now on the surface of the sphere, and each point on it corresponds to one particular surface orientation (fig 3.1, left).

Extended Gaussian Image EGI

The EGI of a polyhedron is obtained by placing a mass at each point of the sphere which is equal to the surface area of the corresponding face (fig 3.1, right).

At a first glance, very much information seems to be lost using this representation. Nevertheless, it can be shown that, again up to translation, the EGI defines a convex polyhedron [Smi79]. It is even possible to recover a convex polyhedron from its EGI when using the above definition [Lit83].

properties of the EGI

Extended gaussian images have the following properties [Hor84a]:

a) EGI's are not affected by translation.
b) A rotation of an object induces an equal rotation of the EGI.
c) Mass distributions which lie entirely in one hemisphere of the gaussian sphere do not correspond to closed objects.
d) The total mass of the EGI is equal to the total surface area of the polyhedron.
e) For a closed polyhedron, the projected area will be the same for any pair of opposite directions.
f) The center of mass of an EGI lies at the origin of the sphere (closed object).

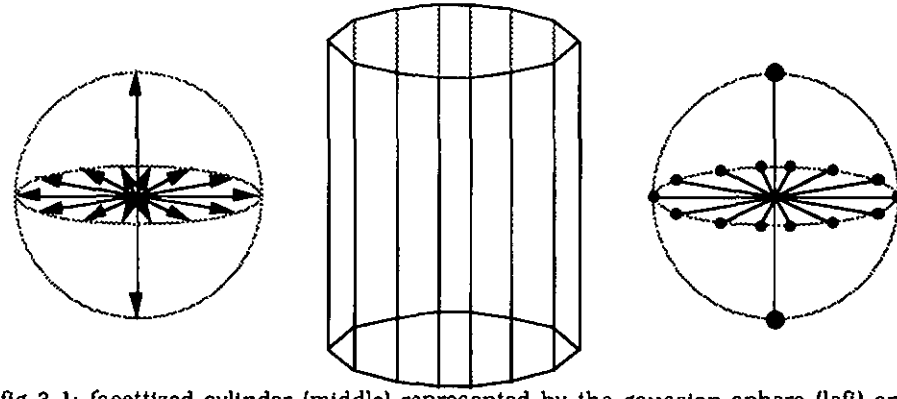


fig 3.1: facettized cylinder (middle) represented by the gaussian sphere (left) and the EGI (right)

extension for non-convex objects

When extending EGI's to non-convex objects, one of the properties from above is lost:

> When several surfaces of one and the same orientation occur, they will be mapped on the same point of the gaussian sphere and participate to a same mass point of the EGI. Since such surfaces of same orientation can occur, the representation is no more unique and the theorem of Minkowsky cannot be applied any more.

natural scene data

For objects whose data has been acquired by scene data acquisition systems, the acquired data will probably not be complete since self occlusion of the objects can occur, some more properties will be, at least partially, lost.

Still, EGI's are very useful for a fast comparison of the visible parts of objects.

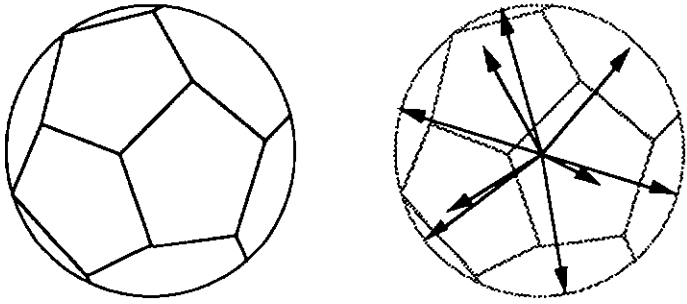


fig 3.2: tesselated sphere (dodecahedron) and its needle map

needle maps

For applications on computers, we use a discrete form of both Gaussian sphere and EGI, the needle maps. Tesselating the Gaussian sphere regularly, we get a limited number of planar patches which define a polyhedral so that we can apply the rules defined before (fig 3.2) [Hor84b].
A similar representation is known under the name "spike-model" or "needle map" where the vectors are no more of unit length but represent, by their length, the accumulated surface of all patches having the same orientation [Ike83].

Since the discrete surface patches are flat, the resulting EGI will be constituted of impulses whose magnitude depend on the surface area in the objects space: the "needle map".

3.3 Transformation of representations

Since volume and surface based descriptions of objects can appear, we have to discuss two transformations which seem at a first glance to be dual:

- volume based representation to surface based representation
- surface based representation to volume based representation

volume based representation to vision based representation

Usually, this transformation does not imply very much problems. Given a volumetric model, a surface based representation can be computed straightforward. However, there are two problems to be mentioned.

i) Some representations are rather rough approximations of the object (space occupancy representations), this due to the finite size of the basic volumes.

ii) The CSG representation does not have any mechanism which allows to distinguish between surface patches which delimit a base element from the surrounding free space and patches delimiting base elements from another one.

surface based representation to volume based representation

If a complete description of the object's surface is available, the transformation is feasible. Unfortunately, the descriptions of the acquired objects are rarely complete in robot vision.
Therefore, the result of this transformation will be uncertain: a low level representation as the space occupancy representation will know three states ("occupied", "free", but also "unknown"), a high level representation as the CSG will be either impossible (incomplete data) or not unique (decomposition).
Hence, its application is restricted to special cases, where the set of CSG base elements is very small (i.e. one type of element only) and where a rough approximation is sufficient [Phi86]. In either case, this approach is not useable for a precise identification and recognition of objects.

conclusions on the transformation of representations

For the reasons mentioned, the only really feasible transformation is the one from volume based models to vision based models.

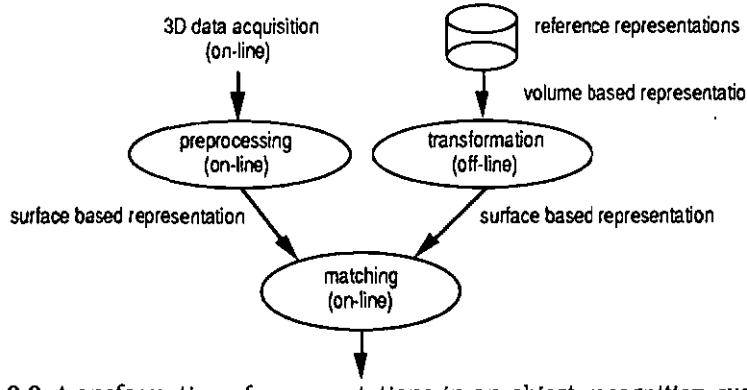


fig 3.3: transformation of representations in an object recognition system

Practically, this transformation will be used to transform once for all volume based reference representations to vision based reference representations, a mean to speed up the recognition process itself (fig 3.1).

3.4 Choice of the Planar Face Representation Graph PFRG

Until now, we have discussed the data to be retained for a high level representation, but not yet the data structure to be used. Therefore, we discuss in the present section first the use of attributed relational graphs for the representation of data, choose a particular high level representation, called PFRG, establishing correspondences between the real world features and the high level representation.

Finally, we compare the representation chosen with the well known EGI's and develop a visual representation of the data retained.

3.4.1 Attributed Relational Graphs ARGs

A high level representation of an object consists essentially of two classes of information:

i) elements (e.g. faces, edges,...) with their attributes (area, orientation,...) and
ii) relationships (e.g. neighbourhoods, ...).

The data structure "attributed relational graph" ARG is very well suited to represent such kind of data: nodes stand for the elements from above, arcs for relationships.
Additionally, we can profit from the well established theory of graphs for the subsequent steps of our recognition task (characterization, matching, particular subgraphs,...).

3.4.2 Planar Face Representation Graph

According to the problem definition of chapter 1, we have chosen the correspondences between object features and graph features as illustrated in figure 3.4, called "Planar Face Representation Graph" PFRG.

3D world	<->	PFRG
(part of an) object	<->	(sub-) graph
face	<->	node
surface area	<->	node attribute "surface area"
surface orientation	<->	node attribute "surface orientation"
neighbourhood relations	<->	arc
border length	<->	arc attribute "border length"

fig 3.4: correspondences between 3D world and the PFRG

The number of features kept for the high level representation is extremely small, the data reduction very high. Nevertheless, this representation allows to realize good object recognition results as we will see later on this report.

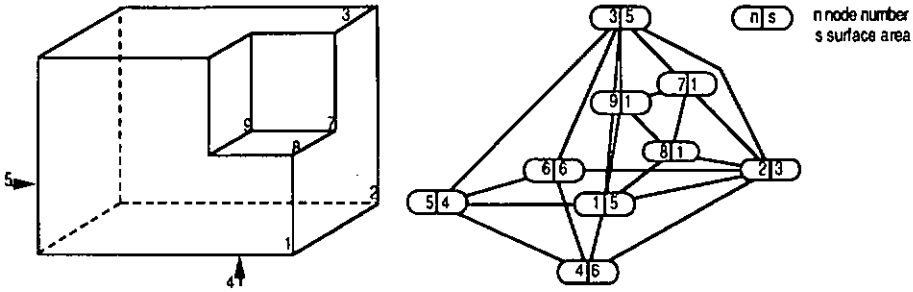


fig 3.5: object "BLOCK" and its PFRG (attributes partially shown)

3.4.3 Correspondences PFRG <-> needle map

Our high level representation PFRG consists of the information contained in a discrete EGI, a "needle map", enhanced by the neighbourhood relations. Figure 3.6 next page shows the correspondences between 3D world features (left), the needle map (middle) and the high level representation PFRG.

3D world	needle map	PFRG
face	needle	node
surface area	needle magnitude	attribute "surface area"
surface orientation	needle orientation	attribute "surface orientation"
neighbourhood-relations		arc
border length		attribute "border length"

fig 3.6: correspondences, 3D world <-> needle map <-> PFRG

For visualization purposes, we introduce an additional representation, the "extended needle map" ENM, an enhanced form of the "needle map".

3.4.3 Extended Needle Map ENM

Beside the needles N_i and N_k which indicate the orientation and, by their magnitude, the surface area of the faces F_i and F_k respectively, we introduce arcs so that neighbouring faces F_i and F_k will have their respective needles N_i and N_k interconnected by an arc $\mathcal{A}(F_i, F_k)$.
This visual representation shows, except the 'border length" attribute which is missing, all attributes of the PFRG.

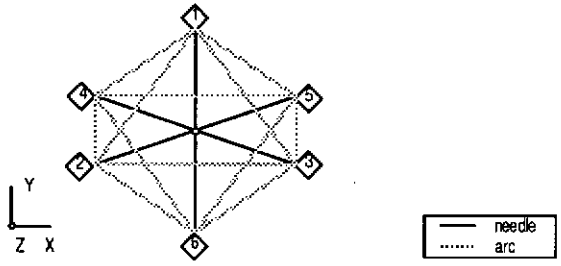


fig 3.7: extended needle map of a convex object (cube)

For non-convex objects, we do not follow exactly the definitions from above, we represent separately parallel needles which are due to separate surfaces of the same orientation. Then we interconnect them with arcs as before.

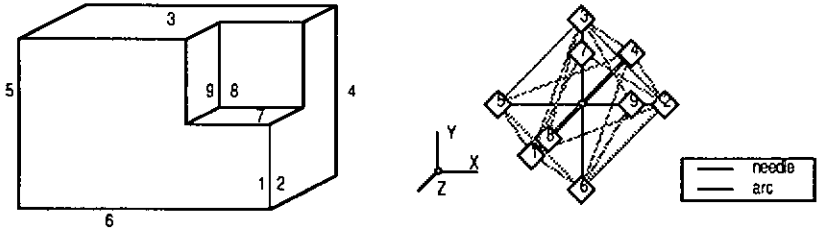


fig 3.8: a non-convex object (left) and its extended needle map (right)

3.5 High level preprocessing

In the previous chapter, we have chosen the PFRG representation for the further work. Unfortunately, due to scene acquisition and low level processing nonidealities, the PFRG representation of a real scene object will by far not be as simple and pure as the reference object PFRG shown before (e.g. fig 3.5).

fig 3.9: from the 3D scene to a prune PFRG representation

In fact, the processing path leading from a 3D scene containing some objects to be recognized to a high level representation passes in our realization by intermediate representations, shown in fig 3.9.

In the following, we discuss shortly the data representations used at the different intermediate levels and the transformations between them, leading from the initial 3D scene data to the pruned PFRG representation used further on. The complete discussion can be found in [Ama90].

3.5.1 Data representations, data conversions

depth map

After data acquisition, the 3D scene data is represented as a discrete depth map (fig 3.10). Due to acquisition nonidealities, for a certain number of pixels (x,y), it will not be possible to determine an associated depth z:

$$d(x,y) = \begin{cases} z & \text{depth known} \\ \text{NIL} & \text{depth unknown} \end{cases}$$

More generally, we can define a pixel as a triplet $(\mathbf{x}_{ij}, i, j)$, containing the data vector $\mathbf{x}_{ij}$ as well as the coordinates (i, j) of its position: a depth map can be described as a set of triplets, forming a generalized image $\mathbf{X}_{mxn}$, a table containing pixels $(\mathbf{x}_{ij}, i, j)$ [Mai90]:

$$\mathbf{X}_{mxn} = \{ (\mathbf{x}_{ij}, i, j) \mid (\mathbf{x}_{ij} \in (\mathbb{R}^n \cup \{NIL\}) \wedge (i,j \in \mathbb{Z})) \}$$

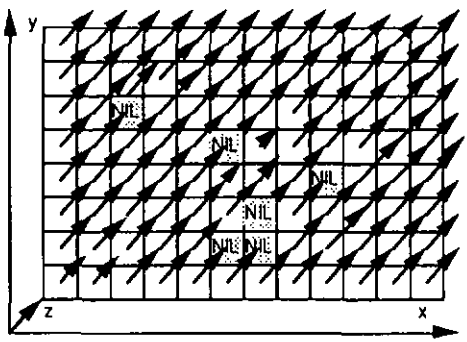


fig 3.10: depth map representation

region / border representation

Next, the depth map is converted into a region / border representation, using some region growing algorithm (chapter 2, [Mai89], [Mai90]). Sets of pixels of the depth map can be established, forming connex regions R which satisfy some homogeneity criterion $\mathbb{H}(R)$, separated by borders B_i.

The original depth map will be partioned and represented as

i) a set of regions $\mathcal{R} = \{..., R_j, ...\}$ and
ii) a set of borders $\mathcal{B} = \{..., B_i, ..\}$, where each border B_i is formed by a set of border segments BS_k : $B_i = \{..., BS_k, ...\}$

In our application, we use the term "region" for homogeneous, connex regions where the homogeneity criterion $\mathbb{H}$ is established on the orientation of the faces the pixels represented (planar patches).

Each region R_j has at least the attributes

i) surface area area(R_j)
ii) surface normal normal(R_j)

as well as some structural information

iii) a non-ordered list of border segments BS_j delimiting the region R_j.

Each border segment BS_j has at least the attribute

i) border segment length

and the structural information

ii) neighbouring region R_a
iii) neighbouring region R_b

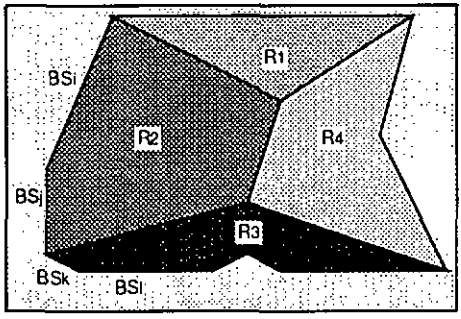


fig 3.11: regions, borders and border segments

Unfortunately, some regions will not be completely known [Mai89]. We will encounter

i) regions with unknown attributes
   - regions created where the underlying depth map elements where NIL
   - regions too small to create useable attributes

ii) a region representing the surfaces outside the boundaries of the original acquired depth map

With the definitions and correspondences established in paragraph 3.4, these nonidealities will propagate to the high level representation PFRG. Therefore, the outcoming representation is generally not a graph in the mathematical sense and will have elements with unknown features:

i) a border with a neighbouring NIL region leads to an arc which has a start-, but no end node,
ii) a region with unknown features leads to nodes with unknown attributes, and
iii) a border between regions R_a and R_b can be broken into several segments.

In the following section, we describe the methods implemented to eliminate these nonidealities.

3.5.2 Preprocessing of the PFRGs

Figure 3.12 shows the sequence of algorithms applied to each PFRG obtained from the region growing step.

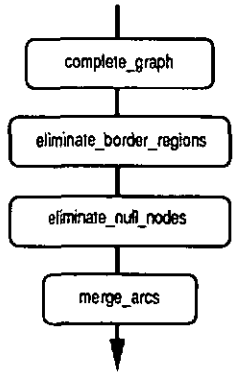


fig 3.12: preprocessing procedures

complete graph

Borders B with just one known neighbouring region R_a would lead to arcs λ with just one known neighbour node N_a. The unknown neighbour region R_b resp. node N_b can be represented by a placeholder node called "supernode SN" (fig 3.13). We use a single supernode SN to represent all unknown regions: it will not be matchable and has no attributes.

fig 3.13: incomplete PFRG (left) and after completition (right)

eliminate border regions

The region growing program used [Mai89] tends to create small, long regions R_{bj} separating adjacent regions R_i and R_k at the edges of the objects (fig 3.14). The attributes of R_b are generally unknown. In order to reconnect the separated regions R_i and R_k by borders B of correct length, we "compress" the border regions R_{bj} until they have a virtual width of zero, merge the border segments and add it to the neighbouring bordersegments BS_{ik} which separate the same two known regions R_i and R_k. This need for an existing border segment BS_{ik} guarantees that no jump edge disappears.

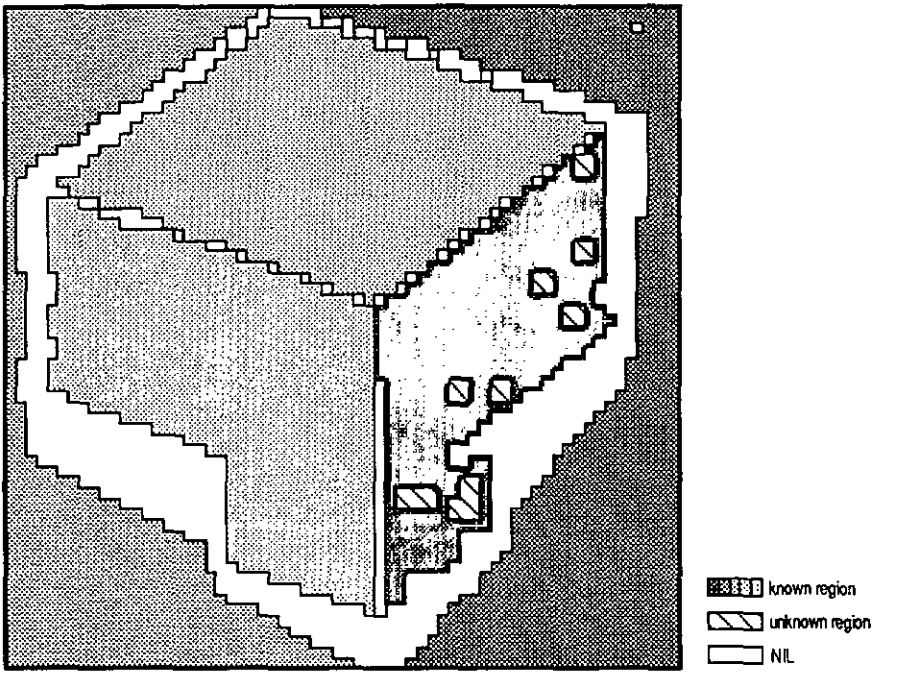


fig 3.14: example "CUBE1" after region growing

Additionally, we limit us to "border regions" having just two completely known neighbouring regions, this in order to avoid ambiguities.

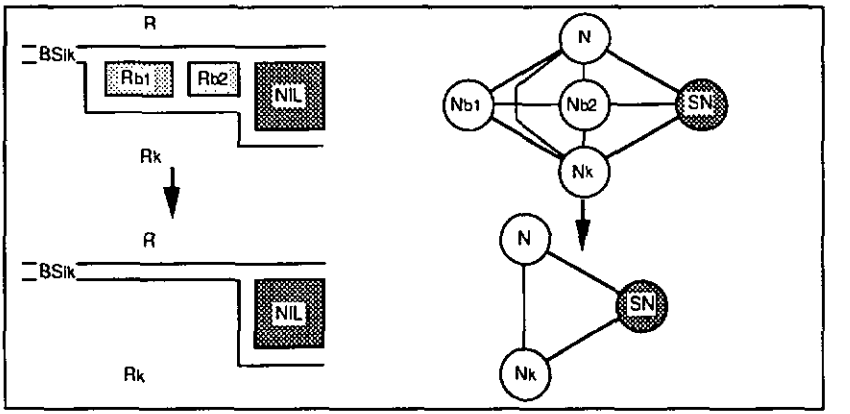


fig 3.15: border regions R_{bl}, before (top) and after transformation (bottom)

Finally, the PFRG can be simplified, deleting the node N_b which stood for the border regions as well as its arcs.

Figure 3.15 shows a typical situation. On top the initial situation in its region/border (left) and PFRG representation (right). Having applied the algorithm discussed, the border regions R_{b1} and R_{b2} disappear and the border segment BS_{lk} between the regions R_l and R_k which already existed before, grows in length.

eliminate null nodes

Nodes N_l which represent regions R_l whose attributes are not sufficiently defined are marked in the original region / border representation. The algorithm eliminates these nodes and the arcs leading to it.

merge arcs

A border B between two adjacent regions R_a and R_b can be formed by a set of border segments. This leads in the non-pruned PFRG to multiple arcs $\mathcal{A}_i$ which link in parallel node N_a, representing region R_a, with node N_b, representing region R_b (fig 3.16, top). The algorithm merges the arcs $\mathcal{A}_i$ while combining the attributes.

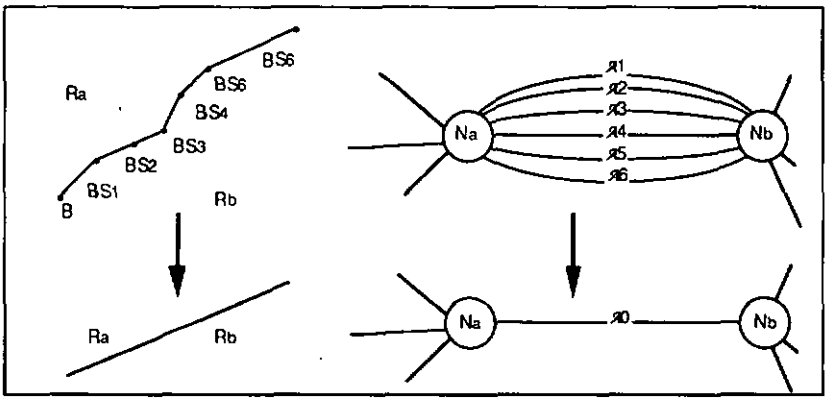


fig 3.16: border segments BS_l (left) and corresponding arcs $\mathcal{A}_i$ (right)

3.5.3 Examples

In the following, we discuss two examples, first "PYRAM2" which allows us to follow the path from the original region border representation to the high level attributed graph representation. The second example "CUBE1" shows in particular the result of the suppression of border regions.

example "PYRAM2"

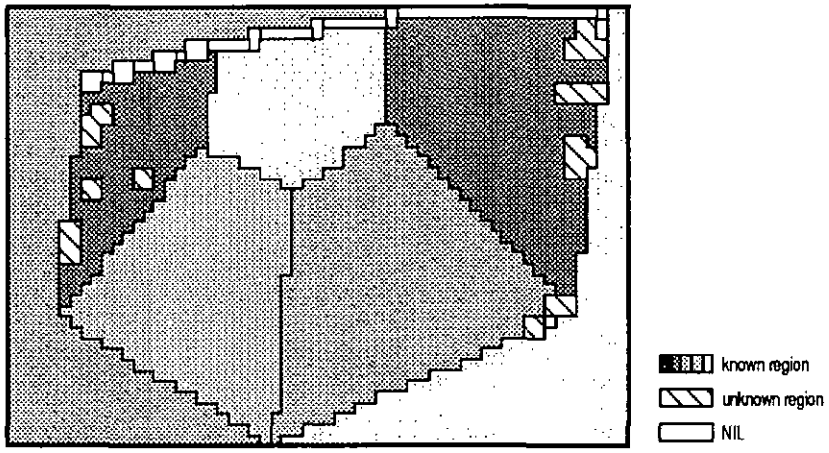


fig 3.17: example "PYRAM2": region/border representation

Figure 3.17 shows the object "PYRAM2" with its regions and borders as they have been extracted from the depth map. The edges of the object (discontinuities in orientation and/or depth) are of particular interest since they lead to regions with unknown attributes as well as to border segmentation. Figure 3.18 bellow shows the extracted PFRG after pruning. the significant regions only appear.

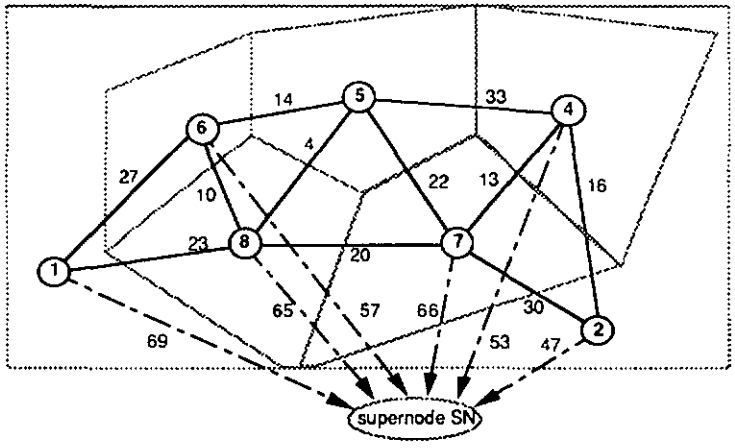


fig 3.18: example "PYRAM2", pruned PFRG

example "CUBE1"

In this second example, we deal with a big number of "border regions", this due to the region growing algorithm (fig 3.13). Thanks to the "border region elimination" method presented before, the outcoming pruned PFRG (fig 3.20) represents very well the situation.

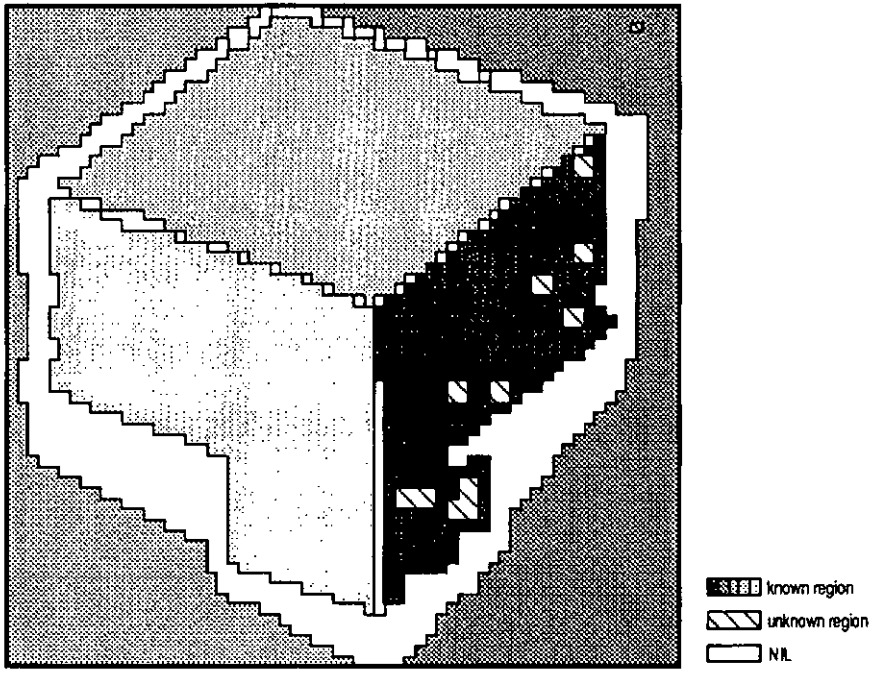


fig 3.19: example "CUBE1": region / border representation

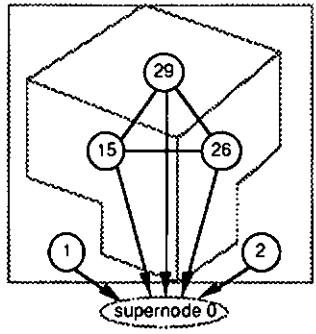


fig 3.20: pruned PFRG for "CUBE1"

Further details on PFRG pruning and some statistics can be found [Ama90].

3.5.4 Conclusions on PFRG pruning

The present algorithm allows to convert the incoming region/border representation to the attributed graph representation needed for the subsequent matching process.
At the same time, some undesirable effects of the previous acquisition and representation conversion steps, that is segmentation and creation of additional, badly defined regions, are eliminated and the PFRG simplified.

A new version of the region growing algorithm, presently under work [Mai90], should allow to reduce significantly the amount of badly defined regions and border segments.

3.6 Conclusions

In the present chapter, we have established first the requirements for a high level representation usable in the domain of object recognition. Second, we discussed some typical examples of representations while taking into account the aspect of the transformation of representations. Next, we introduced the "Planar Face Representation Graph" PFRG, the high level attributed graph representation we use in the subsequent matching, recognition and verification steps. Additionally, a graphical representation of the high level data available has been introduced, the "extended needle map" ENM which is useful as a visual representation of the data and can be used for test purposes.
Finally, we showed in chapter 3.6 some high level preprocessing we apply to the PFRG representation. The aim of this preprocessing is the extraction of a representation which depends very few from low level data partitioning.

4. Model library

4. Model library

Beside the acquired data, we need reference objects and their descriptions at least in the high level representation. For further verifications of the matching results lower level representations will be necessary too. Therefore, we create a library of reference objects, adding eventually supplementary knowledge about the objects represented (fig 4.1).

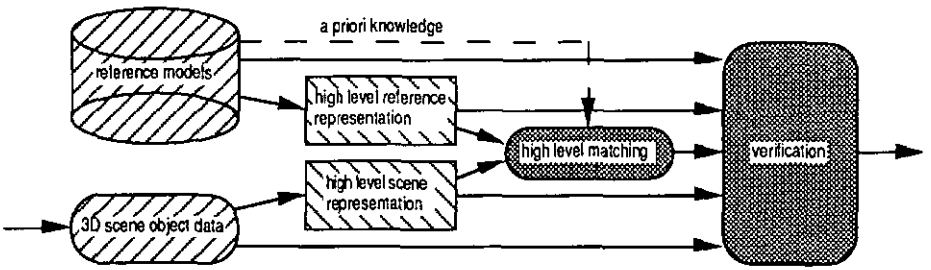


fig 4.1: data streams in a recognition system

4.1 Reference objects

There are different ways to acquire knowledge on reference objects. We discuss the following three approaches (fig 4.2):

- knowledge acquisition by learning
- use of (CAD) data bases
- creation of artificial objects

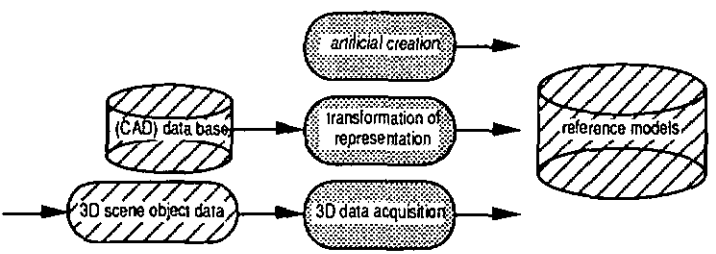


fig 4.2: creation of a reference library

4.1.1 Reference representation acquisition by learning

A very natural way to acquire reference information is acquisition by learning. The reference object is presented to the acquisition device, the data acquired, preprocessed, represented internally and finally, gets attributed a label by the supervisor.
This method, very useful in other domains, has some drawbacks in the case of object recognition:

- the data acquired is rarely complete (partial view, occlusion,...)
- data acquisition, preprocessing and representation tend to misform the data

Therefore, the use of this method is limited to applications where no exact high level description of the object exists. If it should be used as a reference, multiple data acquisitions of a same object have to be combined [Lu88].
The problems concerned with this "reference objects from learning" approach were not subject of our investigations. Therefore, this method will not be discussed.

4.1.2 Using complete, external descriptions of real objects

A second way uses exact, acquisition system independent descriptions of the objects and transforms them to the internal representations of the recognition system. Typically, for objects out of the machinery world, volume based CAD descriptions are available and can be transformed, for test purposes. A supervisor can introduce models at any level of representation, a method we often use.

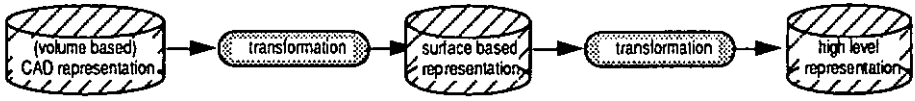


fig 4.3: from (volume based) CAD descriptions to reference models

4.1.3 Using artificially created objects

This third approach has less practical use. Nevertheless, it can be rather interesting for test purposes, allowing the creation of artificial objects. For this project, we developed a set of programs performing this task. The most important one has been written for the creation of cube-based artificial objects, this in a volume based description (space occupancy). The remaining programs permit the transformation of the representations into the high level representation chosen. Details can be found in [Ama88d].

basic idea

Artificial objects of volume V can be created automatically by "glueing" elementary volumes EV to already created objects of volume V-1. For reasons of simplicity, we have chosen cubes of unitary volume, called elementary cubes EC.
A very simple algorithm [Ama86b] allows to create systematically objects from the elementary one (one EV) on to objects of a given maximal volume, including objects with embedded holes. The difficulty resides not in the creation of new objects but, since we are interested to create each object just once, in the test of preexistance.

test of preexistance

We list the tests which are to be executed until a difference occurs in increasing order of complexity:

i) comparison of the size of the box enclosing the objects
ii) comparison of the first and possibily second moments
iii) comparison of the space occupied (including rotations and translations of the objects)

Tests i) and ii) must be done including rotations, test iii) including rotations and translations of the objects. Figure 4.4 next page shows the first 42 objects created in a perspective view of a space occupancy representation. Circles represent elementary spaces occupied while lines indicate adjacencies in the directions parallel to the coordinate axes x, y and z.

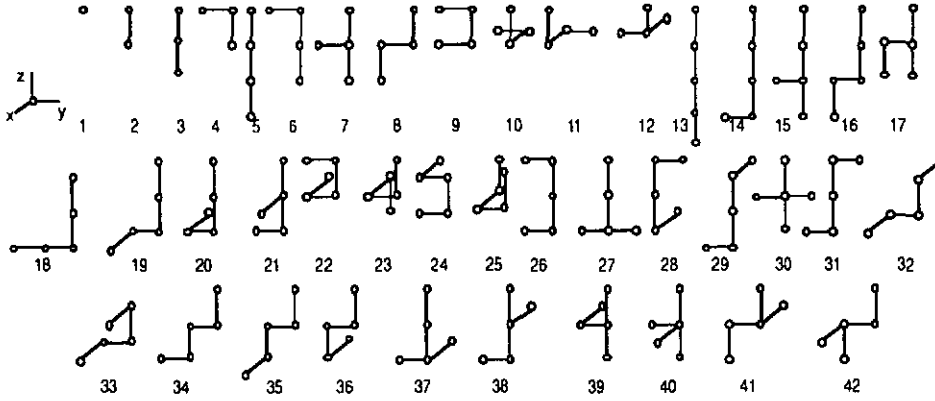


fig 4.4: "skeleton" representation of objects of volume 1 to 5

Figures 4.5 and 4.6 below show the result of the subsequent transformations for objects number 4 and 12. The original space occupancy representation is replaced, first by a surface based representation, then by the high level representation chosen in chapter 3, the "Planar Face Representation Graph" PFRG.

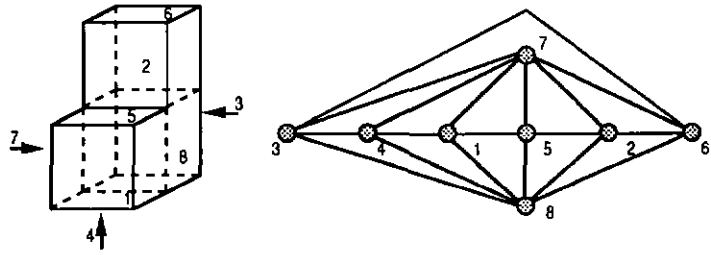


fig 4.5: object number 4 and its PFRG (attributes not shown)

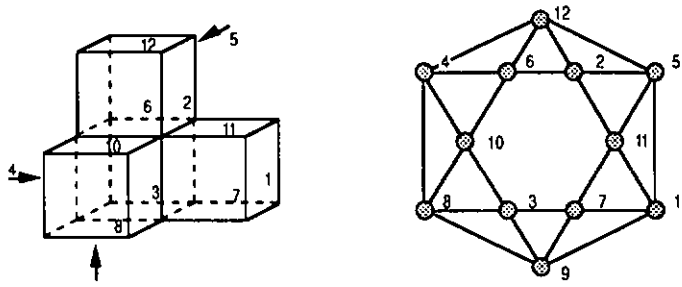


fig 4.6: object number 12 and its PFRG (attributes not shown)

4.2 A priori knowledge

[Shn87], [Ste87] and others have used the fact that off-line preparation of data and knowledge can be used to enhance the performances of an object recognition system. The points investigated cover

i) off-line transformation
The initial knowledge base representation is transformed off-line to a representation easy to use in the recognition task.

ii) redundant data
Preparation of additional redundant information in order to speed up later on-line computing.

iii) exclusion of impossible situations
Use of a priori knowledge for the pruning of "impossible situations".

On a higher level, strategy planning would be interesting too, covering

iv) recognition strategies,
v) strategies for interactive recognition systems, and
vi) strategy planning for object handling.

The transformation of representation has been discussed in paragraph 3 of the present report. Let us just recall that CAD-models, often used as knowledge base in industrial robot application, are not very well suited for object recognition tasks. Usually, the "CAD-Model" has to be transformed first into a "Vision-Model".
Since this kind of enhancements to the general recognition algorithms are heavily application dependent, it is difficult to enter into details. We just touch some points of the list above in order to illustrate the ideas.

exclusion of impossible situations

Until now, we based on the idea that scene objects can be presented to the acquisition device in any orientation. While this assumption will hold for a general 3D scene, it will not for a robot environment. In the system described in [Hor87], profit is taken from the following simple assumptions:

- the objects to recognize are presented to the acquisition device on a flat surface (table, conveyor belt), and
- there is no superposition of objects.

Under these conditions, general objects have only a limited number of stable positions which can be computed in advance. For objects with curved surfaces, an approximation, using objects composed of planar surface patches, can be used. Again, there is the possibility to reduce in a large extent the on-line computation charge since it becomes possible to predict the possible partial views corresponding to the limited number of stable positions.

strategy planning for recognition

Object recognition in an industrial environment is always a race with the time. Therefore, the recognition algorithm should be exactly as complex as needed for a given application. Therefore, it is reasonable to create multi-stage recognition algorithms which need, of course, a supervisor unit. This unit decides at which extent the multi-level recognition algorithm has to be executed. Once the application known, the recognition strategy can be integrated in the supervisior task.

strategy planning for interactive recognition systems

While the above approach uses a fix amount of information at different levels in its recognition task, the present approach has an interactive component. A basic recognition system works straightforward. it uses one set of acquired data to fulfill the recognition task. More complex systems can provide a feedback from the recognition system back to the acquisition device in order to complete the data. So, 3D space where the recognition is up to fail, can be inspected again, possibly from another point of view. The displacement can be done either by the "natural" movement of the objects, i.e. on a conveyor belt, or by commanding a robot's arm which has the acquisition system mounted on.
Again, the planning of strategies how to complete data, can be prepared in advance.

strategy planning for object handling

In the very same manner, the handling strategies, the way how objects are to be grasped, can be planified in advance.They get their importance when unknown or misformed object have to be treated.

4.3 Conclusions

The first approach to create reference objects, knowledge acquisition by learning, turns out to be very difficult. The CAD approach allows to create specific reference objects and is therefore well suited to tasks in the industrial domain (robots, manipulators, ...). The third approach finally, the automatic creation of artificial objects, is essentially useful for test purposes and possibly in scenes where no a priori knowledge on the objects to identify is available, but the choice of an appropriate elementary volume is difficult.
However, we used this "random" approach for the debugging of the matching program implemented.

For the tests shown later on in this report, the supervisor approach has been used, introducing data directly in the reference data base. The more complicated tasks concerned with a knowledge base of higher complexity are not subject of the present work.

5. Representation matching

5. Representation Matching

Until now, we discussed the two main branches of an object recognition system. The first one is the scene data branch, constituted of data acquisition, preprocessing and high level representation. The second one is concerned with the reference data in the model library (fig 5.1).
In the present chapter, we merge the two branches in the representation matching task, the most important one on the path towards an object recognition system.

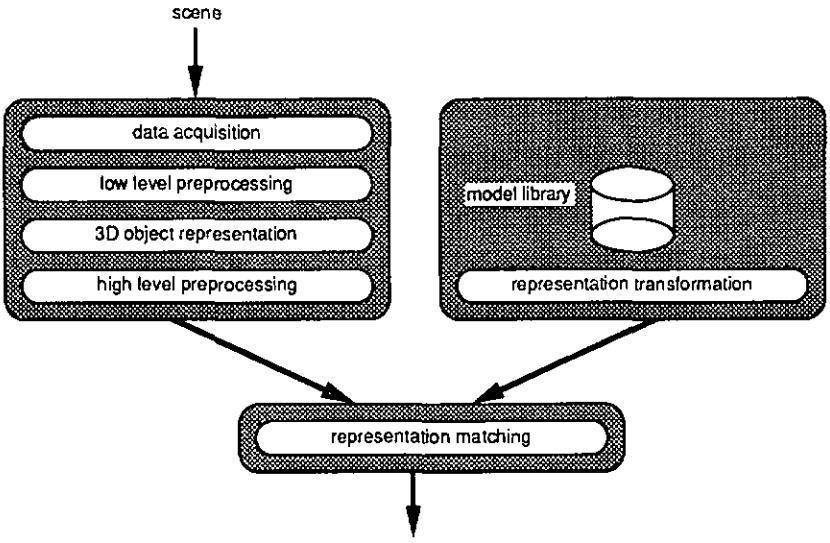


fig 5.1: an object recognition system

First, we dress the problem in a general way, discussing some aspects which are due to the fact that natural scene data will be used.

Second, we give some definitions, covering some important terms out of the graph theory ("subgraph", "isomorphism", ...) and formalize the mathematical representation of graphs.

Next, we treat the problem of isomorphic subgraph-graph matching (§5.3), first as a problem related exclusively with the topological aspects of the graphs, later extended to attributed graphs.

In paragraph 5.4, the algorithms are extended to the more general case of matches between subgraphs of any connexe graph.

The nature of the scene data acquired is such that exact matching has a too limited scope. Therefore, in paragraph 5.5, inexact matching tactics will be discussed with a special interest for a promising method, finally realized and described in chapter 7 of this report.

The chapter terminates with some conclusions.

5.1 General problem definition

The aim of a 3D object recognition system is to get answers to one or more of the following questions.

i) What is in the 3D scene ?	recognition of the object
ii) Where is it ?	localization of the object
iii) What is its orientation ?	orientation of the object
iv) What is the reliability of the result ?	measure of confidence

Naturally, there are a lot of crosslinks between the answers to these questions, e.g. if the orientation of an object is unknown, it will be difficult to match it with a given reference object and vice versa.

data representation

In the following, we will use object representations in the form of attributed relational graphs (ARG). The PFRG representation of chapter 3 is a particular case of ARGs.

completeness of data

A reference graph R represents an object of the model library, supposed to be completely known. The scene graph T in turn, the representation of the scene object to be recognized, is very often i) an incomplete description since only partial views are available and ii) inaccurate, since data acquisition and data extraction will never be free from errors. Therefore, our problem consists of the matching of partial, "incomplete" and perhaps inaccurate scene (sub-)graphs with a set of reference graphs R_i.

5.2 Terms and definitions

There are at least two ways to represent a graph G: i) "enumeration" in the form of sets of nodes and arcs, and "adjacency matrices". Obviously, they can be used in parallel and be converted one to the other. We will use both descriptions, selecting for each paragraph the more convenient one.

enumeration of the elements forming a graph G

$G = \{N_g, A_g\}$ where n_i = node i

$N_g = \{n_1, \ldots, n_i, \ldots, n_\kappa\}$ i = node index

$A_g = \{a_1, \ldots, a_\mu\}$ κ = number of nodes

$a_l = (n_i, n_j)$ μ = number of ares

A graph G is composed of a set N_g of nodes and a set A_g of arcs where an arc $a_l(n_i, n_j)$ is defined as a connection between two nodes n_i and n_j of graph G.

description of a graph G by its adjacency matrix $\underline{G}$

The topology of a graph can also be described by a matrix $\underline{G}$ where each matrix element $g_{ij} = 1$ stands for an existing arc, $g_{ij} = 0$ for a nonexisting arc between the i^{th} and the j^{th} node of graph G.

$$\underline{G} = [\, g_{ij} \,] \qquad g_{ij} \in \{0,1\} \qquad i, j = 1 \ldots \kappa,\ \kappa = \text{number of nodes}$$

definitions [Even79]

graph isomorphism

Two graphs $T = (V_T, E_T)$ and $R = (V_R, E_R)$ are said to be isomorphic if there are 1-1 correspondences $f: V_T \rightarrow V_R$ and $g: E_T \rightarrow E_R$ so that for each edge e, linking vertices u and v in T there is an edge $g(e)$ in R which links $f(u)$ to $f(v)$ [Even79].

subgraph

A graph $T (V_T, E_T)$ is called subgraph of graph $R (V_R, E_R)$ if both V_T and E_T are subsets of V_R and E_R respectively.

isomorphic subgraph

Given a subgraph $R' (V_R', E_R')$ of R such that E_R' contains the complete set of edges linking the vertices of V_R'.
Graph $T (V_T, E_T)$ is said to be an isomorphic subgraph of $R (V_R, E_R)$ if T and R' are isomorphic.

nodedegree

The nodedegree ND of a node n of graph A of size α is defined as the number of arcs connecting it to its neighbor nodes:

$$ND(n,A) = \sum a_{nj} \qquad a_{nj} \in \{0,1\} \qquad j = 1, .., n-1, n+1, .., \alpha$$

5.3 Isomorphic graph/subgraph matching of relational graphs

problem definition 1

> Given a connexe testgraph T of size τ and a connexe reference graph R of size ρ, bigger or equal to τ.
> Find the isomorphic (sub-)graph matches between T and R resp. the subgraph R' of R.

Temporarily, we ask for subgraph isomorphism between the (sub-)graphs T and R' brought into correspondence where the graphs are non-attributed, non-directed and have simple arcs. We will extend the problem to subgraph-subgraph matches and to inexact matching later.

5.3.1 Graph and subgraph isomorphism test

An easy way to check for isomorphism between graphs T and R is to use the adjacency matrices $\underline{T}$ and $\underline{R}$ for the test of identity. Since the order of rows and columns in the adjacency matrices is free (depending on the manner how the nodes have been labeled) and the number of nodes can be different in both graphs, we use first the hypothesis on node-node correspondences to relabel (reorder) and eventually reduce in size one of the adjacency matrices: e.g. each label of graph $\underline{R}$ is replaced by the corresponding label of the graph T, creating the relabeled matrix R_L' (fig 5.2). Now, the test for identity can be done between matrices $\underline{T}$ and $\underline{R}_L'$.

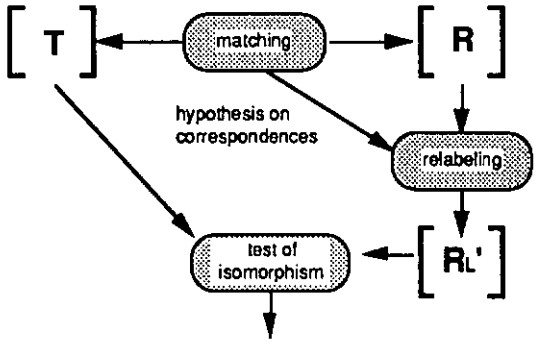


fig 5.2: isomorphism test: hypothesis, relabeling and test

Using a $(\kappa \cdot \kappa)$ matrix $\underline{M}_i$ for the representation for correspondences of hypothesis H_i, we formalize the relabeling by

$$\underline{R}'_L = \underline{M}_i \cdot (\underline{M}_i \cdot \underline{R})^T \qquad \text{where } \kappa = \min(\tau, \rho)$$

While the inner term exchanges columns, the outer one, after transposition, exchanges the rows. For the case of directed graphs, the resulting $\underline{R}_L$ matrix must be transposed since the adjacency matrices are no more symmetric. Next, we apply the tests for isomorphisms, subgraphs ... .

subgraph check

$$\forall\ (T[i,j] = 1) \Rightarrow (T[i,j] = R_L'[i,j]) \qquad (1 \le i \le \tau,\ 1 \le j \le \rho)$$

Graph T exists as a (sub-)graph R' in graph R if for each edge e_T in graph T, there exists a corresponding edge e_R in R', but not necessarily vice versa.

subgraph isomorphism

$$\forall(\underline{T}[i,j]) \Rightarrow (\underline{T}[i,j] = \underline{R}_L'[i,j]) \qquad (1 \le i \le \tau,\ 1 \le j \le \rho)$$

Graph T exists as an isomorphic (sub-)graph R' in graph R if for each edge e_T in graph T, there exists a corresponding edge e_R in R' and and for each edge e_R of R', there exists a corresponding edge e_T in T.

The problem is now to find all the correspondence matrices $\underline{M}_i$ which stand for an isomorphism between the graphs T and R:

find $\underline{M}_i$ so that $\underline{T} \equiv \underline{R}_L' = \underline{M}_i \cdot (\underline{M}_i \cdot \underline{R})^T$

We will discuss in the next section two methods for this correspondence search problem.

5.3.2 Correspondence search

There are basically two different approaches when looking for sets of correspondences (matches). The first one is an "exhaustive search" method, the second one an approach where partial solutions are to be extended ("backtrack").

5.3.2.1 Exhaustive correspondence search

Exhaustive search methods create systematically all combinations of correspondences between nodes of both graphs.
According to Ullmann [Ull76], we implemented the exhaustive search in the form of a search tree (fig 5.3) [Ama88d]. Each state S_i of level d ($0 < d \leq \tau$) in the tree stands for a hypothesis H_i on d node-node correspondences. At each transition from a state S_i to a son-state S_k, an additional correspondence (r_d, t_x) between nodes r_d and t_x of reference graph R and scene graph T is established, extending hypothesis H_i to H_k. The number of branches leading to states of the next lower level is (ρ-d) where ρ is the number of nodes of the (bigger) reference graph R.

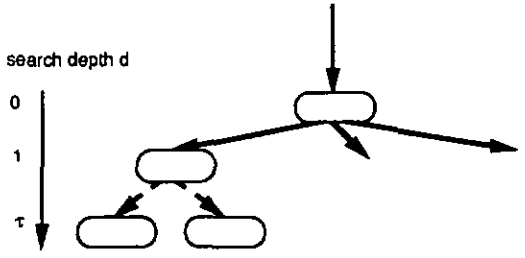


fig 5.3: search tree

The disadvantage of the exhaustive search is the huge number of operations to be executed. In fact, the number of searchtree states to pass through is

$$O(\rho,\tau) = \frac{\tau!}{(\tau-\rho)!}$$

where ρ, τ = number of test and reference nodes respectively

To reduce the computational cost of the exhaustive search method, search tree pruning criterions are introduced, still based on topological aspects. Before we pass to the "backtrack" methods, we discuss them shortly in the next section. For details consult [Ull76], [Ama88d] and chapter 6.

5.3.2.2 Structure based search tree pruning criterions

nodedegree condition

The first condition, the nodedegree condition, uses very few computation time, but can exclude a huge number of correspondences using basic structural information. Given a test graph T of size τ and a reference graph R of size ρ, the condition is:

$ND(t_i,T) = ND(r_j,R) \quad r_j \in R,\ t_j \in T$ for graphs of identical size ($\tau = \rho$)

$ND(t_i,T) \leq ND(r_j,R) \quad r_j \in R,\ t_j \in T$ for graphs of different sizes ($\tau \leq \rho$)

In the second case, we assume that missing arcs in graph T lead to non-existing nodes (e.g. invisible regions in the underlying data). The criterion takes account of one single correspondence. Therefore, it can be applied before entering into the search tree, excluding a priori many of the potential correspondences.

Next, a family of criterions considers the neighborhoods of the nodes to bring into correspondence. The first one has been proposed in [Ull76] under the name of "structural refinement", the second one is the usual neighborhood condition.

structural refinement (forward checking neighborhood condition)

Using a priori knowledge (e.g. nodedegree condition) or establishing a node-node correspondence limits the possibilities for future matches. The present method checks, after each extension of the search tree, if one or several of the lower level states can be excluded while guaranteeing that no set of correspondences standing for an isomorphic solution will be lost.

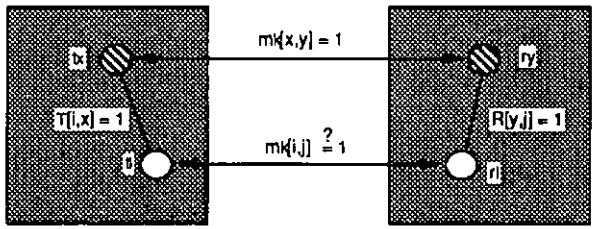


fig 5.4: structural refinement

The idea is the following (fig 5.4):

> Given a correspondence matrix $\underline{M}_k$ at search tree depth d.
> Check if the candidate correspondence (t_i, r_j) could appear in a final hypothesis for a subgraph match.

If there is no node t_x in graph T adjacent to t_i ($\underline{T}[i,x] = 1$) that has been matched ($x \leq d$, $m_k[x,y]=1$) or will eventually be matched ($x > d$, $m_k[x,y]=1$) with a node r_y of graph R adjacent to r_j ($\underline{R}[y,j] = 1$), we exclude the correspondence (t_i, r_j), $m_k[i,j]$ can be set to 0.

The process is recursive: if a correspondence (t_i, r_j) is excluded due to the condition, all other candidate correspondences have to be checked again until no more modifications occur. This routine substitutes also the test for the presence of identical subgraphs, but not the test for isomorphisms (see [Ull76], [Ama88d]).

backchecking neighborhood condition

Having established a correspondence at search tree depth d, the number of candidate correspondences which could be establish at depth (d+1) can be reduced using a neighborhood condition.

> If there is a node t_x in graph T adjacent to t_i ($T[i,x] = 1$, $i = d+1$) that has been brought into correspondence ($x \leq d$) with node r_y of graph R ($m_k[x,y]=1$) which is not adjacent to r_m ($R[y,m] = 0$), then we can exclude the candidate correspondence, $m_k[d+1,m]$ becomes 0 (see fig 5.5, next page).

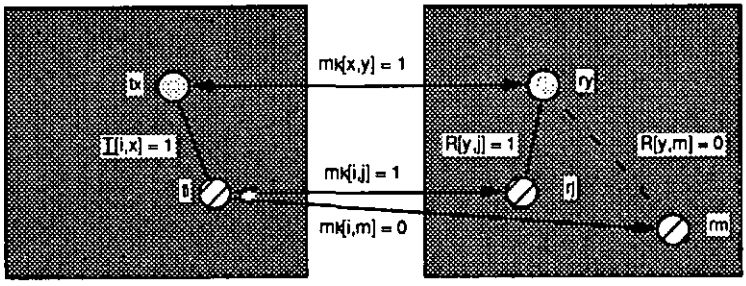


fig 5.5: neighborhood condition (backward checking)

Again, the routine replaces the test for identical subgraphs, but not the one for isomorphisms.

5.3.2.3 Correspondence search by "backtracking"

A second class of methods is based on "backtracking". Once a partial solution is found, it is used as a base for the extension to a bigger one.

Since a "backtrack" algorithm will be used and discussed in detail in chapter 7, "inexact matching", we limit us to a short introduction.

node sets

For our application, we establish the following sets for both reference and test graphs (fig 5.6):

> "candidate" node sets T_n, R_n
> Sets of nodes not brought yet into correspondence.

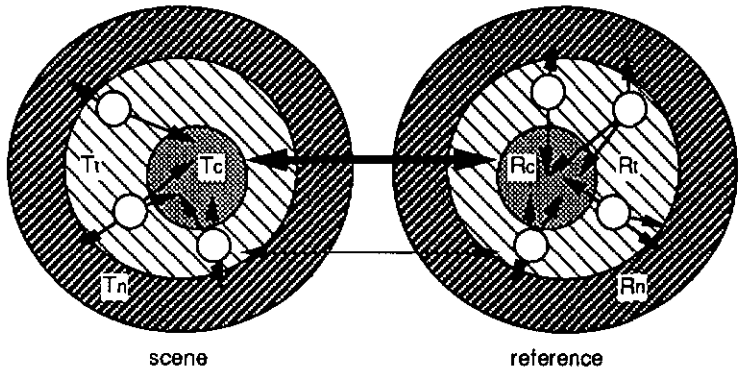


fig 5.6: node sets for the "backtrack" approach

"core" node sets T_c, R_c
Sets of nodes brought into correspondence. The arcs lead exclusively to nther nodes of the same core node set.

"terminal" node sets T_t, R_t
Sets of nodes brought into correspondence having at least one arc which leads to the "candidate" node set.

algorithm

Next, we apply the "backtrack" algorithm according to [Esh84], [Esh86].

i) Bring a first couple of nodes t_i, r_j into correspondence ($\tau \rho$ possibilities)
t_i and r_j change from the candidate node set to the respective terminal node set

ii) Take a scene terminal nodes t_i and find a neighbornode t_k which is element of the candidate node set T_n. Add t_k to the terminal set T_t.

iii) Take a reference terminal nodes r_i and find a neighbornode r_k which is element of the candidate node set R_n. Add r_k to the terminal set R_t.

iv) If scene node t_k and reference node r_k fulfill some matching conditions, establish the correspondence. else return to iii) for another candidate reference node r_l.

v) If by this operation either t_k or r_k has lost its last non-matched neighbor node, transfer t_k resp. r_k to the respective core node set T_c or R_c.

vi) continue with step ii)

By this algorithm, attempts are made to add simultaneously scene respective reference nodes to the already reconstructed subgraphs.
The choice of the condition for the addition of new nodes decides on the final result. Basically, exact isomorphic matching as well as inexact matching is possible as long as the graphs are connexe.

computational cost

The worst case for this algorithms can be described:

> All the N nodes of a graph are linked with all the other nodes of the same graph. For this case the number of significant operations grows with $O(n) = (n!)^2$. This huge number is done to the fact, irrealistic for scene interpretation, that each terminal node of a subgraph of size m has (n-m) candidate nodes and this, of course, in both reference and test graphs.

For a more realistic estimation, let us consider a case with a mean value of the nodedegree of k:

> In this case, the number of candidate nodes for each terminal node is much smaller, the computational complexity grows with
>
> $$O(n,k) = (n-1) \cdot (k-1)^2$$

By using structure and attribute based criterions, the computational complexity can be reduced further. This is in particular true for an intelligent choice of startpoints for the extensions, called "root-matches", a point discussed later on in this chapter.

"Backtrack" approaches are described in [Esh86], [Esh84], [Ber73], [Cor70], [Bit75], [Ama90] and chapter 7 of this work where additional, specific pruning criterions are presented.

5.3.3 Other approaches

The subgraph matching problem appears regularly in the domain of graph theory. Evidentially, there exist a lot of other approaches to resolve it. Here just two ideas.
[Kno68] proposes a graph decomposition, [Ber73] a special grammar, the "K-formulas". In the context of inexact subgraph-subgraph matching, the propositions in [Cor70] and [Fen88], concerning the extraction of particular classes of subgraphs (cycles, cliques, ...), usable as startpoints for subsequent expansions with backtrack procedures, can be interesting.

5.3.4 Extension to attributed graphs

Until now, only structural information has been considered, but, as the result of data acquisition of real scenes, attributed graphs are available (e.g. PFRGs). In the present section we discuss how this supplementary information can be used i) to reduce the complexity of the search tree and ii) to check the reliability of the graph matching results.

We can distinguish two classes of attributes: i) node attributes and ii) arc attributes.

node attributes

Under node attributes, we understand attributes related to single nodes. Therefore, they can be used for dissimilarity measures independently.

For example, the dissimilarity between scene node t_i and reference node r_j can be computed as the weighted sum over the dissimilarities with respect to the a_n different node attributes.

$$d_n(t_i,r_j) = \sum_{k=1}^{a_n} \alpha_k \, d(t_i, r_j, k)$$

where α_k = weighting factor, k = attribute index

arc attributes

Under arc attributes, we understand attributes related to arcs or depending on the adjacent nodes (e.g. node attribute "orientation" allows to extract the arc attribute "angle"). Exactly in the same way as before, we can introduce a global arc attribute dissimilarity, e.g.

$$d_a(\mathcal{A}(t_i, t_j), \mathcal{A}(r_l, r_m)) = \sum_{k=1}^{a_a} \beta_k \, d(\mathcal{A}(t_i, t_j), \mathcal{A}(r_l, r_m), k)$$

combination of arc- and nodeattribute dissimilarity measures

Finally, both attribute dissimilarities can be defined by some application dependent law, e.g.

$$d(T,R) = \gamma \sum d_n(t_i,r_j) + (1-\gamma) \sum d_a(\mathcal{A}(t_i, t_j), \mathcal{A}(r_l, r_m))$$

where γ = weighting factor

The term above stands for the accumulated dissimilarities over the matched (sub-)graph elements and depends, therefore, on the number of elements matched. For the comparison of matches of different size, an interesting feature when looking for "promising" subsolutions, we will use a "cost" function instead (chapter 7).

comments

The dissimilarity measures are useful at different levels:

i) combined or separately in similarity criterions for pruned search,
ii) combined for the determination of "promising submatches" where further search is worthwhile and
iii) for the classification of the hypotheses on matches and their reliability.

Their use is application dependent, letting the user a huge degree of freedom. We mention just some of the most important open questions when applying attribute dissimilarity measures:

- How to measure the attribute dissimilarity (relative, absolute, ...) ?
- How to combine the partial results (weighting factors, ...) ?
- How to handle with missing attributes ?
- How to compare dissimilarities of matches of different size ?

Unfortunately, there are no general solutions and special solutions have to be investigated for each application. For further discussion consult [Sha85], [Esh85.1], [San83], [Won85], for an example see paragraph 7.7 .

5.3.5 Limits of the algorithm

With the actual implementation of the algorithm, we can resolve the following problems for both attributed and non-attributed, non-directed graphs:

i) test of isomorphism between graphs of the same size
ii) subgraph check between graphs of different sizes
iii) test of subgraph isomorphism between graphs of different sizes

In turn, we cannot resolve the more complicated problems listed bellow:

iv) subgraph check between a subgraph of A and a subgraph of B
v) isomorphism check between non-attributed and attributed non-directed subgraphs
vi) all inexact matching

5.4 Extension to isomorphic subgraph-subgraph matching

Until now, we made the assumption that all the nodes of the smaller graph T could be matched with nodes of a subgraph of the bigger graph R (graph-subgraph match). For representations of natural data, this assumption is irrealistic.

We eliminate now this condition, introducing matching between subgraphs of both graphs.

problem definition 2

Given a connexe testgraph T of size τ and a connexe reference graph R of size ρ.
Find isomorphic subgraph matches between the subgraph T' of T and the subgraph R' of R.

Still, the graphs are non-directed and have simple arcs, their attributes will be used later.

5.4.1 Subgraph Matching

For the correspondence search problem, we have presented two types of solution (§5.3.2):

i) "exhaustive search" with subsequent tests
ii) "backtrack" procedures, extending partial matches to bigger ones

It is easy to see that for the subgraph/subgraph isomorphism problem, the second family of algorithms fits much better. Once a first match established, the following attempts to match are limited to the neighborhood of the nodes already matched, this on both graphs.

One major problem is the choice of the first couple of nodes to bring into correspondence; either we try them exhaustively or we apply an algorithm for the extraction of particular subgraph matches, the "root matches".

5.4.2 Root matches

Root matches are subgraph matches based on particular information out of the respective graphs. The particularity can be i) the structure of the underlying graphs, ii) the attributes or iii) both. The aim of the root match is to create for search algorithms favoring "promising partial solutions" good and reliable initial subgraph matches. Subsequently, they are extended using the "backtrack" approach.

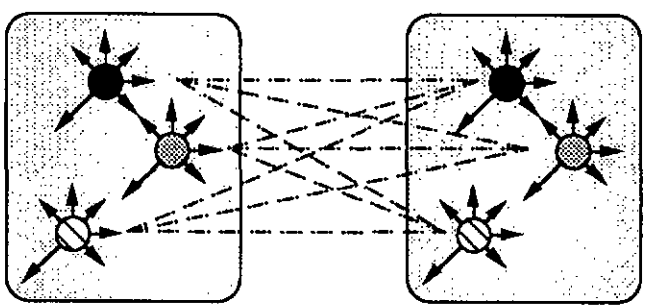


fig 5.7: root matches and subsequent extensions

structure based root matches

For a limited number of typical graphs, algorithms exist in the theory of graphs which allow to extract them rapidly (cycles, cliques, ...) [Bron73]. An application has been proposed in [Fen88].

structure and attribute based root matches

We will limit ourselves to the discussion of one interesting example, Oshima's "kernels" [Osh87]. He proposes the extraction of an "important" couple of adjacent nodes in the ARG representation of the scene object, called "kernel", which allows to establish a reliable first match between elements of scene and model representations. The criterions for the regions

representated are i) surface area, ii) type of area (planar, curved,...), iii) not occluded, iv) number of adjacent regions, etc... .
Once the data-driven "kernel" (root-) match established, the algorithm becomes model-driven (fig 5.8). Based on the elements adjacent to the model-"kernel", the backtrack algorithm attempts to find the corresponding elements in the scene representation.

Why this distinction "data-driven"/"model-driven"? Only surfaces which have really been acquired can be used as "kernels". Therefore, the algorithm is based on the data extracted from the scene. For the extension of a partial match, the situation is slightly different. While the model driven approach guarantees that we are looking for additional parts of the same model, a scene driven approach could, eventually, not distinguish between scene data from different scene objects, the approach is also intended to scenes with multiple objects.

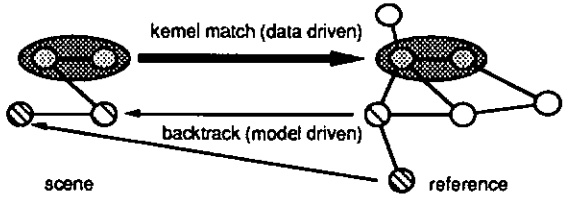


fig 5.8: "kernel" match and "backtrack" extension

5.4.3 Conclusion on partial matches

In order to match partial graphs efficiently, criterions on a "root"-match can be very helpful. They allow to limit in an important way the number of branches in a correspondence search-tree. The criterions are based on the attributes of the elements represented in the graphs and, evidentially, will be weighted depending on their importance (size, exclusivity of the attribute, reliability,...).
Having discussed the possibility to realize partial matches of graphs (subgraph/subgraph isomorphism) we have now a tool we can extend for inexact matching.

5.5 Inexact matching

The algorithms discussed are well suited for applications where we have to treat with exact graph matching. In real scenes, unfortunately, we will have nonidealities in the path which leads from the original scene data to the high level representation, this despite of the preprocessing steps proposed in paragraphs 2.5 and 3.5 .
Let us just recall the most important reasons.

i) data acquisition (illumination, resolution of the input device, ...)
ii) input data segmentation (§2)
iii) data smoothing (e.g. surface approximation)
iv) data reduction (high level representation)

With this few ideas in mind, we can see that even for identical natural scenes we cannot be sure to obtain the same high level representation. Hence, the graph matching algorithm should have some flexibility concerning the structural aspects. In fact, the term "inexact" is applied in the context of structural aspects only, the attributal part has been considered as inexact from the beginning, resulting in dissimilarities.
We discuss the problems for our choice of a high level representation, the PFRGs.

The meaning of "inexact matching" not being defined yet, we could interpret it in one of the following ways.

i) Establishment of a set of exact, partial and compatible matches, covering only partially the original scene representation.

ii) An extension of point i), accepting non-exact extensions of the matches once all exact matches done (fig 5.9)

iii) Matching using weak neighborhood rules from the beginning, punishing insertion / deletion of graph elements by an incremented dissimilarity [Esh84], [Esh86].

iv) Modification of the ARGs by split and merge of nodes and data so that exact matching becomes possible.

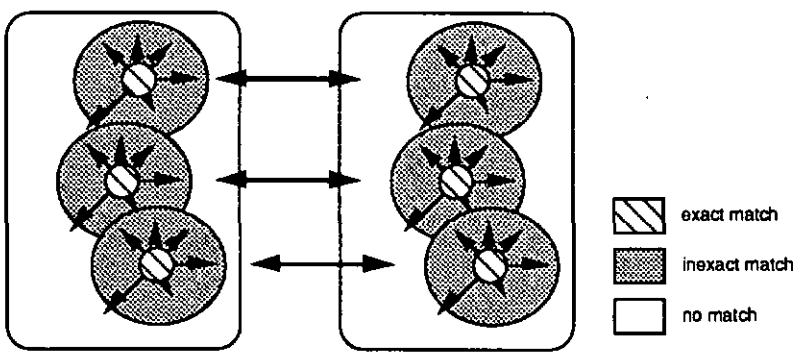


fig 5.9: compatible, exact partial matches with subsequent inexact extensions

Compatible exact subgraphs

Point i) and ii) are relatively restrictive interpretations. In terms of the original exact subgraph/subgraph match, they can be interpreted as the combination of compatible, partial exact matches. The compatibility essentially concerns the nodes involved in the partial matches, eventually also some attribute based informations like as Faugera's "rigidity constraint", described in §7.4 [Fau87]. Nevertheless, the method can be used for the establishment of hypotheses if there is a strong verification [Gmu89].

inexact subgraph-subgraph matching

The third point is new, leading to a certain freedom in the extension of the subgraph matches. To introduce the subject, we give some ideas used later on in the realization of chapter 7.

Basically, we will use the backtrack algorithm discussed in §5.3.2.3, applying a single condition for an extension of a partial match: the candidate nodes must be connexe to the respective matched subgraphs.

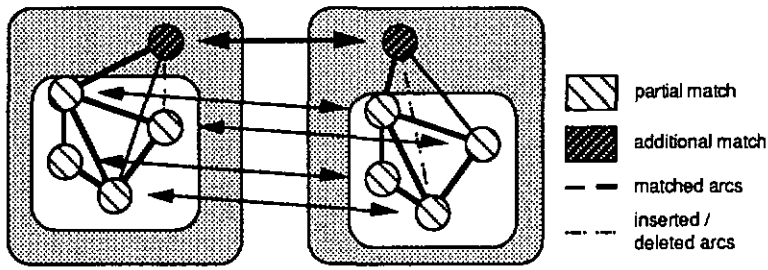


fig 5.10: missing and additional arcs

Using this "node correspondence" based mechanism, missing and additional nodes cannot be detected as such and their correspondences contribute normally to the global dissimilarity measure. In turn, additional resp. missing arcs can be detected. A comparison of the graph structures allows to determine differences in the structure of the matched (sub-)graphs. Attributing to each correspondence between an additional resp. missing arc and an existing arc a relatively high dissimilarity value, we favore matches having no or few inexactely matched elements.

modification of the ARGs by split and merge of nodes and data

While extending a partial subgraph/subgraph match to a bigger one, attempts could be made to merge nodes, either in the scene ARG or in the model ARG in order to obtain similar attributes for the candidate graph elements in both graphs (fig 5.11). Since reference graphs are supposed to be more robust, scene graphs have to undergo this process.

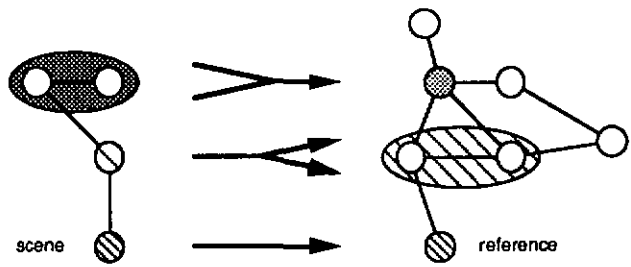


fig 5.11: inexact matching by node merging

For an application, it will be difficult or even impossible to find a split and

merge method. The list bellow gives some ideas on the potential problems.

i) Where to stop the search for candidate nodes for an eventual split/merge ?
ii) How to define criterions for a split/merge ?
iii) How to handle with the possible iterative behavior of the algorithm?

conclusions on inexact matching

Inexact matching is a difficult, but very important subject in the domain of object recognition. While the points i) and ii) of the discussion here above are just extensions to the rigid matching techniques, iv) seems to go too far and will be very difficult to realize. Therefore, we favore the proposition iii) which allows an implementation in a straightforward manner for both exact and inexact matching, treating inexactitudes over an enhanced dissimilarity. The limitation to connexe graphs can be overgone later by using some combination methods to combine compatible partial matches.

5.6 Conclusions

Having discussed data acquisition, preprocessing, high level representation and model libraries before, the present chapter has been devoted to the representation matching task.

In increasing order of complexity, exact subgraph-graph matching for non-attributed and attributed graphs has been discussed, passing then to the more general case of exact subgraph-subgraph matching.

A short glance on the original scene input data and to the subsequent steps (acquisition, approximation, segmentation, high level representation) explains that exact matching is of limited scope. Therefore, in §5.5, the idea of inexact matching has been introduced, accompanied by a short collection of ideas for its interpretation.

The following chapter 6 illustrates the use of the simpler "exact matching" approach, allowing to collect some experience in the domain. Chapter 7 in turn documents in detail the more important "inexact matching" method implemented for the final realization.

6. First experiences using an exact matching method

6. First experiences using an exact matching method

Exact matching is a first, limitative approach to the more general graph matching problem. We have already discussed some drawbacks and limits of this approach. Nevertheless, we realized a package of programs, called ISOMOR and documented in [Ama88d]. It allowed to acquire some experience for the case of exact matching before we passed to the more complicated inexact matching, discussed in chapter 7.

In the first paragraph (§ 6.1), we recall the problem definition and the algorithm used for the establishment of hypotheses on correspondences between nodes of both scene and reference graphs. It is, for the structural aspects, essentially the one we presented before in paragraph 5.3 [Ull76].

According to the discussion of paragraph 5.5 ("inexact matching"), the aim of the implementation of the exact matching method was not a 3D object recognition system, but its use as a base for some experiments with high level representations.
We treat the exact graph matching problem in two stages: first, on the topological level, matching non-attributed graphs, a pure mathematical problem, second, introducing some heuristics based on the attributes of both arcs and nodes (§ 6.2).

The chapter terminates with some conclusions on exact matching, denumbering its drawbacks and limitations which lead us finally i) to the high level representation PFRG (§ 3.4) and ii) to the inexact matching algorithm of chapter 7.

6.1 Problem definition, algorithm

In the following, we use the symbols defined in the preceding chapter (§ 5.2) and algorithm proposed in [Ull76] which has been described in paragraph 5.3. The problem is limited to exact subgraph-graph matching.

> Given a connexe testgraph T of size τ and a connexe reference graph R of size ρ, $\tau \leq \rho$.
> find the isomorphic (sub-)graph matches between T and R, resp. the subgraph R' of R.

The method implemented consists of two passes:

i) a search for hypotheses on matches, initially exhaustive, later pruned applying some criterions on structure and attributes, and

ii) the test of the hypotheses with the criterions "subgraph match" and "isomorphic subgraph match".

Since the algorithm has been discussed in the preceding chapter (§ 5.3), including an extension to attributed graphs, we pass directly to the experiences we made using a software package called ISOMOR [Ama88d].

6.2 Reference objects and high level representation

high level object representation

For the following experiences, we use a high level representation, intended to be used for simple, planar faced objects (fig 6.1).

3D world	<->	high level representation
face	<->	node
surface area	<->	node attribute "surface area"
neighborhood relations	<->	arc
angle between adjacent faces	<->	arc attribute "angle" (+ convexe, - concave)

fig 6.1: correspondences between the real world and a first high level representation

reference objects

Figures 6.2 to 6.6 show the reference objects used (left) and the respective high level representation (right).

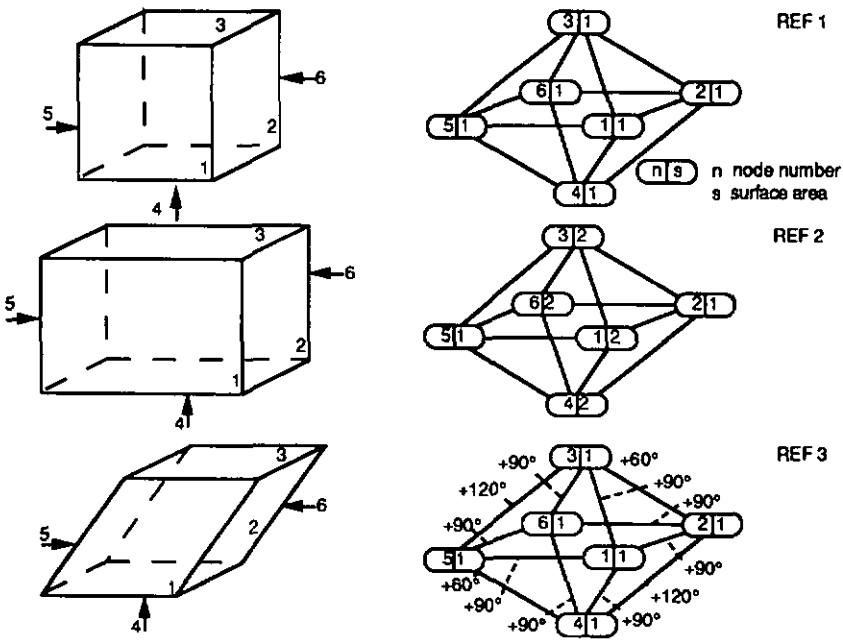


fig 6.2-4: reference graphs "Ref 1" (top) to "Ref 3" (bottom).
(angles not shown = + 90°)

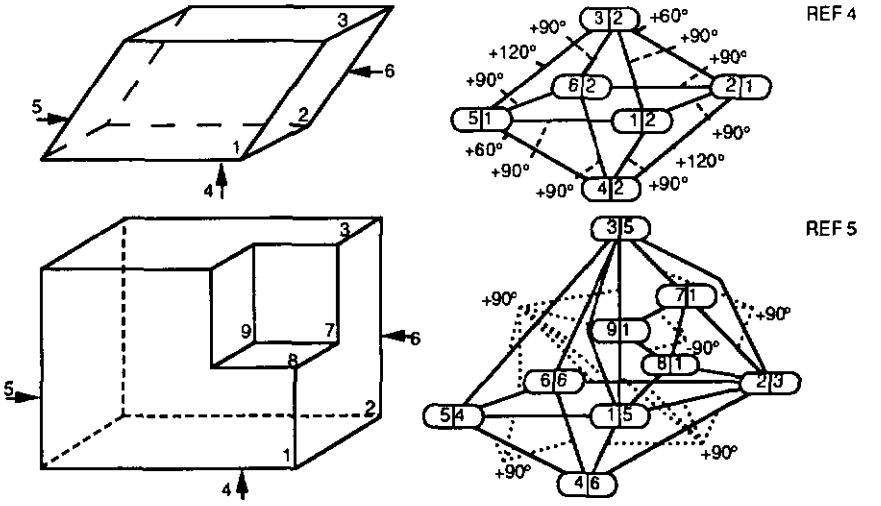


fig 6.5-6: reference graphs "Ref 4" (top) and "Ref 5" (bottom)
(angles not shown = + 90°)

6.3 Experiences and examples

Beyond a general introduction to the subject these experiences have been done with the aim to find answers to the following questions:

- What information is necessary for a good representation of simple planar-faces objects?
- What are the difficulties when passing from a surface based, real world description of an object to the purely mathematical form of an attributed graph?
- Are there *a priori* conventions when a human observer "matches" objects and, if yes, how to implement in a recognition system ?

The experiences have been made before the final PFRG representation (§3.4) has been chosen.

6.3.1 Symmetries and isomorphisms

For a first test, we use the non-attributed subgraph of the test graph "Test 0" (fig 6.7).

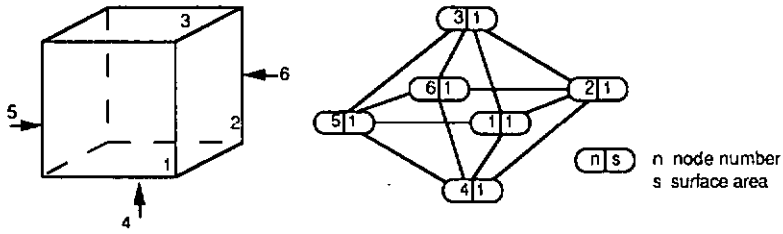


fig 6.7: test graph "Test 0"

Depending on the number of nodes in the testgraph, we determine the number of isomorphisms found with the method proposed.

test nb	test nodes involved	reference graph				
		ref 1	ref 2	ref 3	ref 4	ref 5
1	1	6	6	6	6	9
2	1...2	24	24	24	24	42
3	1...3	48	48	48	48	90
4	1...4	48	48	48	48	108
5	1...5	48	48	48	48	96
6	1...6	48	48	48	48	96

table 6.8: number of isomorphisms for non-attributed (sub-) graphs

Since only structural data has been submitted, it is not surprising that the number of isomorphisms found are very similar for all the reference objects. Nevertheless, these numbers are surprisingly high. A first explanation is the regularity of the structures, but a second point needs some discussion.

We examine the problem again, matching a test graph "Test 1", a subgraph of "Test 0", and the reference graph "Ref 1", both non-attributed.

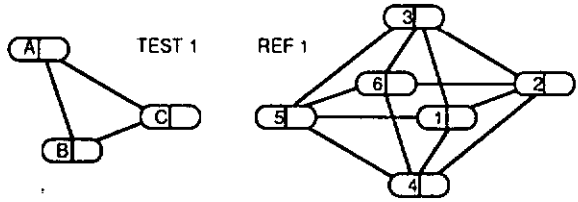


fig 6.9: non-attributed test- and reference-graphs

There are eight subgraphs in the reference graph which are isomorphic to the test graph. They correspond to triplets of adjacent faces, standing for triplets of regions enclosing each a vertex of the original object. Selecting arbitrarily the subgraph of "Ref 1" containing nodes (1 2 3), we have three possibilities to fix a first correspondence (e.g. A-1). Then, there are still two

possibilities left for the rest, combining either (B-2) and (C-3) or (B-3) and (C-2), we get 48 possible solutions.

On the original object's level, we examine the problem again. There are eight ways to match the vertex of the test object surrounded by the surfaces (A B C) with the eight vertices of the reference object (fig 6.10) and three ways to establish a first surface match (e.g. (A-1)).

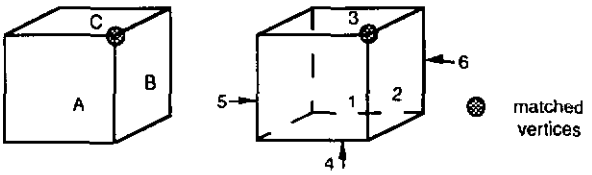


fig 6.10: test- and reference-objects

This correspondence established, the rest of the matches is given, the number of possible isomorphism is 24.

deduction 1

The difference between the two approaches, the graph based approach used by the algorithm implemented in ISOMOR and the object surface based approach used by a human observer, lies in the quantity of information used. While the graph based approach uses only structural information, the object based approach uses implicit conventions: a human observer distinguishes between "inside" and "outside" of an object. Then, he takes into account the order (clockwise, counterclockwise) the faces have, with respect to the vertex enclosed.
Applying the arc attribute helps to resolve the "inside/outside" ambiguity, but not the one concerning the order, the representation chosen does not allow to resolve the problem.

"mirror objects"

Under "mirror objects" we understand symmetric objects having both the same high level representation. Again, the only attribute potentially useable to distinguish such objects is the one concerned with the relative orientation of surfaces, in our choice of § 6.2, the arc attribute "angle".

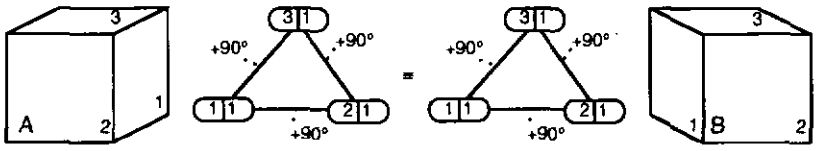


fig 6.11: partial views of symmetrically labeled cubes A and B and the corresponding identical graphs

Unfortunately, it is not helpful either in avoiding "mirror object" solutions, e.g. cubes A and B of figure 6.11 have identical high level representations.

deduction 2

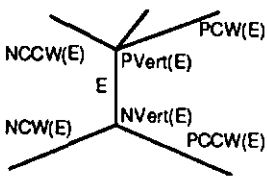
Again, we need a richer high level representation to avoid matchings with "mirror objects".

discussion

Following the results of the above experiences, we enhance the data of the high level representation. A first approach has been made, using the Baumgart's rich and redundant "winged edge representation" [Bau72], [Bal72], [Ama88d].

Baumgart's "winged edge representation"

The surface of a real object can, at least locally, be projected to a planar surface. Therefore, it becomes possible to enumerate and order the edges leaving a vertex. Figure 6.12 shows an example. Each vertex E is attributed with a list of important neighbor edges. For the set of edges forming an object, we get a redundant description of the neighborhoods.



PVert(E)	previous vertex with respect to E
NVert(E)	next vertex with respect to E
PCW(E)	previous edge clockwise with respect to E
NCW(E)	next edge clockwise with respect to E
PCCW(E)	previous edge counterclockwise with respect to E
NCCW(E)	next edge counterclockwise with respect to E

fig 6.12: an edge E and its attributes in the "winged edge representation" [Bau72]

For our particular choice of a high level representation, the assumption of "local planarity" of the graphs is also true. Using an additional convention ("outside view"), we can use Baumgart's approach and have a solution to the two problems mentioned.
While the rich and redundant information is useful for a rapid search tree pruning, there are important drawbacks:

i) the representation "enumerates" the sequence of the edges to appear (difficult to extend to inexact matching)
ii) inconsistencies in the case of partial representations (occlusions and shading)
iii) the representation is too complicated
iv) the representation needs a lot of space (redundancy)

surface orientation

Finally, an other, very natural solution has been chosen: the arc attribute "angle" is replaced by the surface attribute "orientation", or more precisely by the information contained in the outward pointing normal vector of the respective surface.

This new attribute has a larger scope, can be used in a wider extent than for the distinction of the cases discussed before and is clearly extractable from raw input data.

6.3.2 Other experiences

neighborhood relationships

Basically, the definition of a "neighborhood relationship" is rather evident. Nevertheless, there can be effects of subgraph matchings which are surprising. Here an example.

Using the program ISOMOR, we attempted to bring the attributed test graph "TEST 1" into correspondence with the "visible part" of reference graph "REF 5".

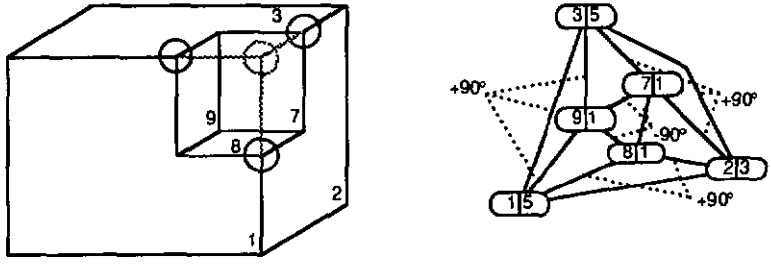


fig 6.13: a partial view of an object and its corresponding graph

The number of isomorphism was 4, not 3 as we thought initially. This is due to the fact that faces 1,2 and 3 of the original object form a virtual vertex (shaded circle), not distinguishable from the real objects vertices, except if well chosen attributes and dissimilarity measures are used.

background

Until now, we discussed matches between subgraphs representing "partial views" of the scene objects and graphs representing entirely the reference objects.
Naturally, in a real scene, the background will also appear as one or more regions in the segmented input data and, therefore, as one or more nodes in the high level representation.

Hence, the additional problem of the distinction between "object nodes" and "background nodes" will appear. Two approaches for a solution are imaginable:

i) a priori knowledge
A scene acquisition system can acquire an "empty scene", determining later the differences in input data when an object is presented. This solution is feasible in a well defined environment which changes slowly or not at all.

ii) on line separation
Using appropriate hypotheses on the objects (e.g. rigidity [Fau87]) or model driven correspondence search (e.g. [Osh87]), it becomes possible to separate scene objects and, therefore, also the background from the scene.

The first approach is simpler to realize. Still, there are some questions concerning shaded or nonreachable regions of the background when scene objects are presented. The second approach is more general and will be used in the realization of chapter 7.

6.4 Conclusions

The program ISOMOR has been written with one main goal: to get some experience in the domain of graph-matching. The goal has been achieved.

Beyond the experiences presented above, it allowed us to get some "feeling" for the problems of graph matching, difficult to document but also very important.
We dress just a small and incomplete list of domains on which the experiences had an influence.

i) graph attribute dissimilarities
   - selection of the attributes
   - computation rules
   - combination of dissimilarities providing from different attributes
   - determination of weighting factors

ii) selection of the matching approach
   - selection of the inexact matching approach
   - need for object separation

iii) high level representation
   - choice of the high level representation PFRG

Taking advantage of the experiences made and the "feeling" collected, we pass in the following chapter 7 to the realization of an inexact subgraph-subgraph matching method.

7. Inexact matching of high level representations

7. Inexact matching of high level representations

In the present part, we treat the task of inexact high level matching. Given two high level object representations in the form of attributed graphs (PFRGs), we attempt to find the most promising (partial) match so that we can extract a hypothesis on the scene object, a similarity measure as well as a indication which rotation the reference object has to undergo so that it maps best to the scene data (fig 7.1).

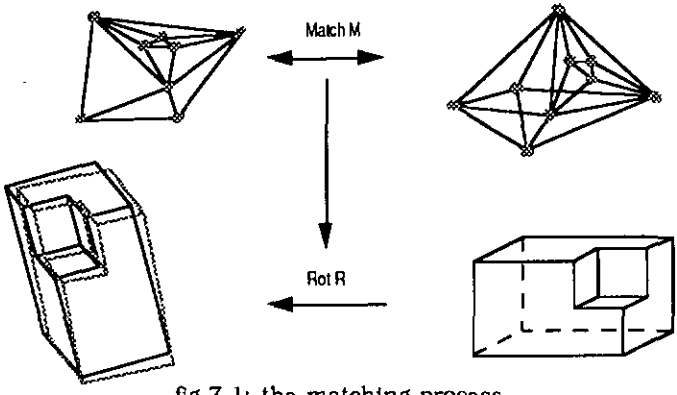


fig 7.1: the matching process

Since we will use a graph matching technique for connexe graphs which is based on some heuristics, we have to put the general graph matching problem in context with our object recognition environment. In chapter 7.1, we recall our representation of the natural data in the form of an attributed graph.

Chapter 7.2 is devoted to the subject of graph matching as a tree search problem. Once the basic terms defined, we formalize the meaning of a search tree state as well as the algorithms to apply when creating sons from a given parent state. A separate paragraph is devoted to the tree expansion tactic itself, an important point when a time-effective search has to be done.

In chapter 7.3, we focus on the tree pruning conditions which are guided by both structural and attributal aspects. Next, the computation rules for the determination of dissimilarities between subgraphs of both reference and scene graph are defined, as well as a "cost" which assembles the different dissimilarities in a single measure for subgraph/subgraph matches, a number which allows to guide the expansion of the search tree.

Having put the bases for the expansion of a pruned search tree as well as the computation rules for values characterizing the quality of the matches, we define in chapter 7.5 some global tree expansion stopping conditions.

Next, we extract the solutions. We specify the conditions which define a leaf state of the search tree as the end point of a path from the root state which includes a solution: a set of matches established, a similarity measure and a hypothesis on the rotation to apply to the reference object in order to map it on the scene object.
Some results and the conclusions terminate this part.

7.1 Bases, Problem definition

7.1.1 3-D objects and their mathematical representation in the form of a graph

We recall the correspondences between the region border representation obtained from the previous data acquisition / region growing steps and our high level representation PFRG as it was defined earlier.

Planar Face Representation Graph PFRG

region	≡ node	attributes:	surface area of the region normal vector of the region
border	≡ arc	attribute:	border length

table 7.2: correspondence region/border representation <-> PFRG

One or several 3D objects are represented in the form of an attributed relational graph where a node stands for a region or face (supposed to be approximatively planar), attributed with a surface area and the normal vector, and an arc stands for the relationship between adjacent regions or faces, attributed with the length of the border common to the regions involved (fig 7.3).

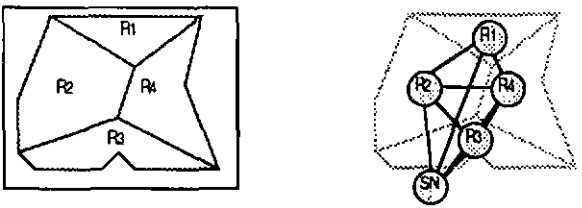


fig 7.3: regions, borders, and the corresponding PFRG

We use the following notations to represent scene and reference graphs:

$G_S = (N_S, \mathcal{A}_S)$	scene graph
$G_R = (N_R, \mathcal{A}_R)$	reference graph
N_{Sn}	node n of the scene graph
N_{Rm}	node m of the reference graph
$\mathcal{A}(N_{Sm}, N_{Sn})$	non-directed arc interconnecting adjacent scene nodes N_{Sm} and N_{Sn}
$\mathcal{A}(N_{Rm}, N_{Rn})$	non-directed arc interconnecting adjacent reference nodes N_{Rm} and N_{Rn}

reference graph

A reference PFRG is an attributed graph representation of a reference object, supposed to represent all faces of an object and to contain reliable informations.

scene grsph

A scene PFRG is an attributed graph representation of a scene, having undergone data acquisition and preprocessing, and supposed to be incomplete and to hold less reliable information.

7.1.2 Problem definition: inexact matching

We can describe the inexact attributed subgraph matching problem as follows.

problem

Given two planar face representation graphs (PFRGs) G_S and G_R. Find the "best" set of scene node/reference node correspondences following a "cost" function.

hypothesis

We establish only 1-1 node correspondences.

solution

A solution Sol_i can be defined as a record containing

- M_i — the set of nodes brought into correspondence $\{\ldots,(NC_{Ri}, NC_{Sj}),\ldots\}$
- $cost_{Soli}$ — the global $cost_{Soli}$

and, as we will see in chapter 7.2, additional information

- Rot_{Soli} — the best fitting rotation
- others — (i.e. missing/spurious arcs)

cost function

A cost function takes into account all information on the dissimilarity of subgraphs matched. It allows to determine a "best" match if more than one solution is available.

discussion

i) Since we establish only node/node correspondences and their respective dissimilarities. Therefore, missing nodes cannot be detected and cannot be considered in the global cost-function.

ii) Arc/arc correspondences will be established implicitly where possible (inexact matching), participating, therefore, in the global cost function. A comparison of the graph structures will allow us to determine differences in structure of matched (sub-)graphs. This information (missing/spurious arcs) can be used for the global "matching cost", too.

iii) Beside the local arc/arc and node/node dissimilarities, there exist also more general dissimilarity functions based on sets of correspondences (i.e. rotation fitting error).

In the following chapters, we will discuss our implementation in detail.

7.2 A state space lattice for the search of the best match

In order to find a "best" match between reference graph G_R and scene graph G_S, we apply a search algorithm in a state space lattice. This allows us to explore all possible sets of correspondences between reference and scene nodes.

state space lattice

Evidentially, the state space lattice is particular in the following sense:

i) Each state S of the state space lattice corresponds to a set of 1-1 node correspondences

ii) We will use a "backtrack" correspondence search approach (§5): a partial solution will be extended to a bigger one.
Therefore, the extension of the set of correspondences of size n of state N creates a path to state M with its set of correspondences of size n+1. The states can hence be classified according to the size of their set of correspondences.
Generally, a state can be extended in multiple ways, creating more than one "son state". A graphical representation of the state space lattice leads therefore to a pyramid, where the top corresponds to a state with an empty set of correspondences, the "root state".

iii) Inherently, the "backtrack" approach provides different ways to extend a set of correspondences of size n to a set of size m ($m \geq n+1$).
Therefore, different paths could lead from the root state to a given state in the lattice. If each extension of a set of correspondences is preceded by a test of preexistence (§7.3, table 7.6), the state space lattice becomes a search tree.

tree search

In the present chapter, we show first some terms which appear regularly in the context of tree searching. Then, we discuss the informations contained in each state of the search tree. Since a non-restricted expansion of the search tree would lead to an amount of computation we cannot afford, we introduce a first search tree expansion condition, the connectivity condition. This allows us to formalize the process of expansion for a general state as well as for the root state of the search tree.

The last part of this paragraph is devoted to the question of the order of expansion in the search tree.

7.2.1 Terms

search tree	rooted graph in which two nodes have at most one path between them
root state S_{root}	topmost state of the search tree (no ancestors, level 0)
leaf state	state without descendants
parent state	state of level n, giving rise to the creation of a son state of level n+1
son state	state of level n+1, created from a parent state of level n
state S	general state having as well preceding as subsequent states
path	A path through a tree connects a sequence of states. It can be represented as an ordered list that records the nodes in the order they occur in the path.
level	search space level, path length of the path from the root state to the actual state

7.2.2 A state in the search space

Having defined the most important terms, we describe now the representation of our matching problem in the search tree.

match

A state S_k corresponds to the match of a reference subgraph SG_R of size l with a scene subgraph SG_S, of the same size. This match is represented in our realization by

core nodes

- the reference core node set $C_{Rk} = \{..., N_{Ri}, ...\}$
  Set of l matched nodes N_{Ri}, forming the node set of reference subgraph SG_R.

- the scene core node set $C_{Sk} = \{..., N_{Sj}, ...\}$
  Set of l matched nodes N_{Sj}, forming the node set of scene subgraph SG_S.

- the set of l couples of corresponding core nodes $M_k = \{..., (N_{Ri}, N_{Sj}), ...\}$

At each state, we associate two additional node sets (fig 7.4, next page):

terminal nodes

- the reference terminal node set $T_{Rk} = \{..., N_{Rm}, ...\} = 1\ N_{Rk} \cdot C_{Rk}$

  $(\forall N_{Rm})\ \exists\ (\mathcal{A}(N_{Rm}, N_{Rn}))$ where $N_{Rn} \in C_{Rk}$, $N_{Rm} \in T_{Rk}$

  Subset of nodes N_{Rm} of the reference graph, where each element N_{Rm} is adjacent to at least one element of the core node set C_{Rk}.

- the scene terminal node set $T_{Sk} = \{..., N_{Sn}, ...\}$

  $(\forall N_{Sm})\ \exists\ (\mathcal{A}(N_{Sm}, N_{Sn}))$ where $N_{Sn} \in C_{Sk}$, $N_{Sm} \in T_{Sk}$

  Subset of nodes N_{Sn} of the scene graph, where each element N_{Sn} is adjacent to at least one element of the core node set C_{Sk}.

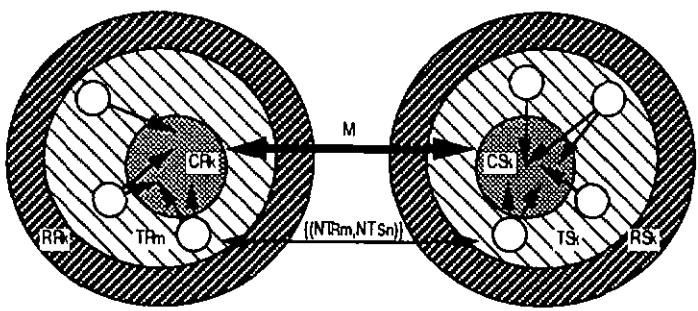


fig 7.4: reference node sets (left) and scene node sets (right)

Additionally, we can define two more sets which will be used implicitly:

rest nodes

- the reference rest node set $R_{Rk} = \{..., N_{Rp}, ...\}$
  Subset of nodes N_{Rp} of the reference graph where no N_{Rp} is element of neither C_{Rk} nor T_{Rk}.

- the scene rest node set $R_{Sk} = \{..., N_{Sq}, ...\}$
  Subset of nodes N_{Sq} of the scene graph where no N_{Sq} is element of neither C_{Sk} nor T_{Sk}.

The following attributes belong to each state S_k:

- search level
  Size of the core node sets C_{Rk} and C_{Sk}, corresponding to the size of the matched subgraphs SG_{Rk} and SG_{Sk} of nodes matched.

- number of matched arcs AM_k
  Number of arcs put into correspondence for the matched subgraphs SG_{Rk} and SG_{Sk}.

$$AM_k(S_k) = \sum \Delta AM_{ik}$$

  where ΔAM_{ik} stands for the number of additional matched arcs due to the transition from parent state S_i to son state S_k.

- number of inserted arcs AI_k
  Number of arcs inserted in the scene graph in order to get isomorphicly matched subgraphs.

$$AI_k(S_k) = \sum \Delta AI_{ik}$$

  where ΔAI_{ik} stands for the number of inserted arcs due to the transition from parent state S_i to son state S_k.

- number of deleted arcs AD_k
  Number of arcs deleted from the scene graph in order to get isomorphicly matched subgraphs.

$$AD_k(S_k) = \sum \Delta AD_{ik}$$

  where ΔAD_{ik} stands for the number of deleted arcs due to the transition from parent state S_i to son state S_k.

- cost of the subgraph match $cost(S_k)$
  Global measure of dissimilarity for the subgraphs SG_{Rk} and SG_{Sk} matched.

7.2.3 Expansion of the search tree

We discuss now the search space expansion algorithm implemented, and establish the connectivity condition for the expansion of a state S_i of level l.

connectivity condition

Given the core node sets C_{Ri} and C_{Si} of size l, standing for a match M_i. Are candidate nodes for an expansion of the match M_i to match M_k of level l+1 all couples of terminal nodes (NT_{Rm}, NT_{Sn}) of the candidate match set CMS_i:

$$CMS_i = \{ (NT_{Rm}, NT_{Sn}) \mid (NT_{Rm} \in T_{Ri}) \wedge (NT_{Sn} \in T_{Si}) \}$$

NT_{Rm} and NT_{Sn} have not been used yet in the match of state S_i and are adjacent to at least one core node of the respective sets.

$$CMS_i = \{(NT_{Rm}, NT_{Sn}) \mid (NT_{Rm} \notin C_{Ri}) \wedge (NT_{Sn} \notin C_{Si}) \wedge (\mathcal{A}(NT_{Rm}, NC_{Ri})) \wedge (\mathcal{A}(NT_{Sm}, NC_{Si}))\}$$

where $N_{Ri} \in C_{Ri}$, $NC_{Si} \in C_{Si}$

7.2.3.1 Expanding a general state

For each state S_i, except the root state S_{root}, we can apply the following expansion rule:

For each element of the set $CMS_i = \{ (NT_{Rn}, NT_{Sm}) \mid (NT_{Rn} \in T_{Ri}) \wedge (NT_{Sm} \in T_{Si}) \}$, we add a new state S_k if some expansion conditions are fulfilled.

For the new state S_k, we create the node sets defined before and compute the attributes mentioned before.

update core node sets

The candidate terminal nodes are added to the respective core node sets

- reference core node set $C_{Rk} = C_{Ri} \cup \{ NT_{Rn} \}$
- scene core node set $C_{Sk} = C_{Si} \cup \{ NT_{Sm} \}$

The candidate terminal nodes NT_{Rn} and NT_{Sm} are eliminated from the terminal node sets. Nodes N_R respectively N_S of the rest node sets R_R and R_S which are adjacent to the candidate terminal nodes are transferred from the rest node sets R_R and R_S to the terminal node sets T_R and T_S respectively.

update terminal node sets

- reference terminal node set
$$T_{Rk} = T_{Ri} - \{NT_{Rn}\} \cup \{N_R \mid \mathcal{A}(N_R, NT_{Rn})\}, \; N_R \in R_R$$
- scene terminal node set
$$T_{Sk} = T_{Si} - \{NT_{Sm}\} \cup \{N_S \mid \mathcal{A}(N_S, NT_{Sm})\}, \; N_S \in R_S$$

update rest node sets

- reference rest node set
$$R_{Rk} = R_{Ri} - \{N_R \mid \mathcal{A}(N_R, NT_{Rn})\}, \; N_R \in R_R$$
- scene rest node set
$$R_{Sk} = R_{Si} - \{N_S \mid \mathcal{A}(N_S, NT_{Sm})\}, \; N_S \in R_S$$

The new match (NT_{Rn}, NT_{Sm}) is added to the set of matches M_i, the rest of the attributes updated and the new cost of the match M_k computed.

- set of couples of matched core nodes
  $M_k = M_i \cup \{ (NT_{Rn}, NT_{Sm}) \}$
- search level
  $level(S_k) = level(S_i) + 1$
- number of arcs matched
  $AM(S_k) = AM(S_i) + \Delta AM_{ik}$
- number of inserted arcs
  $AI(S_k) = AI(S_i) + \Delta AI_{ik}$
- number of deleted arcs
  $AD(S_k) = AD(S_i) + \Delta AD_{ik}$
- cost of the subgraph match
  $cost(S_k)$

7.2.3.2 Expansion of the root state

The root state S_{root} of the search tree has a special meaning since the core node sets are empty. According to the definitions, the terminal sets as well as the set of matches are empty, too.
Therefore, we have to adapt slightly the rules establish for the general case. First, we initiate some sets and attributes:

- reference core node set	C_{Rroot}	$= \{ \}$
- scene core node set	C_{Sroot}	$= \{ \}$
- set of couples of matched core nodes	M_{root}	$= \{ \}$
- reference rest node set	R_{Rroot}	$= \{ N_R \mid N_R \in G_R \}$
- scene rest node set	R_{Sroot}	$= \{ N_S \mid N_S \in G_S \}$
- search level-	$level(S_{root})$	$= 0$
- number of arcs matched	$AM(S_{root})$	$= 0$
- number of inserted arcs	$AI(S_{root})$	$= 0$
- number of deleted arcs	$AD(S_{root})$	$= 0$

For each couple of nodes (N_{Rn}, N_{Sm}), we attempt to create a new state, setting N_{Rn} and N_{Sm} virtually as candidate terminal states:

$NT_{Rn} = N_{Rn}$ where $N_{Rn} \in R_{Rroot}$
$NT_{Sm} = N_{Sm}$ $N_{Sm} \in R_{Sroot}$

With these modifications realized, we can apply the rules for the general case of the previous section.

7.2.4 Order of expansion

Until now, we have discussed how to expand the search tree under the condition of connectivity, but we are not fixed yet on the priorities under which we will expand one state out of the set of the non expanded states.

We discuss three approaches:

- breadth-first search
- depth-first search
- best-first search

breadth-first

Breadth-first considers every state at each level of the tree before going deeper into the space, all states are first reached along the shortest path from the start state. Breadth-first is, therefore, guaranteed to find the shortest path from the start state to the goal, a leaf state of the tree. Furthermore, since all states are first found along the shortest path, any states encountered a second time are found along a path of equal or greater length. On the assumption that the cost of a partial path is positive, there is no chance that duplicate states were found along a better path, the algorithm simply discards them.

Unfortunately, if there is a large branching factor, i.e. states have a high average number of descendants, the combinatorial explosion may prevent the algorithm from finding a solution using the available space. This is due to the fact that all unexpanded states for each level of the search must be kept in memory. For deep searches, or state spaces with a high branching factor, this can be quite cumbersome.

The space utilization of breadth-first, measured in terms of the number of states kept in memory, is an exponential function of the path at any time. If each state has an average of B children, we get B^n states on level n.

depth-first

Unlike breadth-first search, a depth-first search is not guaranteed to find the shortest path to a state the first time. Later in the search, a different path may be found to some state.

Depth-first search gets quickly into a deep search, so depth-first search can get lost in deep search, missing (temporarily) shorter paths to the goal.

The space utilization of depth-first, measured in terms of the number of states kept in memory, is a linear function of the path length n at any time. If each state has an average of B children, we get (B-n) states.

best-first

While a breadth-first search leads to a big number of states of relatively low level and lengthen uselessly the search, a depth first algorithm instead focuses very much on a particular solution which initially was promising, but which could have a decreasing importance.

Therefore, we haven chosen a third way, called "best-first search", which allows us to reevaluate the situation after each expansion of a search tree state, this using some cost function (fig 7.5).

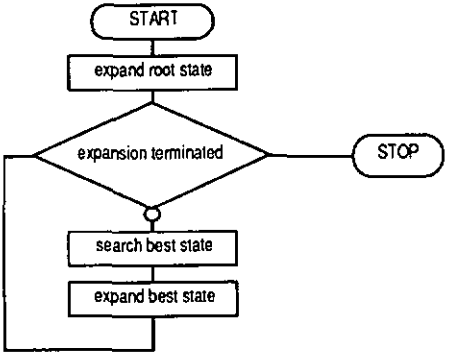


fig 7.5: search tree expansion routine

Best-first search applies an evaluation to the states which are not expanded yet, and the list is sorted according to the values found. This brings the "best" state to the front of the list. Note that a next state to be examined may be from any level of the state space.
In our application, we select the non-expanded state with the minimum "cost" attribute. A well adapted cost computation rule allows us to favours well developed paths.

7.3 Search tree heuristics

The only condition which limited the expansion of the search tree until now is the connectivity condition. This condition is based entirely on the structure of the graphs. In order to prune the search tree further, we will take profit of some more information represented in the graphs.

Some conditions, based on structural aspects only, are usable for both attributed and non-attributed graphs, the rest exclusively for attributed graphs. Table 7.6 gives an overview on the pruning conditions used, the order following the implementation in the complete algorithm. The second column indicates whether the condition has been introduced using some heuristics, whether it is problem definition dependent or whether it is just a condition which is due to our particular implementation.

1. connectivity condition	problem definition	structural
2. supernode condition	implementation	attributal
3. rigidity condition	problem definition	attributal
4. visibility condition	problem definition	attributal
5. new match condition	implementation	structural
6. nodedegree condition	heuristic	structural
7. neighbor match condition	heuristic	structural
8. cost threshold condition	heuristic	structural/ attributal
9. subgraph similarity condition	heuristic	attributal
9.1 node similarity condition	heuristic	attributal
9.2 arc similarity condition	heuristic	attributal
9.3 rotation fitting condition	heuristic	attributal
10. search depth condition	problem definition	structural

table 7.6: search tree pruning conditions

7.3.1. Connectivity condition

structural information

The aim of the matching process is to establish hypotheses on models which could possibly correspond to the objects found in the scene. We can see that the use of adjacent regions of the scene, represented in the graph by adjacent nodes, plays an important role in the process, since they represent corners and edges, elements which are easy to locate, and which contain a huge amount of information (discontinuities of surface properties such as intensity, orientation, ...).
Therefore, we have established the connectivity condition: the candidate terminal nodes NT_{Rm} and NT_{Sn} have to be adjacent to at least one, already matched core node of the respective sets.

connectivity condition

Given the core node sets C_{Rl} and C_{Sl} of size l, standing for a match M_l (fig 7.7, next page). Are candidate node couples for an expansion of the match M_l to match M_k of level l+1 all couples (NT_{Rm}, NT_{Sn}) which are members of the following set:

$$CMS_l = \{ (NT_{Rm}, NT_{Sn}) \mid (NT_{Rm} \in T_{Rl}) \wedge (NT_{Sn} \in T_{Sl}) \}$$

where NT_{Rm} and NT_{Sn} are members of the respective terminal node sets T_{Rl} and T_{Sl}.

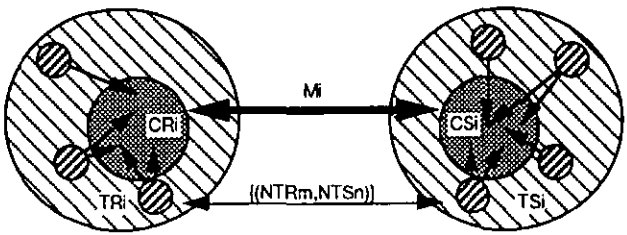


fig 7.7: connectivity condition: terminal nodes (hatched) and their adjacency relations to the core node set (dotted)

7.3.2 Supernode condition

attributal information

In order to represent incomplete scene data in the form of pure graphs in the mathematical sense, we have introduced earlier a placeholder node, the supernode SN_R resp. SN_S which does not stand for a particular region found in the scene, but for the union of all unknown or badly defined regions.

While the existence of the supernode delivers some information on the structural aspects of the graph, the attributes are not useable and we do not create matches with supernodes: all candidate match couples, where either one or both of the nodes are supernodes, will be discarded from the candidate match set CMS_l:

$$CMS_l = \{(NT_{Rm}, NT_{Sn}) \mid (NT_{Rm} \in T_{Rl}) \wedge (NT_{Sn} \in T_{Sl}) \wedge (NT_{Rm} \neq SN_R) \wedge (NT_{Sn} \neq SN_S)\}$$

7.3.3 Rigidity condition

hypothesis of rigidity

All objects are rigid, the angles between the face normals are constant and rotation invariant.

attributal information

This condition can only be applied for attributed graphs which contain information on the orientation of the represented elements. In our application, these are the normal vectors on the regions. Since the regions themselves correspond to nodes in the graph representation, the region normal vectors will appear as node attributes.

hypothesis of rigidity

-> both scene and model objects are rigid

When attempting a subgraph match M_i, corresponding to search state S_i, we establish a hypothesis on the correspondences between nodes of scene and reference graph and, therefore, also on the correspondences of the attributes of the nodes involved. This allows us, in general, to establish a hypothesis on the "best fitting rotation" Rot_{Si} which transforms "best" the set of unitary normal vectors of the reference core nodes N_{Ri}, $N_{Ri} \in C_{Ri}$, to the set of unitary normal vectors of the scene core nodes N_{Si}, $N_{Si} \in C_{Si}$, while respecting the matches M_i established.

minimization problem

The problem to find a "best fitting rotation" Rot_{Si} can be seen as a "minimal squared error fitting problem" where the error is defined as the sum of squared euclidian distances between rotated reference vectors and corresponding scene vectors.

$$\min \sum \gamma_j \; | \; \text{normal}(N_{Sj}) - Rot_{Si} (\text{normal}(N_{Rj})) \; |^2$$

The weighting factor γ_j is defined as the geometric mean of the surface areas of both reference and scene regions involved. It is computed with the following ideas in mind:

- the regions represented should participate according to their importance (area)
- the reliability of the orientation information depends on the surface area of the region on which it is based

$$\text{weighting factor } \gamma_j = \sqrt{\text{area}(N_{Sj}) \, \text{area}(N_{Rj})}$$

computation

The minimizing problem itself is a complex one and is worth to be discussed in detail. In fact, an intelligent solution allows to reduce in an important manner the computation time, an interesting point since this complex work has to be done again at each state of the search tree. The details on the realization, which has been formalized using a special algebra for hypercomplex numbers, can be found in chapter 7.6.

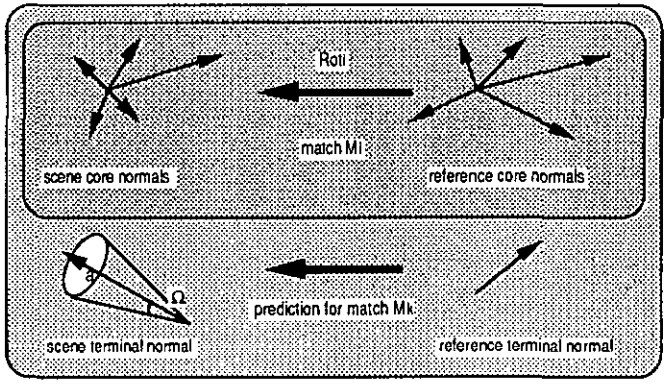


fig 7.8: rigidity constraint: prediction of a scene terminal node normal

Once the rotation Rot_{Si} known for state S_i, we can restrict the candidate match set CMS_i to couples of nodes (NT_{Sm}, NT_{Rn}) which fulfill the rigidity condition:

$$| \{ (normal(NT_{Sm}), (Rot_{Si} (normal(NT_{Rn})) \} | \leq TH_{\Omega}$$

If the scene normal vector of node NT_{Sm} is outside a cone defined by the opening angle Ω and axis **a** formed by the predicted normal vector, then the corresponding node/node couple (NT_{Sm}, NT_{Rn}) is discarded from CMS_i.

parent state level	sizes of matched subgraphs	rotation
0	0	undetermined
1	1	infinite number of rotations
≥ 2	≥ 2	generally overdetermined -> "best fitting" rotation

table 7.9: rigidity constraint

The use of this condition is restricted to the prediction of search tree son states of level 3 and higher (table 7.9). For lower levels, the underlying data is not sufficient to determine uniquely a rotation and, in this case, we do not apply the condition.

7.3.4 Visibility condition

concept of visibility

Using a 2.5 D acquisition system, by definition only a part of the scene, and, therefore, also the scene object, is visible. We determine this subspace VS_S (visible scene space) and, having established a hypothesis on the orientation

of the scene object using the rotation matrix determined before, transform VS_S to the reference space, defining $VS_R = Rot_{Si}^{-1}(VS_S)$.

This allows us to prune the reference terminal node set.

> Are only candidates for a correspondence the reference terminal nodes with an orientation attribute compatible with the visible reference space VS_R.

Practically, we assume a monocular acquisition system for which we determine an observation vector OV_S in the scene space. Transformed to the reference space using the rotation Rot_{Si}^{-1}, we define a reference observation vector OV_R and an associated visible reference space VS_R. In terms of the gaussian sphere representation, VS_R is approximatively a semi-sphere GS_{VSR}, positioned symmetrically with respect to the reference observation vector $OV_R = Rot_{Si}^{-1}(OV_S)$.

(Since all conditions have been implemented with a certain tolerance ε, we neglect the influence of perspective and data acquisition inaccuracy. In this particular case, the opening angle Ω of the cone GS_{VSR} will not be 90 degrees but $90° + \varepsilon$.)

The visible half space allows to predict which ones of the reference vectors out of the candidate set are visible after rotation to the scene space (occlusion not considered). The candidate reference node set T_{Ri} is pruned to nodes N_{Rj} which fulfill the visibility condition, i.e. their normal vectors must be inside GS_{VHR} and the candidate match set CMS_i is restricted further:

$$CMS_i = \{(NT_{Rm}, NT_{Sn}) \mid (NT_{Rm} \in T_{Ri}) \wedge (normal(NT_{Rm}) \in GS_{VSR}) \wedge (NT_{Sn} \in T_{SI})\}$$

This condition has been realized implicitly in the algorithm for the rigidity condition, at least for predictions on matches of level 3 and higher. Since

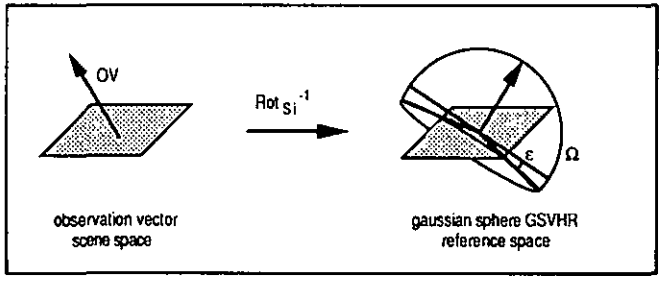


fig 7.10: visibility condition

the scene vectors are visible by definition, the condition hereafter, introduced in chapter 7.3.3, covers at the same time the visibility condition too:

$$| \sphericalangle (\text{normal}(NT_{Sm}), \text{normal}(Rot_{Si} (NT_{Rn}))) | \leq TH_{\Omega}$$

If the reference normal vector of node NT_{Rn} is outside a cone of opening angle Ω ($\Omega = \varepsilon$) and axis **a** formed by the scene normal vector, the corresponding node/node couple (NT_{Sm}, NT_{Rn}) is to be discarded from CMS_i.

The rigidity condition we have used is "forward checking" and has its impact on level n+1, using data from level n. As we mentioned, we are limited to levels $n \geq 2$, the implicit visibility condition is available from level 3 on. For level 1, we cannot apply any condition since there is an infinite number of rotations Rot_{Si} which map reference elements to the scene. For level 2 in turn, we can implement a "backward check". Once a match established, we can check if it was acceptable in the sense of the visibility condition, else we discard it.

7.3.5 New match condition

structural information

In a search tree, different paths may reach the same state S_i, or, what is equivalent for our problem, there are different ways to expand subgraph matches so that identical, bigger, subgraph matches occur (fig 7.11, next page).

Therefore, for each attempt to create a new state S_k of level k, we check if there exists already a state S_p of the same level which stands for an identical subgraph match.

7.3.6 Nodedegree condition

structural information

This condition is concerned with the structural aspect of the graphs involved. The nodedegree of node N of a non-directed graph G can be defined as the number of nodes N_a, adjacent to node N.

If the difference in nodedegrees of reference node NT_{Rm} and scene node NT_{Sn} is bigger than a given threshold TH_{ND}, we discard the couple (NT_{Rm}, NT_{Sn}) from the candidate match sets CMS_i.

$$| \text{nodedegree}(NT_{Rm}) - \text{nodedegree}(NT_{Sn}) | \leq TH_{ND}$$

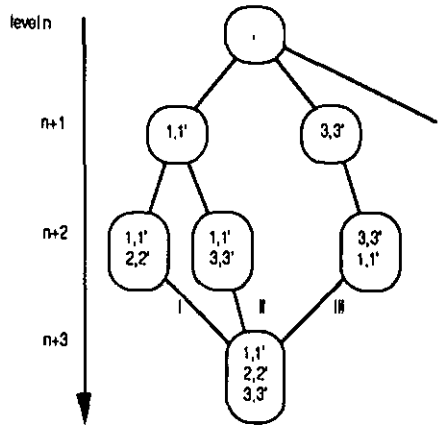


fig 7.11: multiple paths leading to a same state

The idea is to request similar structural neighborhoods but there are two important points to take into account when fixing the threshold.

i) Considering the non-idealities of a system based on acquired natural data, we have to accept some tolerance.

ii) By definition, scene objects have invisible faces. The arcs in the scene graph which should represent the borders between visible faces and the background will exist in the form of one single arc to the supernode.

Therefore, the application of the nodedegree condition is very delicate and in fact limited to nodes which represent faces which are not adjacent to invisible parts of objects.

7.3.7 Neighbor match condition

structural information

By definition, each node NT_{Rm} of the reference terminal node set T_{Rk} and NT_{Sn} of the scene terminal node set T_{Sk} respectively has at least one core node of the respective node sets C_{Rk} respectively C_{Sk} as a neighbor.
Now we establish a condition which guarantees that the candidate terminal nodes have an adjacency to core nodes which have been matched before, this in the respective graphs. This allows to match neighboring reference nodes with neighboring scene nodes.

We establish the neighbor match condition:

$$((NT_{Rm},NT_{Sn}) \in CSM_l)) \quad \Leftrightarrow \quad (\mathcal{A}(NT_{Rm},NC_{Ri}) \wedge \mathcal{A}(NT_{Sn},NC_{Sj}) \wedge ((NC_{Ri},NC_{Sj}) \in M_l))$$

The candidate couple (NT_{Rm},NT_{Sn}) is element of the candidate match set

CMS_i if and only if there exists an adjacency $\mathcal{A}(NT_{Rm}, NC_{Rk})$, an adjacency $\mathcal{A}(NT_{Sn}, NC_{Sj})$ and if the core node correspondence (NC_{Rl}, NC_{Sj}) is element of the match set M_i.

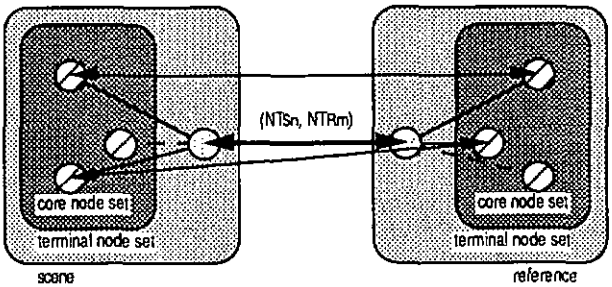


fig 7.12: neighborhood condition

7.3.8 Cost threshold condition

structural/sttributal information

When adding a candidate couple (NT_{Rm}, NT_{Sn}) to a match M_i, represented in the search tree by state S_i, in order to create a new match M_k, represented by state S_k, we modify the global matching cost by $\Delta cost(S_i, S_k)$. This cost can be computed, for non-attributed graphs based on structural differences and, for attributed graphs, based on attribute dissimilarities. Details on the computation of the cost will be discussed in chapter 7.6.
If $\Delta cost(S_i, S_k)$ is bigger than a given threshold TH_{cost}, we discard the couple (NT_{Rm}, NT_{Sn}) from the candidate match set CMS_i.

$$\Delta cost(S_i, S_k) \leq TH_{cost}$$

The idea is to abandon search branches which would lead to an important increment in the global cost.

7.3.9 Additional subgraph similarity conditions

attributal information

Beside the global cost $\Delta cost(S_i, S_k)$, due to the addition of a new state, a particular similarity condition can be established for each attribute of both arcs and nodes. The idea is to abandon search branches where important attribute dissimilarities have been found.

node similarity condition

The additional node dissimilarity $\Delta diss_{node}(S_i, S_k)$ due to the addition of the candidate match (NT_{Rm}, NT_{Sn}) must be below a given threshold TH_{node}.

$$\Delta diss_{node}(S_i, S_k) \leq TH_{node}$$

arc attribute similarity condition

The additional arc attribute dissimilarity $\Delta diss_{arcatt}(S_i, S_k)$ due to the addition of the candidate match (NT_{Rm}, NT_{Sn}) must be below a given threshold TH_{arcatt}

$$\Delta diss_{arcatt}(S_i, S_k) \leq TH_{arcatt}$$

rotation fitting condition

The additional rotation fitting error $\Delta e_{rotfit}(S_i, S_k)$ due to the addition of the candidate match (NT_{Rm}, NT_{Sn}) must be below a given threshold TH_{rotfit}

$$\Delta e_{rotfit}(S_i, S_k) \leq TH_{rotfit}$$

7.3.10 Search depth condition

structural information

In some cases, it can be sufficient to stop the expansion of the search tree in depth once a certain condition on the extent of the subgraphs matched is fulfilled, for example, when a partial match is sufficient. We have implemented a condition on the number of nodes matched, corresponding to the search level: the expansion of a search tree state S_k is stopped when the search level threshold $TH_{maxlevel}$ is reached.

$$level(S_k) \leq TH_{maxlevel}$$

7.4 Global Search Tree Stopping Conditions

After the search tree pruning conditions discussed in chapter 7.3, we pass now to the conditions which stop completely the search.

end of expansion

The first stopping condition is very natural. Once there are no more states to expand, the search is definitely finished and the process stopped.

solutions found

The second condition is concerned with the goal of the search, defined by some parameters. If we are just looking for the first solution, which is supposed to be the best one ("best first" approach), we can set the threshold for the number of solutions $TH_{solutions} = 1$.

An exact definition of the term "solution" will be given in the chapter 8.

implementation dependent conditions

Further conditions are implementation dependent, limiting the memory

space and CPU time available. These two implementation dependent conditions can be useful when the match is difficult or even impossible or when the parameters are badly tuned:

- the computation time t_{CPU} has reached the fixed threshold TH_{CPU}, or
- the number of states has reached the fixed threshold TH_{states}.

7.5 Rotation fitting

As we described shortly in chapter 7.3.3, we would like to resolve the problem formalized below and illustrated by figure 7.6.1.

definitions

Given a set of reference surfaces $S_R = \{...,S_{ri},...\}$ and the respective set of unitary normal vectors $N_R = \{...,r_i,...\}$.
Given a set of scene surfaces $S_S = \{...,S_{si},...\}$ and the respective set of unitary normal vectors $N_S = \{...,s_i,...\}$.
Given a set of correspondences $M = \{ ..., (S_{ri}, S_{si}), ... \}$ between reference surfaces S_{ri} and scene surfaces S_{si}.

problem

Find a rotation R which transforms "best" the reference normal vectors r_i of set N_R to the scene normal vector s_i of the set N_S while respecting the correspondences of set M.

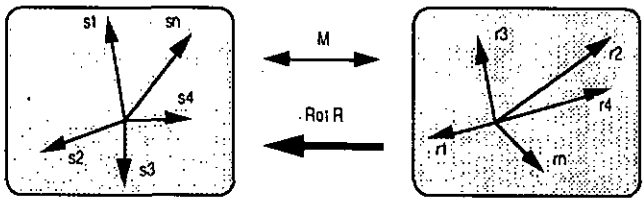


fig 7.13: the rotation fitting problem

Depending on the size of the set of correspondences $M = \{..., (S_{ri}, S_{si}), ...\}$, we can distinguish between two cases.

i) If just one correspondence (S_{r1},S_{s1}) is known, an infinite number of rotations fulfilling the conditions can be found. The result has no practical use and will not be discussed further.

ii) For bigger sets of correspondences, the transformation is in general overdetermined, a "best fitting" solution has to be found.

7.5.1 Condition of optimization, selection of the approach

condition

Given n correspondences (S_{ri}, S_{si}), $2 \le i \le n$, the rotation R which transforms normal vector r_i to s_i is generally overdetermined. We are looking for a "best fitting" rotation R, using a "least squared error fitting" condition:

$$\min e^2 = \min \sum_i | R(r_i) - s_i |^2$$

approaches

Several representation of rotations are possible [Fau87b]:

- orthonormal matrix R: $R^tR = I$
- axis **p** and angle Θ
- quaternion multiplications

While the first representation leads to a high dimensional space of constraints, the second one has a non-polynomial condition. In both cases, an iterative algorithm is the only solution. The third representation gives the simplest way and will be used [Fau87b].

7.5.2 The quaternion approach

First, we introduce a small set of terms and computation rule from the algebra of quaternions. For an excellent introduction to the subject of hypercomplex numbers see [Kan89], for our specific 3D rotation problem see [Ama90a].
Second, we replace in the minimization condition from above the transformation R by quaternion multiplications and resolve the "fitting problem" in a pretty way.
The chapter will terminate with some investigations on the "fitting error" as well as on some aspects concerned with the particular implementation.

7.5.2.1 Introduction to quaternions

A quaternion q is a hypercomplex number, consisting of a real part and three imaginary parts. It can, therefore, be written as $q = a + bi + cj + dk$ where a,b,c,d are reals and i,j,k imaginary numbers. From a geometric point of view, we can interpret **i**, **j** and **k** as unit vectors in a space (i,j,k), so

$$q = a + b\,\mathbf{i} + c\,\mathbf{j} + d\,\mathbf{k}$$

where **i**, **j**, **k** are unit vectors of an orthonormal space (i,j,k) which is R^3.

Hence, for the "rotation fitting problem" we can write in an abbreviated notation

$$q = v + \mathbf{i}^t\mathbf{w} \qquad \text{where} \qquad v = \text{real part}, \ \mathbf{w} = \text{vector part}, \ \mathbf{i} = \begin{bmatrix} i \\ j \\ k \end{bmatrix}$$

In the following, we will use this abbreviated notation, keeping in mind that the quaternions have a much larger definition and use than the one we discuss here [Kan89].

definitions

quaternion $q = v + \mathbf{i}^t\mathbf{w}$

conjugate $\bar{q} = v - \mathbf{i}^t\mathbf{w}$

quaternion addition $q + q' = (v+v') + \mathbf{i}^t(\mathbf{w}+\mathbf{w}')$

quaternion multiplication $q \otimes q' = (vv'-\mathbf{w}\cdot\mathbf{w}') + \mathbf{i}^t(\mathbf{w}\wedge\mathbf{w}'+v\mathbf{w}'+v'\mathbf{w})$

vector of R^3 ≡ pure vector quaternion q

$q = \mathbf{i}^t\mathbf{w} \qquad v = 0$

(In the following, we use **bold** typesettings for vectors, matrices as well as pure vector quaternions)

properties

With these definitions, we can write

module (extended norm) $|q|^2 = |\bar{q}|^2 = q \otimes \bar{q} = \bar{q} \otimes q = v^2 + |\mathbf{w}|^2$

multiplicity $|q \otimes \bar{q}|^2 = |q|^2 |\bar{q}|^2$

7.5.2.2 Quaternions and rotations in 3D

Using the quaternion approach for rotations, the usual transformation matrix/vector expression can be replaced, using quaternion multiplications [Kan89], [Ama90a].

Knowing the rotation axis $\mathbf{p}$ in R^3 and the rotation angle Θ (positive angles following the "right hand rule"), we write for the R^3 vector $\mathbf{u}$ which is represented in the quaternion space as a pure vector quaternion $\mathbf{u}$:

$$\mathbf{Ru} \equiv q \otimes \mathbf{u} \otimes \bar{q} \qquad \text{where} \quad q = v + \mathbf{i}^t\mathbf{w} \quad |q| = v^2 + |\mathbf{w}|^2 = 1$$
$$v = \cos\frac{\Theta}{2} \quad \mathbf{w} = \sin\frac{\Theta}{2}\mathbf{p}$$

The vector $\mathbf{w}$ is parallel to the rotation axis and scaled following the rotation angle, the real part v depends only on the rotation angle. The module of q must be equal to 1.

7.5.3 Application of the quaternion approach to our problem

We can write the original fitting error minimization condition as

$$\min e^2 = \min \sum_i |\, R\mathbf{r}_i - \mathbf{s}_i \,|^2 = \min \sum_i |\, q \otimes \mathbf{r}_i \otimes \bar{q} - \mathbf{s}_i \,|^2$$

Since the quaternion module is multiplicative, we can write

$$e^2 \, |q|^2 = \sum_i |(q \otimes \mathbf{r}_i \otimes \bar{q} - \mathbf{s}_i) \otimes (q)|^2$$

$$= \sum_i |(q \otimes \mathbf{r}_i)\,(\bar{q} \otimes q) - \mathbf{s}_i \otimes q|^2$$

$$= \sum_i |(q \otimes \mathbf{r}_i)\,|q|^2 - \mathbf{s}_i \otimes q|^2$$

and, with $|q| = 1$

$$e^2 = \sum_i | \, q \otimes \mathbf{r}_i - \mathbf{s}_i \otimes q \, |^2$$

The definition of the quaternion multiplication shows that $(q \otimes \mathbf{r}_i - \mathbf{s}_i \otimes q)$ is a linear function of q. Therefore, there exist symmetric matrices A_i such that

$$e^2 = \sum_i | \, q \otimes \mathbf{r}_i - \mathbf{s}_i \otimes q \, |^2$$

$$= \sum_i \mathbf{P}^t \mathbf{A}_i \mathbf{P}$$

$$= \mathbf{P}^t \mathbf{B} \mathbf{P}$$

where $\mathbf{B} = \sum_i \mathbf{A}_i$, $\mathbf{I}^t \mathbf{P} = p = a + bi + cj + dk$, $\mathbf{P} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$, $\mathbf{I} = \begin{bmatrix} 1 \\ i \\ j \\ k \end{bmatrix}$

Finally, we can write the minimization problem as

$$\min e^2 = \min \sum_i | \, R \mathbf{r}_i - \mathbf{s}_i \, |^2 = \min \mathbf{P}^t \mathbf{B} \mathbf{P} \qquad \mathbf{B} = \sum_i \mathbf{A}_i$$

The resulting best fitting transformation occurs in the form of the quaternion **P**. each couple of corresponding normal vectors (r_i, s_i) leads to a particular matrix $\mathbf{A}_i$.

Since **A**, and, therefore, also **B**, are symmetric, the solution to the minimization problem is the eigenvector $\mathbf{P}_{min}$ corresponding to the eigenvalue λ_{min} whose absolute value is minimal. This leads, by the use of the initial correspondences quaternion/rotation, to the final result:

i) the rotation axis **p**
ii) the rotation angle Θ

7.5.4 Error determination

Evidently, a fitting error can be found. With the last equation we got, leading

to an eigenvalue λ_{min} and an associated eigenvector $\mathbf{P}_{min}$, we write

$$\min e^2 = \min \sum_i | R \mathbf{r}_i - \mathbf{s}_i |^2 = \min \mathbf{P}^t \mathbf{B} \mathbf{P} \quad \Rightarrow \quad \lambda_{min}, \mathbf{P}_{min}$$

Inserting these values as well in the formula representing the correspondence between matrix **B** and its eigenvalue λ_{min} and eigenvector $\mathbf{P}_{min}$, as in our formula from above, we get

$$\mathbf{B} \mathbf{P}_{min} = \lambda_{min} \mathbf{P}_{min}$$

$$e^2_{min} = \mathbf{P}_{min}{}^t \mathbf{B} \mathbf{P}_{min} = \lambda_{min} \mathbf{P}_{min}{}^t \mathbf{P}_{min}$$

$$e^2_{min} = \lambda_{min}$$

Since we have still the constraint $|p_{min}|^2 = \mathbf{P}_{min}{}^t \mathbf{P}_{min} = 1$, we find the final result: the minimal squared accumulated error e^2 is equal to λ_{min}.

7.5.5 Extraction of the transformation matrix R from axis p and angle Θ

Once the quaternion $\mathbf{P}_{min}$ found, axis **p** and rotation angle Θ can be found. For practical reasons, we convert this axis/angle representation of the rotation into the form of an orthonormal transformation matrix.
Following [Bron80], a cartesian coordinate system [$\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$] can be rotated around an axis **p** by an angle of Θ (mathematically positive) as follows:

$$\alpha = \cos(\sphericalangle \mathbf{e}_1, \mathbf{p}) \qquad \beta = \cos(\sphericalangle \mathbf{e}_2, \mathbf{p}) \qquad \gamma = \cos(\sphericalangle \mathbf{e}_3, \mathbf{p})$$

$$\mathbf{Q} = \begin{bmatrix} \cos\Theta + \alpha^2(1-\cos\Theta) & \gamma\sin\Theta+\alpha\beta(1-\cos\Theta) & -\beta\sin\Theta+\alpha\gamma(1-\cos\Theta) \\ -\gamma\sin\Theta+\alpha\beta(1-\cos\Theta) & \cos\Theta+\beta^2(1-\cos\Theta) & \alpha\sin\Theta+\beta\gamma(1-\cos\Theta) \\ \beta\sin\Theta+\alpha\gamma(1-\cos\Theta) & -\alpha\sin\Theta+\beta\gamma(1-\cos\Theta) & \cos\Theta+\gamma^2(1-\cos\Theta) \end{bmatrix}$$

Final transformation: $\mathbf{R} = \mathbf{Q}^{-1}$

7.5.6 Extension to weighted coefficients

The influence of the different components can be weighted, using a weighting coefficient γ_i for each couple (r_i, s_i) of normal vectors. Putting it in the respective formulas of the previous chapters, we get

$$\min e^2 = \min \sum_i \gamma_i | R(r_i) - s_i |^2 = \min \sum_i \mathbf{P}^t \gamma_i \mathbf{A}_i \mathbf{P} = \min \mathbf{P}^t \mathbf{B} \mathbf{P} \qquad \mathbf{B} = \sum_i \gamma_i \mathbf{A}_i$$

The weighted minimum squared error is, as before, equal to the smallest eigenvalue λ_{min}. An estimation of the mean squared error due to a single couple (r_i, s_i) can be done using the formula below

$$e^2_{mean} = \frac{\min \sum_i \gamma_i \, |R(r_i) - s_i|^2}{\sum_i \gamma_i} = \frac{\lambda_{min}}{\sum_i \gamma_i} \qquad \text{where } i = 1 \,..\, n$$

7.5.7 Determination of the coefficients of matrix **A**

For the determination of the coefficients of A, we use the equation of chapter 7.5.3 which states:

$$\begin{aligned}
\mathbf{P}^t \mathbf{A}_i \mathbf{P} &= | \; p \otimes \mathbf{r}_i - \mathbf{s}_i \otimes p \; |^2 \\
&= | \; (-\mathbf{w}\mathbf{r}_i - \mathbf{s}_i\mathbf{w}) + \mathbf{i}^t(\mathbf{w} \wedge \mathbf{r}_i + v\mathbf{r}_i - \mathbf{s}_i \wedge \mathbf{w} - v\mathbf{s}_i) \; |^2 \\
&= | \; (-\mathbf{w}(\mathbf{r}_i - \mathbf{s}_i)) + \mathbf{i}^t(\mathbf{w} \wedge (\mathbf{r}_i - \mathbf{s}_i) + v(\mathbf{r}_i - \mathbf{s}_i)) \; |^2 \\
&= (\mathbf{w}(\mathbf{r}_i - \mathbf{s}_i))^2 + | \; \mathbf{w} \wedge (\mathbf{r}_i + \mathbf{s}_i) + v(\mathbf{r}_i - \mathbf{s}_i) \; |^2 \\
&= \sum_i \sum_{m,n=1}^{4} a_{mni} \, p_n \, p_m \qquad \text{where} \quad v = p_1, \quad \mathbf{w} = \begin{bmatrix} p_2 \\ p_3 \\ p_4 \end{bmatrix}
\end{aligned}$$

With the idea in mind that matrix $\mathbf{A}_i$ must be symmetric, the comparison of the coefficients allows to extract, after a long development, the elements of $\mathbf{A}_i$ (Table 7.14).

$$\begin{aligned}
a_{11i} &= (r_{xi} - s_{xi})^2 + (r_{yi} + s_{yi})^2 + (r_{zi} + s_{zi})^2 \\
a_{22i} &= (r_{xi} + s_{xi})^2 + (r_{yi} - s_{yi})^2 + (r_{zi} + s_{zi})^2 \\
a_{33i} &= (r_{xi} + s_{xi})^2 + (r_{yi} + s_{yi})^2 + (r_{zi} - s_{zi})^2 \\
a_{44i} &= (r_{xi} - s_{xi})^2 + (r_{yi} - s_{yi})^2 + (r_{zi} - s_{zi})^2 \\
a_{12i} = a_{21i} &= -2 \, (r_{xi} s_{yi} + r_{yi} s_{xi}) \\
a_{23i} = a_{32i} &= -2 \, (r_{yi} s_{zi} + r_{zi} s_{yi}) \\
a_{13i} = a_{31i} &= -2 \, (r_{xi} s_{zi} + r_{zi} s_{xi}) \\
a_{14i} = a_{41i} &= 2 \, (r_{zi} s_{yi} - r_{yi} s_{zi}) \\
a_{24i} = a_{42i} &= 2 \, (r_{xi} s_{zi} - r_{zi} s_{xi}) \\
a_{34i} = a_{43i} &= 2 \, (r_{yi} s_{xi} - r_{xi} s_{yi})
\end{aligned}$$

Table 7.14: determination of the coefficients of $\mathbf{A}_i$

7.5.8 Application of the approach to the correspondence search problem

The combination of the elements introduced before leads to the following algorithm.

algorithm

1. take a first couple of normal vectors (r_i, s_j)
   - determine the matrix $\mathbf{A}_1$ using the equations of table 7.14
   - (option: determine the weighting factor γ_1)
   - $\mathbf{B}_1 = \gamma_1 \mathbf{A}_1$, n = 1
2. take the next couple of normal vectors (r_k, s_l)
   - n = n + 1
   - determine the corresponding matrix $\mathbf{A}_n$
   - (option: determine the weighting factor γ_n)
   - $\mathbf{B}_n = \mathbf{B}_{n-1} + \gamma_n \mathbf{A}_n$
   - minimize $\mathbf{P}_n^t \mathbf{B}_n \mathbf{P}_n$
     -> quaternion representation of the "best fitting rotation" $\mathbf{p}_{n\ min}$
     -> corresponding squared error e^2_{min} (§7.5.4. §7.5.6)
   - (option: convert the resulting "best fitting rotation" form the quaternion form $\mathbf{p}_{n\ min}$ to the conventional rotation matrix representation R_n (§7.5.5.))
3. continue with step 2

comment

The algorithm must be understood as a summary of paragraph 7.5. For an application, consult the full formulas with the corresponding remarks.

7.5.9 Conclusions

We have presented a method to determine a transformation R which maps "best" a set of n reference normal vectors r_i to the corresponding (matched) scene vectors s_i.

The advantages of the derived algorithm are the following:

- matrix **B** can be adapted iteratively to a growing set of correspondences, an interesting point for our state space search tree environment: $\mathbf{B}_n = \mathbf{B}_{n-1} + \gamma_n \mathbf{A}_n$
- the determination of a best fitting rotation R is straight forward
- the determination of the fitting error is particularly easy
- the minimization problem has been shown to be an eigenvector/eigenvalue problem. Therefore, we can rely on the literature as well as on the software packages existing for this well known problem.

7.6 Dissimilarities and costs

In the following, we present the computation rules we apply for the determination of both structural and attributal dissimilarities.

notations

i) Dissimilarity in attribute z between the elements x_m and y_n

$$d_z(x_m, y_n)$$

ii) Change in dissimilarity in attribute z when passing from state S_i to state S_k. This value is the sum over the dissimilarities in attribute z for the set **A** of couples of elements (x_m, y_n) matched when extending submatch i, represented by state S_i, to submatch k, represented by state S_K.

$$\Delta D_z(S_i, S_k) = \sum_{(xm,yn)\in A} d_z(x_m, y_n)$$

iii) Accumulated dissimilitude in attribute z over a path from the root state S_{root} to state S_k

$$D_z(S_k) = \sum_{root}^{k} \Delta D_z(S_i, S_k)$$

iv) number of elements x due to the transition from state S_i to state S_k

$$\Delta A_x(S_i, S_k)$$

v) accumulated number of x over a path from the root state to state S_k

$$A_x(S_k) = \sum_{root}^{k} \Delta A_x(S_i, S_k)$$

7.6.1 Attribute dissimilarity

We compute dissimilarities for the following arc and node attributes:

- node attribute surface	area (N_i)
- node attribute normal vector	normal (N_i)
- arc attribute borderlength	border (A_i)

For each correspondence (N_{Rm}, N_{Sn}), which induces also a set **A** of arc correspondences, we get:

orientation

$$\Delta e_{rotfit}(S_i, S_k) = e_{rotfit}(S_k) - e_{rotfit}(S_i)$$

where the rotation fitting error $e_{rotfit}(S_q)$ is defined as the (weighted)

sum of the squared euclidian distances between the rotated reference normal vectors and the corresponding scene normal vectors, accumulated over all correspondences (N_{Sn}, N_{Rn}) of M_q:

$$e_{rotfit}(S_q) = \frac{\sum_{root}^{q} \gamma_n \; | \; normal(N_{Sn}) - Rot_{Sq}(normal(N_{Rn})) \; |^2}{2 \sum_{root}^{q} \gamma_n}$$

surface area

$$\Delta diss_{area}(S_i, S_k) = diss_{area}(N_{Rm}, N_{Sn}) = \frac{|\, area(N_{Rm}) - area(N_{Sn}) \,|}{max(area(N_{Rm}), area(N_{Sn}))}$$

where $0 \le diss_{area}(N_{Rm}, N_{Sn}) \le 1$

borderlength

$$diss_{border}(A_{Rm}, A_{Sn}) = \frac{|\, border(A_{Rm}) - border(A_{Sn}) \,|}{max(border(A_{Rm}), border(A_{Sn}))}$$

$$\Delta diss_{border}(S_i, S_k) = \sum_{(x_m, y_n) \in A} diss_{border}(A_{Rm}, A_{Sn})$$

7.6.2 structural dissimilarities

For each missing elements, arcs and nodes, in one of the subgraphs SG_R or SG_S matched, we increase the measure of structural dissimilarity. We have to discuss two types of elements: nodes and arcs.
Basing on the idea that the reference graph contains more solid data and aiming finally an isomorphic match, we can state:

if $n_R > n_S$, there are arcs to be added to the scene graph
if $n_R < n_S$, there are arcs to be deleted from the scene graph

deleted arcs

$$\Delta A_{del}(S_i, S_k) = \begin{cases} n_R - n_S & \text{if } (n_R - n_S) > 0 \\ 0 & \text{else} \end{cases}$$

$$A_{del}(S_k) = \sum_{root}^{k} \Delta A_{del}(S_i, S_k)$$

inserted arcs

$$\Delta A_{ins}(S_i, S_k) = \begin{cases} n_S - n_R & \text{if } (n_R - n_S) < 0 \\ 0 & \text{else} \end{cases}$$

$$A_{ins}(S_k) = \sum_{root}^{k} \Delta A_{ins}(S_i, S_k)$$

We will use these numbers when computing the global dissimilarity or cost.

7.6.3 Weighting and combining dissimilarities

As we mentioned before, we are looking for a search tree expansion algorithm which explores the tree "best first". The term "best" is based on the cost parameter which is a function of the dissimilarities discussed in the previous paragraph.
We combine now the dissimilarities in order to get a single measure of dissimilarity for each subgraph/subgraph match performed which can be used to determine the "best match". Since we would like to compare the cost of states of different levels, we have to introduce also a weighting which depends on the number of elements (nodes, arcs, ...) involved.

With the adjustable weighting parameters

α	=	node dissimilarity weight
β	=	arc attribute dissimilarity weight
γ	=	inserted arc weight
δ	=	deleted arc weight
ε	=	rotation fitting error weight

we write

$$cost(S_k) = \frac{\alpha\, D_{area}(S_k) + \varepsilon\, e_{rotfit}(S_k)}{level(S_k)} + \frac{\beta\, D_{border}(S_k) + \gamma\, A_{ins}(S_k) + \delta\, A_{del}(S_k)}{A_{match}(S_k) + A_{ins}(S_k) + A_{del}(S_k)}$$

The $cost(S_k)$ for a subgraph/subgraph match M_k, represented as states S_k in the search tree, is composed of two parts:

i) the mean dissimilarity on surface areas, weighted by factor α, plus the mean squared rotation fitting error, weighted by factor ε, and
ii) of an arc dissimilarity measure composed of the sum of arc attribute dissimilarities, weighted by factor β, increased by a dissimilarity measure for inserted ($\gamma\, A_{ins}$) and deleted ($\delta\, A_{del}$) arcs divided by the number of arcs involved (matched, inserted, deleted).

7.7. Solutions

In the previous sections, we have discussed how to expand, how to prune and how to stop the search process. In the present one, we define i) what is a solution and ii) discuss the extraction of solutions from the search tree.

7.7.1 Definition

solution

This information is a subset of the information attributed to a state S_i of the search tree, a solution Sol_i corresponds roughly to a state S_i.

A solution Sol_i is defined as the record containing

- M_i the set of matched nodes {...,(NC_{Ri}, NC_{Sj}),...}
- Rot_{Soli} the best fitting rotation
- $cost_{Soli}$ the global $cost_{Soli}$
- additional attributes

7.7.2 Extraction and ordering of solutions

We develop the search tree in such a way that we find a solution of maximal size or a solution of given size (search depth condition, §7.3).
In terms of the search tree, we are looking for leaf states, states which have, by definition, no son states. Attributed to these leaf states S_l are the interesting attributes:

- number of nodes matched $A_{NM}(S_l)$
- number of arcs matched $A_{AM}(S_l)$
- number of arcs inserted in the scene graph $A_{ins}(S_l)$
- number of arcs deleted from the scene graph $A_{del}(S_l)$
- global dissimilarities $D_{area}(S_l)$, $D_{border}(S_l)$
- $cost(S_l)$
- best fitting rotation Rot_{Sl}
- rotation fitting error $e_{rotfit}(S_l)$
- set M_l of node/node correspondences established

Some other implementation dependent conditions have to be satisfied for that a solution Sol_l is accepted.

- min level condition

  In order to have a minimal base for the creation of solutions, we ask for a minimal number of node/node matches, corresponding to a minimal path length in the search tree. The threshold $TH_{minlevel}$ can be set as parameter.

$$level(S_k) \geq TH_{minlevel}$$

- subset condition

Given a first path I, passing by state S_o and leading to the leaf state S_q of level q with its set of matches M_q.
Given a second path II, passing by state S_p and leading to the leaf state S_q' of level q with its set of matches M_q' (fig 7.15).

If both paths I and II exist in a search tree and M_q and M_q' are identical sets of correspondences then S_q and S_q' will be merged, one of the states, either S_p or S_o, will be excluded as a solution.

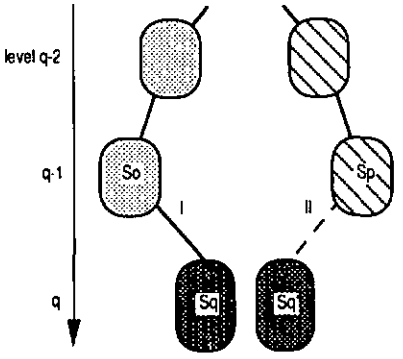


fig 7.15: subset condition

If the child state S_q' from parent state S_p has not been created since one of the expansion conditions of chapter 4 did not hold, S_p will be considered as a leaf state, we will have a solution Sol_p, its set of matches M_p is a subset of M_q of the larger solution Sol_q.

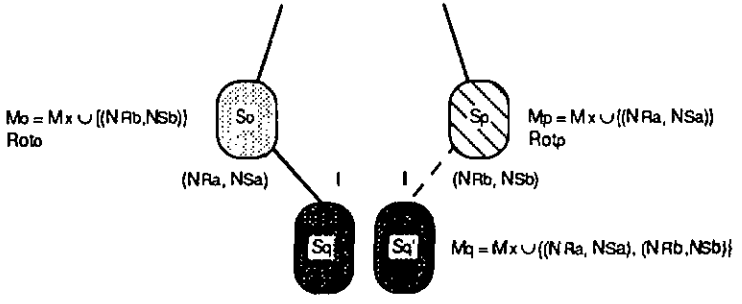


fig 7.16: subset condition, sets of correspondences

Why is it possible that, for two different paths, we get the same final set of matches, but at an intermediate state S_p, for just one of the paths, an expansion condition did not hold ?

The basic reason is that two dissimilarity functions do not steadily grow:

i) the global cost function and
ii) the rotation fitting error

Therefore, it is possible that locally their value lies over the threshold defined, forbidding subsequent extensions using the actual branch of the search tree (i.e. M_q'). Nevertheless, it is pnssible that using another sequence during the establishment of the correspondences leads to M_q, keeping the two functions always below the thresholds.

$M_p = M_x \cup \{(N_{Ra}, N_{Sa})\}$	$\Rightarrow Rot_p$
$M_o = M_x \cup \{(N_{Rb}, N_{Sb})\}$	$\Rightarrow Rot_o$
$M_q = M_x \cup \{(N_{Ra}, N_{Sa}), (N_{Rb}, N_{Sb})\}$	

Table 7.17: correspondences M_i and rotations Rot_i used for the rigidity condition

To prune the set of solutions, we apply the subset condition to couples of solutions:

If the set of matches M_a of solution SL_a is subset of the set of matches M_b of solution SL_b, SL_a will not be considered further.

ordering solutions

Once the solutions are extracted and pruned, they will be ordered "lowest cost" first.

7.7.3 Incomplete solutions

We mentioned before the possibility to introduce global stopping conditions. In order to have some information on partially developed paths which have been considered as "not practically usable" solutions at the moment a "global stop" occured, we add a process which retrieves these "non-recognized" solutions.

incomplete solutions

Incomplete solutions can be found at the leaf nodes S_{Ln} where no attempt for an expansion has been made. If the "incomplete" solutions do not contain a set of correspondences which exists already as a subset in a traditional solution, it will be considered and added to the set of solutions. If the "min level" condition is not fulfilled, it is marked as a "partial solution", else it is considered as the solutions before.

7.8 Results and examples

In the present section, we show the results of the matching program applied to three objects: "CORNER", "PYRAM2" and "CUBE1".
"CORNER" is an artificial example, the "acquired" data has been constructed by myself in order to have some data available which can be checked by hand, an important point for studies on the behavior of the algorithm.
The second one, "PYRAM2", works with real data. A truncated pyramid has been constructed, measured by hand in order to create a model for the reference data base and finally scanned by a 3D data acquisition system.
The third one, "CUBE1", works also with acquired data and has the particularity to be composed of pairwise perpendicular faces.

7.8.1 Example "CORNER"

In the following, we present the results of the matching program, applied to the example "CORNER" and shown in figure 7.18. The example is particularly simple and is presented here in order to demonstrate the program.

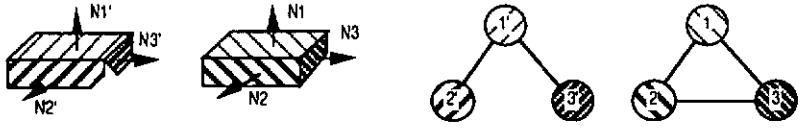


fig 7.18: scene and reference objects (left) and their representations (right)

We start with an arbitrary set of weightings and thresholds, according to the definitions of §7.1 and §7.7, shown in table 7.19. Second, we demonstrate the development of the search tree graphically (fig 7.20). Next, we show the best solutions found as well as some statistics on the search tree and finally we discuss the results.

node dissimilarity weight	α	=	5.00000E-02
arc attribute dissimilarity weight	β	=	4.00000E-02
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	9.00000E-01

$TH_{\Omega} = 20°$	$TH_{maxlevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 2$
$TH_{cost} = 1$	$TH_{node} = 1$	$TH_{arcatt} = 1$	$TH_{rotfit} = \infty$

table 7.19: parameter set

7.8.1.1 Graphical development of the search tree, example "CORNER"

Figure 7.8.3 next page shows the development of the search tree. We use the following notations:

- plain arrows stand for the creation of a child state
- dashed arrows stand for for attempts of creation, abandoned due to the preexistance of an identical child state
- the labels on the arrows indicate the order of creation

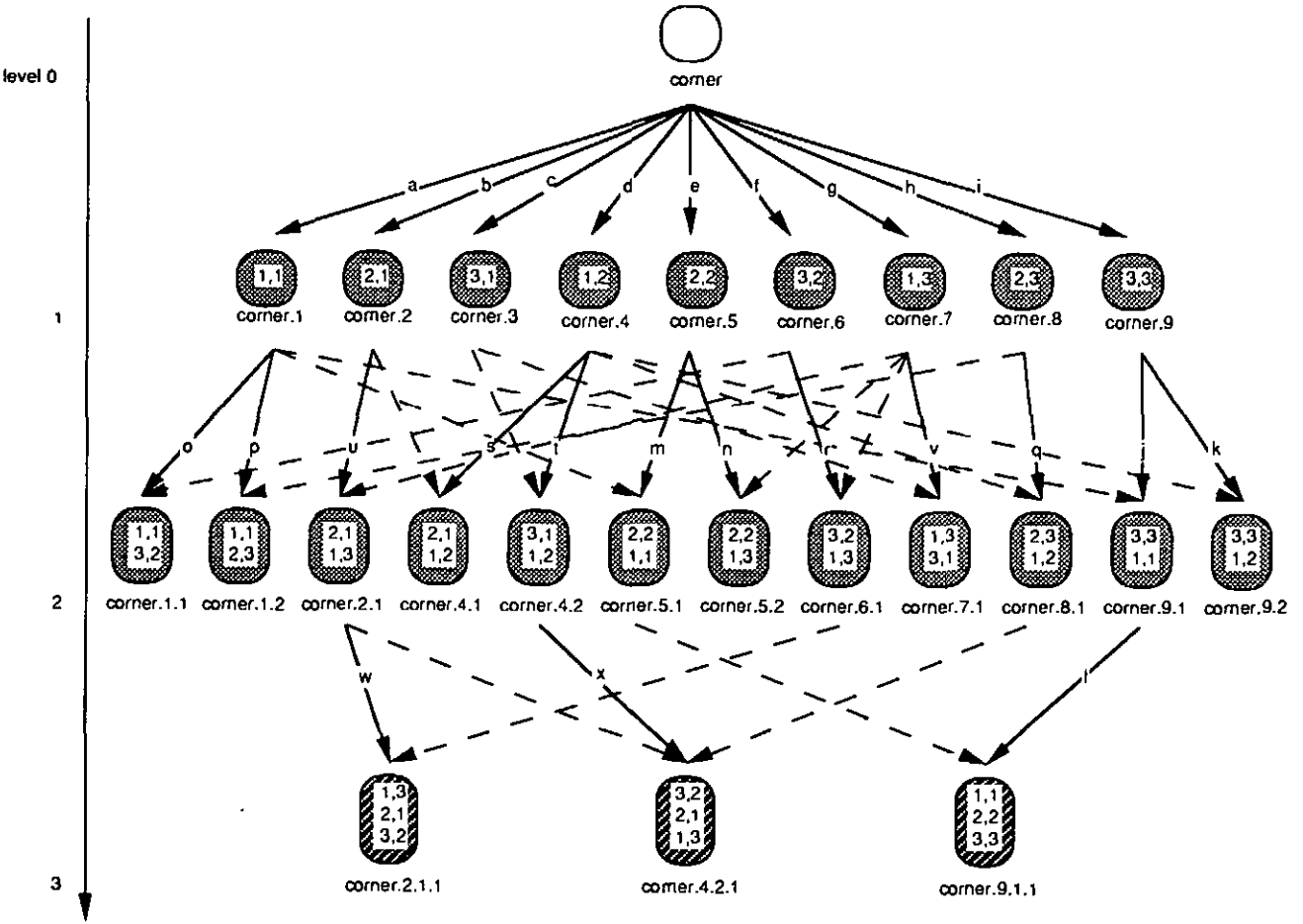


fig 7.20: development of the search tree, example "CORNER"

Starting with the root state at level 0, all states of level 1 are created sequentially. The "best state" found, here state "corner.9", will be expanded to "corner.9.1" and "corner.9.2", then "corner.9.1" to "corner.9.1.1", a first solution is found.
The search continues with the best, not expanded state "corner.5".

7.8.1.2 Solutions

Next, we show the solutions found (fig 7.21). They contain each the solution number (rank), the identifier, the number of nodes matched, the number of arcs matched, inserted or deleted etc ... , the cost, the rotation matrix which maps "best" the reference object to the scene object and finally the core node correspondences established.

```
    1                ; solution_number
corner.9.1.1
    12                        ; state_nb
    3          ; nb of nodes matched
    2           ; nb of arcs matched
    1       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 0.00000E+00              ; node diss
 0.00000E+00          ; arc attr diss
 1.00000E+00          ; arc stru diss
 0.00000E+00               ; rot diss
 1.000  0.000  0.000       R matrix
 0.000  1.000  0.000
 0.000  0.000  1.000
 5.00000E-00                  ; cost
 2'  2 sce / ref             ;matches
 1'  1 sce / ref
 3'  3 sce / ref
`
```

```
    2                ; solution_number
corner.2.1.1
    23                        ; state_nb
    3          ; nb of nodes matched
    2           ; nb of arcs matched
    1       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 1.50000E+00              ; node diss
 8.33333E-01          ; arc attr diss
 1.00000E+00          ; arc stru diss
 4.56997E-08               ; rot diss
 0.000  0.000  1.000       R matrix
 1.000  0.000  0.000
 0.000  1.000  0.000
 4.66667E-02                  ; cost
 3'  2 sce / ref             ;matches
 1'  3 sce / ref
 2'  1 sce / ref
`
```

```
    3                ; solution_number
corner.4.2.1
    24                        ; state_nb
    3          ; nb of nodes matched
    2           ; nb of arcs matched
    1       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 1.50000E+00              ; node diss
 1.00000E+00          ; arc attr diss
 1.00000E+00          ; arc stru diss
 4.56997E-08               ; rot diss
 0.000  1.000  0.000       R matrix
 0.000  0.000  1.000
 1.000  0.000  0.000
 5.00000E-02                  ; cost
 2'  13 sce / ref            ;matches
 3'  11 sce / ref
 1'  12 sce / rel
`
```

fig 7.21: example "corner": solution

Statistics

number of states investigated	:	24
CPU time on MicroVAX 3400 [s]	:	0.57
number of solutions	:	3
best solution at state nb	:	12
cost ratio solution 2 / solution 1	:	9.3

Discussion

Despite of a marginal tuning of the parameters, the best match found is the correct one. "Mirror solutions", solutions which are symmetric with respect to a planar surface ("mirror") do not appear, this thanks to the rigidity condition which allows to take care of the relative orientation of the object faces.

7.8.2 Example "PYRAM2"

Now, we pass to a more general example, "PYRAM2", a truncated pyramid. For this example, the scene data has been acquired by a 3D data acquisition

system, using structured light [Mai89]. Before we discuss any results, we present i) the reference object in its different representations, and ii) the acquired scene object in its respective representations. Next, we, as the supervisor, give the "correct result", i.e. the node/node correspondences we should find in an optimal match between scene data and reference data. This "correct result" is determined directly by observation of the pyramid position.
Finally, we discuss the results obtained for different configurations of the parameters.

7.8.2.1 Reference object "PYRAM2_REF"

Initially, we have the geometric description of the object, a truncated pyramid (fig 7.22). Using the rules discussed before, we extract the high level representation PFRG (fig 7.23, left). A subset of the information contained in this high level representation, the orientation of the faces involved is shown in the extended needle map (fig 7.23, right).

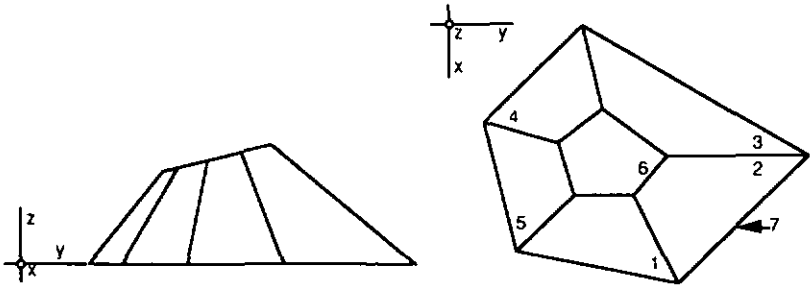


fig 7.22: reference object "PYRAM2_REF": front view (left) and top view

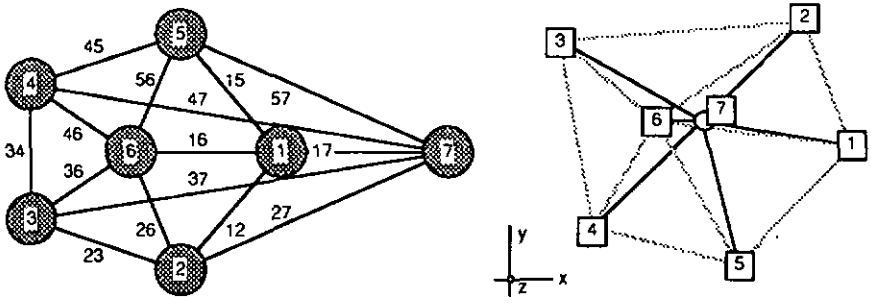


fig 7.23: "PYRAM2_REF": high level PFRG representation(left) and extended n

7.8.2.2 Acquired data for object "PYRAM2"

Next we discuss the acquired scene data. The first figure (fig 7.24) shows the region/border representation as it is delivered by the region growing

program which segments the acquired data in planar patches. The subsequent preprocessing step reduces the information, e.g. it eliminates the "null-regions" in figure 7.24 and generates the PFRG (fig 7.25) as well as an extended needle map (fig 7.26).

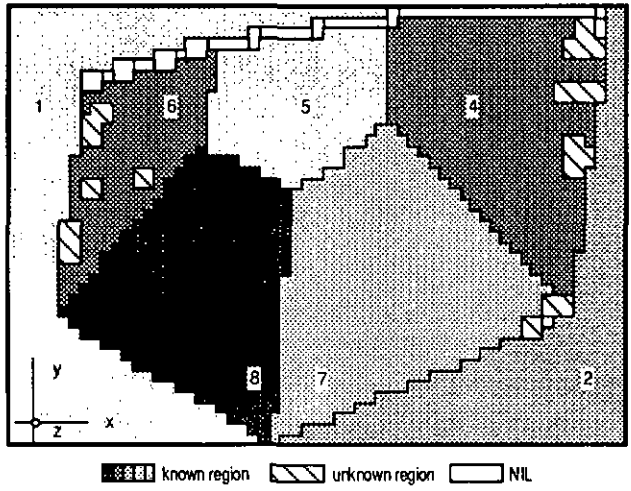


fig 7.24: "PYRAM2": region / border representation

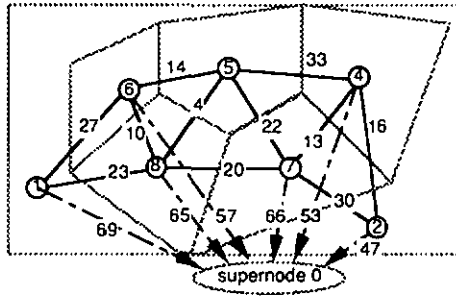


fig 7.25: "PYRAM2": high level graph representation PFRG

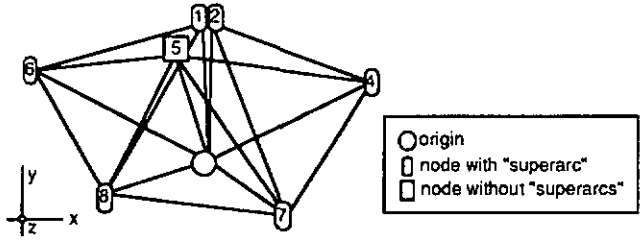


fig 7.26 extended needle map

7.8.2.3 The "correct solution"

scene node	↔	reference node
6	↔	4
8	↔	5
7	↔	1
4	↔	2
5	↔	6

table 7.27: correspondences scene object (top left) / reference object (top right)

The supervisor comparison of scene and reference data leads to the result of table 7.27.

7.8.2.4 Experiences and results

We will now do some experiences and check the influence of different parameter configurations on the solutions found. The aim is to tune the parameters in order to find a "best" solution which is i) correct and ii) distinct from the other ones. In order to determine the attributes needed and their tuning, we have made the following experiences with example "PYRAM2":

i) surface area and border length attributes
ii) orientation attributes only
iii) surface area attributes and weighed orientation attributes
iv) all attributes

7.8.2.4.1 Surface area and border length attributes

For this first test we use the parameter settings of table 7.28 here below. The thresholds and weightings are set so that the surface orientation attribute has no influence on the correspondence search.

node dissimilarity weight	α	=	5.00000E-02
arc attribute dissimilarity weight	β	=	4.00000E-02
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	0.00000E+00

$TH_{\Omega} = 180°$	$TH_{maxlevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 2$
$TH_{cost} = 1$	$TH_{node} = 1$	$TH_{arcan} = 1$	$TH_{rotfit} = \infty$

table 7.28: parameter set

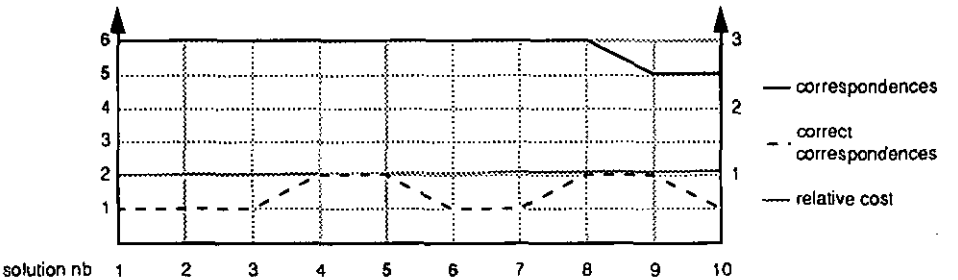
```
   1                ; solution_number
pyram2a.1.1.2.1.2.1
   3606                    ; state_nb
    6          ; nb of nodes matched
    7          ; nb of arcs matched
    3       ; nb of sce arcs inserted
    1       ; nb of sce arcs deleted
 8.91548E-01               ; node diss
 1.08684E+00          ; arc attr diss
 4.00000E+00          ; arc stru diss
1.83104E-01                ; rot diss
-0.971  0.137 -0.198     ; R matrix
-0.181 -0.958  0.224
-0.159  0.253  0.954
1.67044E-02                  ; cost
 5   6 sce / ref              ;matches
 2   4 sce / ref
 7   3 sce / ref
 8   2 sce / ref
 6   5 sce / ref
 1   1 sce / ref
.
```

```
   2                ; solution_number
pyram2a.28.5.2.7.7.3
   2206                    ; state_nb
    6          ; nb of nodes matched
    6          ; nb of arcs matched
    4       ; nb of sce arcs inserted
    3       ; nb of sce arcs deleted
 9.89202E-01               ; node diss
 7.81591E-01          ; arc attr diss
 7.00000E+00          ; arc stru diss
2.91975E-01                ; rot diss
 0.078 -0.954 -0.289     ;R matrix
 0.920 -0.042  0.390
-0.384 -0.296  0.874
1.67153E-02                  ; cost
 6   2 sce / ref              ;matches
 2   5 sce / ref
 7   3 sce / ref
 4   1 sce / ref
 5   6 sce / ref
 8   4 sce / ref
.
```

```
   3                ; solution_number
pyram2a.23.8.3.3.2.1
   3161                    ; state_nb
    6          ; nb of nodes matched
    6          ; nb of arcs matched
    4       ; nb of sce arcs inserted
    1       ; nb of sce arcs deleted
 6.87429E-01               ; node diss
 9.52608E-01          ; arc attr diss
 5.00000E+00          ; arc stru diss
3.24702E-01                ; rot diss
-0.621  0.780  0.078     ;R matrix
-0.716 -0.604  0.350
 0.320  0.162  0.934
1.67416E-02                  ; cost
 1   1 sce / ref              ;matches
 6   5 sce / ref
 5   6 sce / ref
 4   2 sce / ref
 7   3 sce / ref
 2   4 sce / ref
.
```

table 7.29: solutions for"PYRAM2": surface area and border length attributes

Statistics



number of states investigated	:	4318	
CPU time on MicroVAX 3400 [s]	:	3600	STOPPING CONDITION !
number of solutions	:	1013	
best solution at state nb	:	3606	
cost ratio solution 2 / solution 1	:	1.0007	

Discussion

All three solutions of table 7.29 are incorrect. Neglecting the orientation attributes, and, therefore, the rigidity constraint leads to a search tree which is very big. It could be restricted using conditions on the attribute dissimilarities but it is quite difficult to define them for two reasons. First, the scene data itself is prone to error and gives, therefore, rise to erroneous dissimilarity measures and second, the establishment of a condition is quite ambiguous, there is no more direct interpretation of the meaning of the condition.

In the example above, we stopped the search after 1 hour of CPU (MicroVAX 3400) with a search tree size of 4318 states.

7.8.2.4.2 orientation attributes only

For this second test, we limit ourselves to the normal vector attribute and the structural information, the influence of the other attributes is neglected (table 7.30).

node dissimilarity weight	α	=	0.00000E+00
arc attribute dissimilarity weight	β	=	0.00000E+00
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	9.00000E-01

$TH_{\Omega} = 20°$	$TH_{maxdevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 2$
$TH_{cost} = 1$	$TH_{node} = 1$	$TH_{arcalt} = 1$	$TH_{rotfit} = 1$

table 7.30: parameter set

This test corresponds to the "gaussian image" approaches realized by Horn [Hor84a], [Hor84b], Brou[Bro84], Ikeuchi[Ike83] and others. It is limited, in principle, to convex objects.

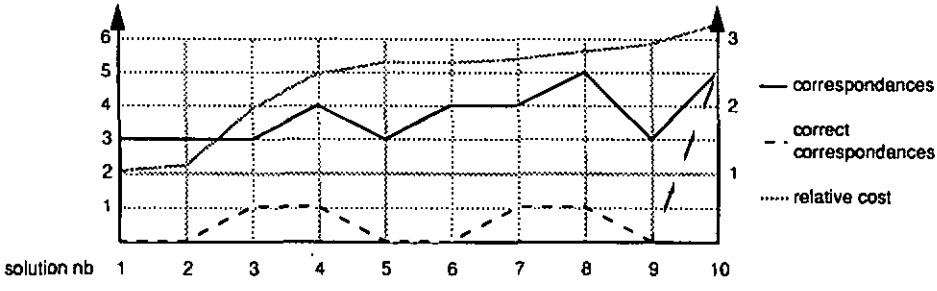
```
    1                ; solution_number
pyram2b.38.5.1
   381               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0       ; nb of sce arcs inserted
     0       ; nb of sce arcs deleted
 1.30197E+00           ; node diss
 8.88415E-01         ; arc attr diss
 0.00000E+00         ; arc stru diss
 1.71970E-03           ; rot diss
  0.440 -0.252  0.862 ; R matrix
 -0.095  0.941  0.324
 -0.893 -0.224  0.390
 5.15909E-04                ; cost
   7   4 sce / ref          ;matches
   2   3 sce / ref
   4   6 sce / ref
```

```
    2                ; solution_number
pyram2b.40.9.1
   398               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0       ; nb of sce arcs inserted
     0       ; nb of sce arcs deleted
 6.41164E-01           ; node diss
 9.71845E-01         ; arc attr diss
 0.00000E+00         ; arc stru diss
 1.85213E-03           ; rot diss
  0.489 -0.452 -0.746; R matrix
  0.816 -0.059  0.572
 -0.302 -0.890  0.341
 5.55639E-04                ; cost
   1   5 sce / ref          ;matches
   8   4 sce / ref
   6   6 sce / ref
```

```
    3                ; solution_number
pyram2b.42.12.1
   400               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0       ; nb of sce arcs inserted
     0       ; nb of sce arcs deleted
 8.68898E-01           ; node diss
 1.22306E+00         ; arc attr diss
 0.00000E+00         ; arc stru diss
 3.30354E-03           ; rot diss
  0.170  0.889 -0.426 ; R matrix
 -0.942  0.020 -0.335
 -0.289  0.458  0.841
 9.91061E-04                ; cost
   1   3 sce / ref          ;matches
   6   4 sce / ref
   8   6 sce / ref
```

table 7.31: solutions for "PYRAM2": orientation attributes only

Statistics



Statistics (continued)

number of states investigated	:	588
CPU time on MicroVAX 3400 [s]	:	182.8
number of solutions	:	90
best solution at state nb	:	381
cost ratio solution 2 / solution 1	:	1.07

Discussion

The result, represented in table 7.31, is insufficient and false. The first correct correspondence, in bold typesetting, appears in solution 3, this beside an erroneous correspondence. This confirms the idea that a method for the determination of object orientations and for object recognition, based on surface orientations only, is insufficient. The facts illustrated by these results lead to an extension of the idea, to the extended gaussian images EGI [Hor84a], [Hor84b] where the influence of the surface area attribute is also considered. The application of the EGI is for sure limited but, for the completeness of the experience, we use this approach in the following chapter, too.
Another important point not discussed yet is the influence of the background. First, we are confronted with the segmentation problem and, second, we have to take into account the possibility that scene background segments could be brought into correspondence with reference object faces.

7.8.2.4.3 Surface area attributes and weighted orientation attributes

This time we used, beside the structural information, the information contained in an EGI, the surface area and surface normal attributes. The contribution of the normals to the rotation matrix R is proportional to the underlying surface (see also chapter 7.5).

node dissimilarity weight	α	=	5.00000E-02
arc attribute dissimilarity weight	β	=	0.00000E+00
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	9.00000E-01

$TH_{\Omega} = 20°$	$TH_{maxlevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 2$
$TH_{cost} = 1$	$TH_{node} = 1$	$TH_{arcatt} = 1$	$TH_{rotfit} = 1$

table 7.32: parameter set

Statistics

number of states investigated	:	584
CPU time on MicroVAX 3400 [s]	:	165.7
number of solutions	:	78
best solution at state nb	:	189
cost ratio solution 2 / solution 1	:	2.17

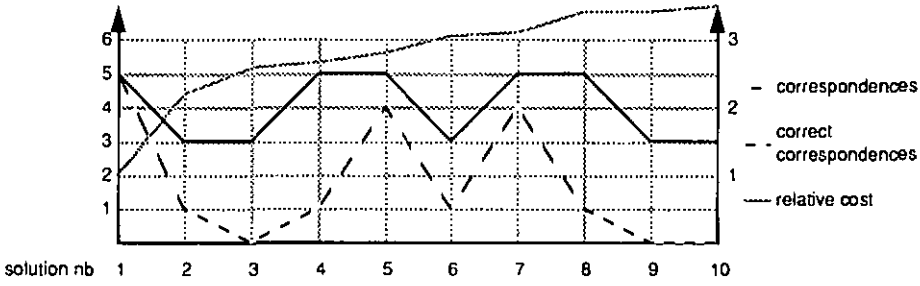
```
    1                ; solution_number
pyram2c.39.4.1.2.1
   189               ; state_nb
     5          ; nb of nodes matched
     7          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
3.21032E-01            ; node diss
7.75557E-01        ; arc attr diss
0.00000E+00        ; arc stru diss
6.00954E-03            ; rot diss
 0.644  0.765 -0.013   ;R matrix
-0.637  0.547  0.543
 0.423 -0.341  0.839
4.29203E-03                ; cost
 6   4 sce / ref          ;matches
 8   5 sce / ref
 7   1 sce / ref
 4   2 sce / ref
 5   6 sce / ref
.
```

```
    2                ; solution_number
pyram2c.35.14.1
   109               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
4.08221E-01            ; node diss
1.20572E+00        ; arc attr diss
0.00000E+00        ; arc stru diss
8.46568E-03            ; rot diss
 0.630  0.110 -0.769   ; R matrix
 0.478  0.725  0.496
 0.612 -0.679  0.404
9.34338E-03                ; cost
 6   6 sce / ref          ;matches
 1   1 sce / ref
 8   5 sce / ref
.
```

```
    3                ; solution_number
pyram2c.28.3.1
   244               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
6.41164E-01            ; node diss
1.16093E+00        ; arc attr diss
0.00000E+00        ; arc stru diss
1.71329E-03            ; rot diss
 0.480 -0.446 -0.756   ; R matrix
 0.822 -0.072  0.564
-0.306 -0.892  0.332
1.12001E-02                ; cost
 6   6 sce / ref          ;matches
 1   5 sce / ref
 8   4 sce / ref
.
```

table 7.33: solutions for "PYRAM2": surface area / weighted orientation attributes

Ststistics (continued)



Discussion

The result, represented in figure 7.33 is already much better. The correct result lies on place 1 and the cost of the second solution is 117% higher. As will be confirmed in the example "CUBE1" bellow, that this approach is successful in the example of the pyramid because the relative orientations are well distinguishable. If we want to distinguish faces with similar relative orientations, we will be confronted again with the weighting and threshold definition problems for the remaining attributes.

7.8.2.4.4 All attributes

We will show on the next page the results for "PYRAM2" using all attributes. We expect that the relevance of the attribute "borderlength" is not too high. Nevertheless, we check it too in order to know its influence. In fact, we would like to show that it can be used where other attributes, like the face orientation attribute, are not sufficient.

node dissimilarity weight	α	=	5.00000E-02
arc attribute dissimilarity weight	β	=	4.00000E-02
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	9.00000E-01

$TH_{\Omega} = 20°$ $TH_{maxlevel} = \infty$ $TH_{minlevel} = 3$ $TH_{ND} = 2$
$TH_{cost} = 1$ $TH_{node} = 1$ $TH_{arcatt} = 1$ $TH_{rotfit} = 1$

table 7.34: parameter set

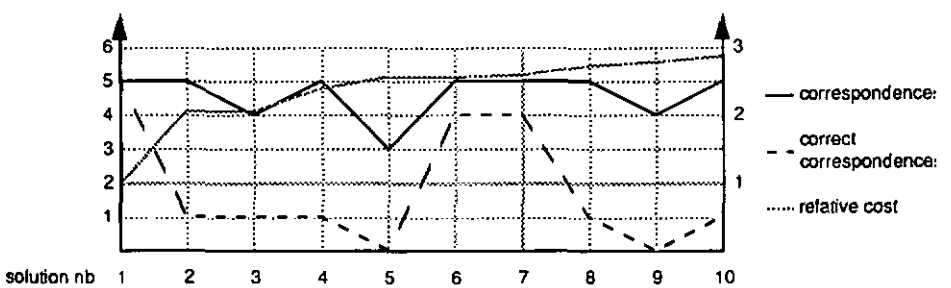
```
1                    ; solution_number
pyram2d.39.4.1.1.1
  214                ; state_nb
    5          ; nb of nodes matched
    7          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
3.21032E-01            ; node diss
7.75557E-01        ; arc attr diss
0.00000E+00        ; arc stru diss
6.00942E-03            ; rot diss
 0.644  0.765 -0.013   ;A matrix
-0.637  0.547  0.543
 0.423 -0.341  0.839
8.72377E-03                ; cost
 8  5 sce / ref            ;matches
 6  4 sce / ref
 7  1 sce / ref
 4  2 sce / ref
 5  6 sce / ref
.
```

```
2                    ; solution_number
pyram2d.28.5.3.2.1
  282                ; state_nb
    5          ; nb of nodes matched
    7          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
9.88582E-01            ; node diss
1.05503E+00        ; arc attr diss
0.00000E+00        ; arc stru diss
8.96565E-03            ; rot diss
 0.971 -0.241 -0.010   ;A matrix
 0.211  0.828  0.519
-0.117 -0.506  0.854
1.75284E-02                ; cost
 6  3 sce / ref            ;matches
 4  1 sce / ref
 7  5 sce / ref
 5  6 sce / ref
 8  4 sce / ref
.
```

```
3                    ; solution_number
pyram2d.39.9.1.1
  315                ; state_nb
    4          ; nb of nodes matched
    5          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
1.08943E+00            ; node diss
6.23813E-01        ; arc attr diss
0.00000E+00        ; arc stru diss
7.25380E-03            ; rot diss
 0.253 -0.967 -0.007   ;A matrix
 0.770  0.197  0.607
-0.586 -0.160  0.795
2.02404E-02                ; cost
 8  3 sce / ref            ;matches
 4  5 sce / rel
 7  4 sce / ref
 5  6 sce / ref
.
```

table 7.35 solutions for "PYRAM2": all attributes

Statistics



Statistics (continued)

number of states investigated	:	584
CPU time on MicroVAX 3400 [s]	:	176.3
number of solutions	:	78
best solution at state nb	:	214
cost ratio solution 2 / solution 1	:	2.00

Discussion

Again, solution 1 meets the "correct result" and the second solution has approximately double the cost of the first solution, the result is comparable to the one we found in chapter 7.8.2.4.3 .
Solutions 2 and 3 are also reasonable candidates: they stand for matches of the scene object rotated by one face left and right respectively about an axis parallel to the surface normal of reference face 6 (fig 7.36).

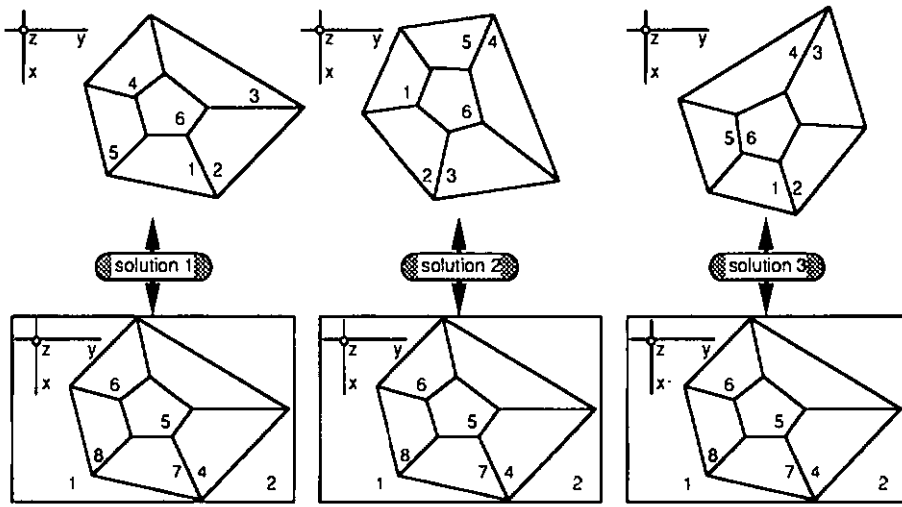


fig 7.36: solutions 1, 2, and 3; reference object (top) and scene object (bottom)

7.8.3 Example "CUBE1"

In the example just before, we have seen that the surface orientation attribute plays an important role. If several angles between faces are identical, the efficiency of the orientation attribute is reduced and we have to use additional attributes to overcome the ambiguity. We test if the attributes we have chosen before are indicated, this using the example "CUBE1".

7.8.3.1 Reference object "CUBE1_REF"

As before, we show in figure 7.37 the reference object (left), the high level representation (middle) and the extended needle map (right).

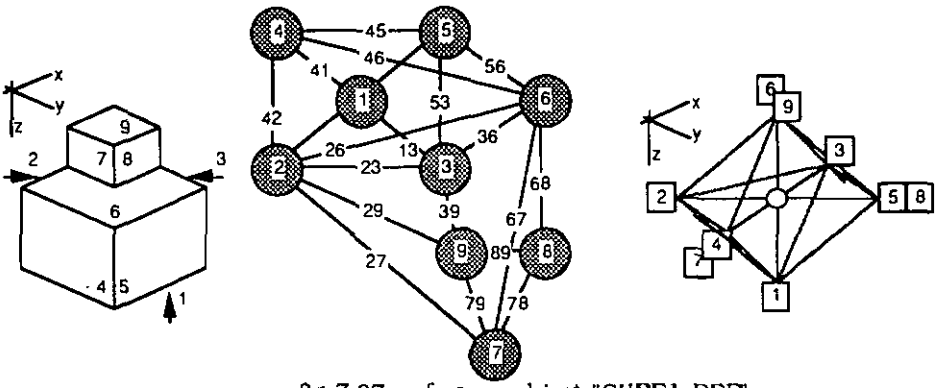


fig 7.37: reference object "CUBE1_REF"

7.8.3.2 Acquired data for object "CUBE1"

As in the previous examples, we show the high level representation PFRG, the extended needle map, and the region/border representation.

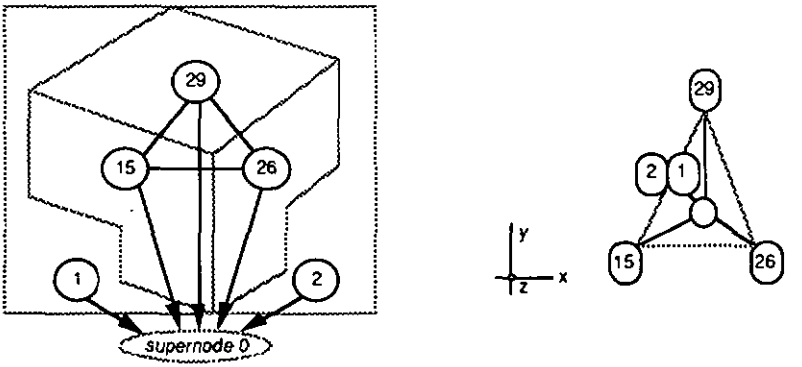


fig 7.38: "CUBE1": PFRG representation and extended needle map

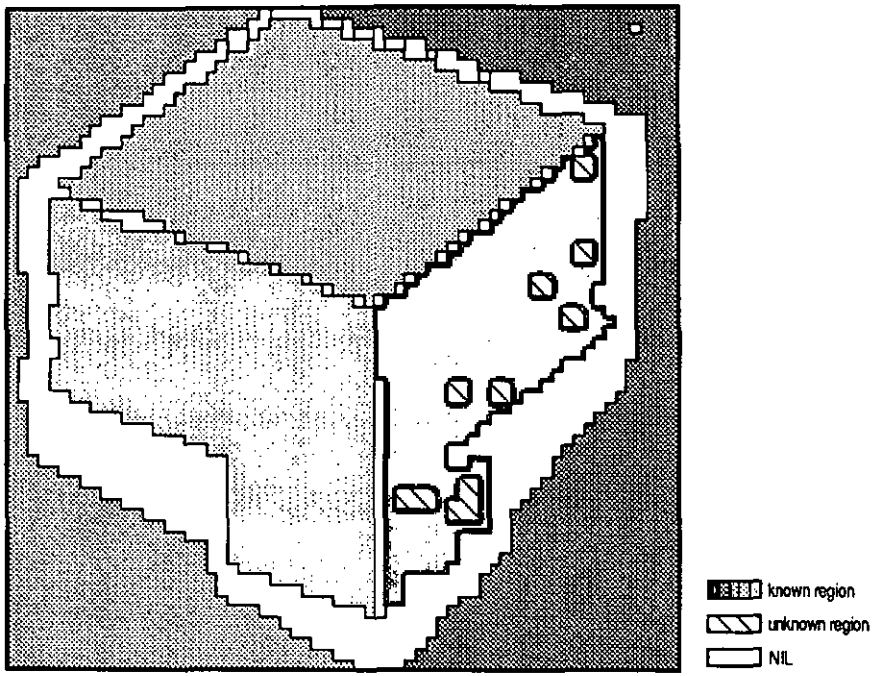


fig 7.38: "CUBE1": region border representation

We observe again that the edges of the object, and specially the jump edges between object faces and the background, give rise to a huge number of small, badly defined regions we cannot use in the further matching. They are eliminated using the high level preprocessing.

The structural information as well as the information on the face orientations is not very relevant. In order to match correctly, we are forced to rely on the correctly weighted dissimilarity measures of the remaining attributes: surface area and border length.

7.8.3.3 The "correct result"

reference node	↔	scene node
2	↔	15
3	↔	26
1	↔	29

table 7.40: "correct matching" for example "CUBE1"

A supervisor comparison of scene and reference data leads to the result of table 7.40.

7.8.3.4 Experiences and results

Since all edges have angles of approximatively 90°, the use of the orientation attribute is limited to the distinction of the different edge types (convex, concave, flat). For the recognition task itself, the information contained in this attribute will not be sufficient.

In the following, we show a parameter set which allows us to find a correct solution. Next, we will try to use the same set again for our previous example "PYRAM2" in order to show that it is not too particular.

7.8.3.4.1 A parameter set for "CUBE1"

After a few tests, we have found the parameter set of table 7.41. The weights are, except the value for the arc attribute dissimilarity, identical to the values of the last test with "PYRAM2" using the parameters of table 7.41, the thresholds have been adapted slightly.

node dissimilarity weight	α	=	5.00000E-02
arc attribute dissimilarity weight	β	=	1.00000E-01
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ϵ	=	9.00000E-01

$TH_{\Omega} = 20°$	$TH_{maxdevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 6$
$TH_{cost} = 0.1$	$TH_{node} = 1$	$TH_{arcatt} = 1$	$TH_{rotfit} = 0.01$

table 7.41: parameter set

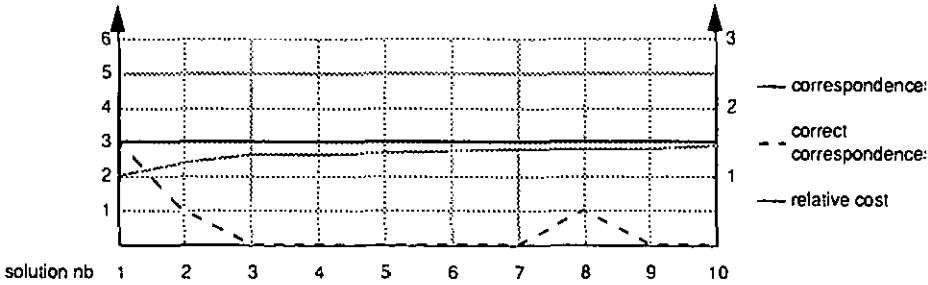
```
    1                ; solution_number
cube1.5.8.1
   136               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 1.02960E+00           ; node diss
 8.87839E-02       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.10173E-05           ; rot diss
  0.812  0.583 -0.023     ; R matrix
 -0.290  0.437  0.852
  0.506 -0.685  0.524
 2.01258E-02               ; cost
 26   3 sce / ref          ;matches
 15   2 sce / ref
 29   1 sce / ref
.
```

```
    2                ; solution_number
cube1.5.6.1
   108               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 7.71061E-01           ; node diss
 3.36468E-01       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.11830E-05           ; rot diss
 -0.812 -0.583 -0.022 ; R matrix
  0.290 -0.437  0.851
 -0.506  0.685  0.524
 2.40730E-02               ; cost
 26   4 sce / ref          ;matches
 15   5 sce / ref
 29   1 sce / ref
.
```

```
    3                ; solution_number
cube1.15.8.1
   149               ; state_nb
     3          ; nb of nodes matched
     3          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 1.07675E+00           ; node diss
 2.05798E-01       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.13030E-05           ; rot diss
 -0.023 -0.812 -0.583     ; R matrix
  0.851  0.290 -0.437
  0.524 -0.506  0.685
 2.48121E-02               ; cost
 26   2 sce / ref          ;matches
 15   1 sce / ref
 29   3 sce / ref
.
```

table 7.42: solutions 1-3 for "CUBE1": all attributes

Statistics



number of states investigated	:	225
CPU time on MicroVAX 3400 [s]	:	28.1
number of solutions	:	60
best solution at state nb	:	22
cost ratio solution 2 / solution 1	:	1.19

Discussion

The correspondences have been established correctly, this thanks to the "border length" attribute which allowed to match correctly the long border between reference regions R_2 and R_3. This shows also the use of the preprocessing step "eliminate border regions", described before.

7.8.3.4.2 Application of the same parameter set to "PYRAM2"

Applying the same parameter set to example "PYRAM2" delivers the same correspondences as found in chapter 7.8.2.4.4 .

7.9 Conclusions

In the present part, we have treated the inexact high level matching task. Given two high level object representations in the form of attributed relational graphs, we extract the most promising (partial) matches so that we can establish hypotheses on the scene object, a similarity measure between the visible parts of both reference and scene objects, as well as the knowledge about the rotation the reference object has to undergo in order to map best to the scene data.

We treat the subject of graph matching as a tree search problem. First, we defined the meaning of a search tree state in our object matching context and discuss the tree extension tactic, using a very weak rule which allows to accept a wide spectrum of topological inexactitude.

Second, we discussed the order of extension. With the idea in mind to find rapidly good solutions which lead to sets of correspondences between scene and reference faces of high similarity, we use a "best first search". This point takes its importance when a best solution should be found in a limited time. Of course, it would be possible to search in the very same manner an optimal solution, but the huge computational effort would forbid this approach.

Next, we focussed on the tree pruning conditions, an important aspect when computation time and complexity are critical. The most important condition we use is the rigidity condition. It takes advantage from i) the rigidity of the objects involved and ii) from the robustness of the orientation information extracted from the scene. Despite the good results obtained with this condition, additional tree pruning conditions based on other graph attributes had to be added to deal with cases where the first condition is not sufficient: e.g. for man-made objects where perpendicular and parallel faces are usual. We follow the idea that the addition of supplementary matches has to be avoided when this leads to an important increase in the overall dissimilarity or when the elements matched are very different themselves.

Finally, we extracted the solutions corresponding to leaf states of the search tree which fulfill some minimal, application specific conditions. The result is a set of solutions, ordered following the overall cost, where each one stands for a hypothesis on a reference object/scene object match, accompanied by a measure of similarity as well as a rotation R which best maps the reference object to the scene object.

The results of chapter 7.8 show first that the preprocessing steps, discussed before, is of utmost importance in order to extract a high level representation of high precision. Second, once this condition is fulfilled, we showed that a given tuning of the parameter set allows to extract good results even for different objects, an important point when the results of the matchings with different objects have to be combined in a recognition task.

8. Verification

8.1 High level verification

8.2 Conclusions

8. Verification

Verification is the process which checks if hypotheses on results obtained previously are most probable to be correct or not. In our application, we verify the hypotheses on the correspondences between scene and reference objects extracted.

We can distinguish different levels of verification, corresponding to the different levels of representation (fig 8.1).

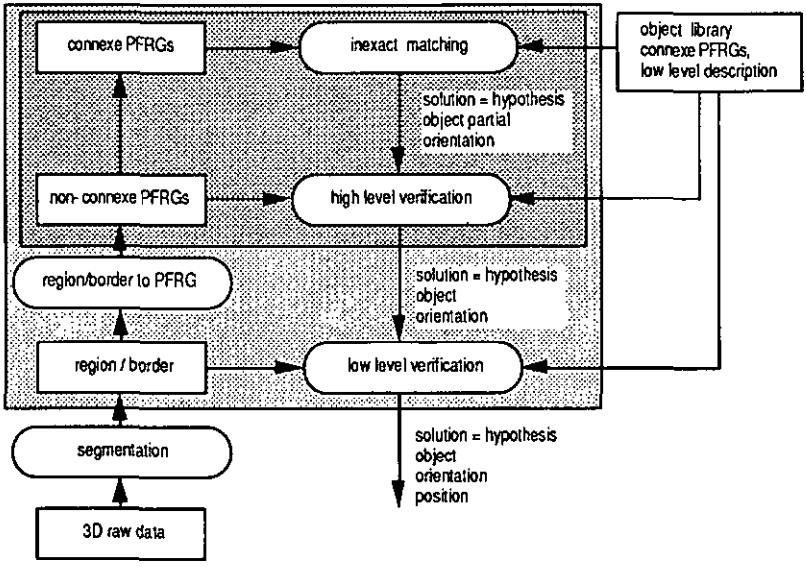


fig 8.1: levels of representation and verification

In the following, we will limit ourselves to the high level verification for PFRGs for the inexact matching method presented in chapter 7.

8.1 High level verification

Having done the inexact matching at the level of the PFRGs, we attempt a first verification at this same level. Before doing so, we remind the data available and, because search tree pruning limits the means for doing verification, discuss the interdependencies between tree pruning and verification.

A second part is devoted to another interpretation of high level verification, the combination of solutions obtained by the match of connexe PFRGs. This extension should allow to merge partial solutions, due to non-connexe PFRGs representing separated scenes, in order to get bigger solutions.

8.1.1 Available data for high level verification

Scene and reference data is represented in the form of PFRGs, the solutions in the form of records, both discussed earlier.

PFRG

An attributed relational graph PFRG contains

i) structural information (nodes, arcs)
ii) node attributes
    - face area
    - face orientation
iii) arc attribute
    - border length

solution

A solution Sol_i is defined as a record containing

i) the set of corresponding pairs of nodes	$M_i = \{...,(NC_{Ri}, NC_{Sj}),...\}$
ii) the best fitting rotation	Rot_{Soli}
iii) the global $cost_{Soli}$	$cost_{Soli}$
iv) other results	

We have also access to indications on structural modifications realized (arcs inserted and deleted resp.). These informations are a subset of the data attributed to each state S_i of the solution search tree.

8.1.2 Connexe PFRGs

Using the connectivity condition introduced before (§7) as a pruning criterion in the search tree, our solutions are limited to matches of connexe (sub-)graphs. If by any reason a scene is represented by several connexe PFRGs, a solution at this level will represent a partial scene object/reference object match only.

verification

Unfortunately, with the high level representation PFRG (§3.4) and the pruning conditions introduced (§7.4), we cannot do any useful verification since there is no more data available which had not been used before in search tree pruning conditions.

search tree pruning vs. verification

The two processes are complementary in the following sense.

> The more data we use in an intelligent manner for tree pruning conditions, the less will be usefully available for subsequent verifications.

In fact, search tree pruning conditions can also been understood as some kind of on-line verification. Having seen the huge complexity of computation (exponential) of a non-pruned search tree, it is rather clear that it is much

more interesting to invest a big amount of information in search tree pruning conditions rather than in off-line verifications (fig 8.2 and fig 8.3).

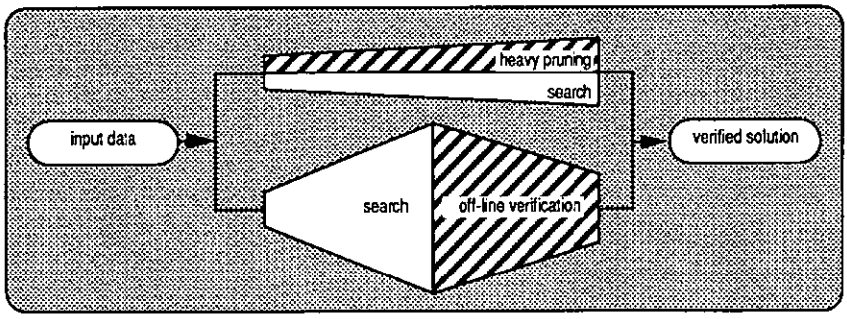


fig 8.2: on-line verification (top), off-line verification (bottom)

graph	- connectivity criterion - node degree criterion
orientation	- rigidity criterion - rotation fitting error criterion - global cost criterion - visibility criterion
surface area	- node attribute dissimilarity criterion - global cost criterion
border length	- arc attribute dissimilarity criterion - global cost criterion

fig 8.3: PFRG elements used in tree pruning conditions (§7)

conclusions

The only features we could use for high level verifications are

i) the exact structure of the graphs,
ii) the modifications applied to the scene (sub-)graph (inserted / deleted arcs) in order to match it isomorphicly with the reference (sub-)graph, and
iii) an indication on the potentially visible faces compared with the faces matched.

We stated before that only an inexact (sub-)graph matching system could fulfill the requirements of an object recognition system. For the well known reasons (natural scene data, acquisition incertitude, high level data reduction, etc ..) we cannot ask for isomorphic matching. Therefore, there is no sense either to use the exact graph description or the indications on inserted / deleted arcs.

The last indication, the one on potentially visible faces, is not useful for verification purposes either since self occlusion, occlusion due to other objects as well as a partial acquisition of a scene object can lead to missing matches.

8.1.3 Non connexe PFRGs

Another interpretation of "high level verification" is the following. We attempt to combine matches based on connexe graphs in order to find bigger ones.

expanded solutions

Once a solution, and hence a hypothesis on the rotation between reference and scene object has been found, we can use it in order to complete the match:

> The reference object's PFRG is mapped to the scene, predicting the attributes to find in the scene object's PFRG. Then, for all non matched scene nodes, we can check if they fulfill a weaker similarity criterion with respect to the mapped reference nodes.

This allows to retrieve node/node correspondences which have not been established during the search process, this due to the connectivity condition and which could not furnish uniquely a hypothesis on a solution, this due to a lack of data.

Unfortunately, the reliability of the correspondences obtained by this expansion is not very high: the amount of data on which they are based is too small. An additional verification at a lower level will be necessary.

compatible solutions

At a higher level, the same reasoning can be used to find solutions Sol_B, Sol_C, ... which are compatible with an initial solution Sol_A.

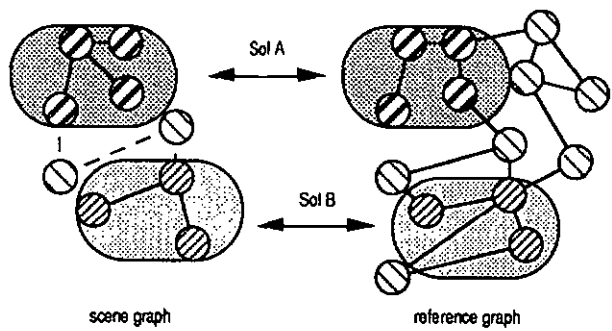


fig 8.4: compatible solutions: matched nodes(dotted) and non-matched nodes (hatched)

We look for sets of solutions C = (Sol_A, Sol_B, ...) whose elements are compatible in the following sense:

> No matched node of neither the reference nor the scene graph appears simultaneously in solutions Sol_A, Sol_B, ...

If the compatibility condition is fulfilled, and if the resulting rotations are similar, we can establish the hypothesis that the solutions stand for the same object and that these solutions can be merged.

If the resulting hypothesis on the rotations are too much different, we have to investigate the presence of some other objects. This point will be discussed later.

The reliability of the results of the present approach is much higher than the one for the previous step "expansion", this due to the bigger amount of data involved in in the basic solutions.

8.2 Conclusions

Using the matching criterions "rigidity" and "visibility", we have inherently used the most important high level verification informations of the connexe PFRGs as pruning criterions during the establishment of the correspondences themselves. The remaining data is less robust and, therefore, only of limited use.

At the next lower level, a compatibility test has been proposed, allowing to merge PFRGs. The result is a solution standing for multiple scene subgraphs matched with multiple reference subgraphs without incompatibility.
This step is very useful for the combination of scene representations, and their respective matches, which have been separated due to non-idealities (occlusion, shading, ...). Additionally, it is a first step towards a multiple object recognition system.

9. Object recognition

9. Object recognition

Until now we worked with a single reference object and a single scene object. The most important problems have been discussed, in particular the problems of

i) partial view
ii) noisy data
iii) background / object distinction

This allowed us to extract an ordered list L of hypotheses H_i on i) the rotation Rot_i to apply in order to map best the reference object to the scene object and ii) the set of correspondences M_i between scene and reference faces.

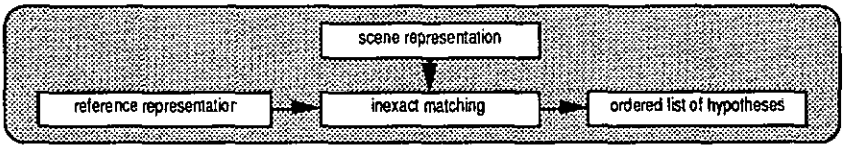


fig 9.1: inexact matching

This step is certainly very important, but not sufficient for the object recognition task where multiple reference representations have to be used, the resulting hypotheses merged and ordered following the global cost and where some more general reflections on the final result must be done.

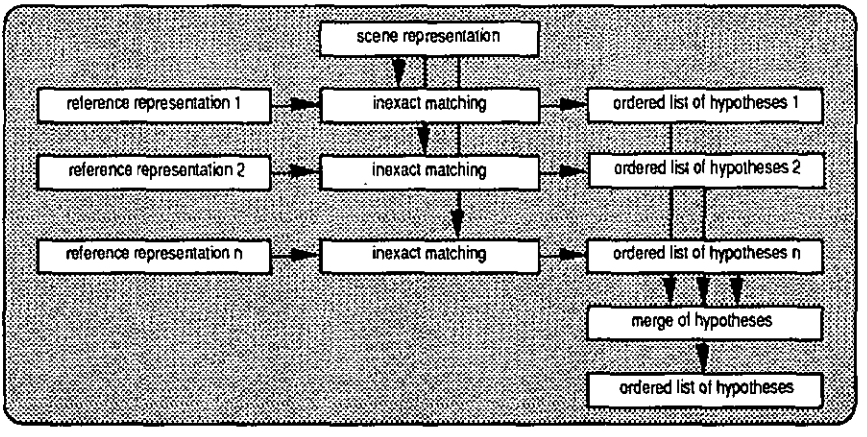


fig 9.2: object recognition

In the present chapter, we present some experimental results concerning object recognition and object separation.

The experiences are listed next page.

multiple reference objects, single scene object

First, we experiment with classical object recognition (§9.1). A high level representation, standing for acquired data of a scene containing a single objects, is brought one-by-one into correspondence with high level representations of reference objects.

A comparison of the solutions and the attributed costs found allows us to check if we are able to recognize simple objects using the present algorithm.

multiple separated scene objects, single reference object

Second, we use a scene representation which contains two non-overlapping objects (§9.2). Again, we apply the inexact matching program and try to find out if it is possible to determine which of both scene objects corresponds to a given reference object.

multiple non-separated scene objects, single reference object

Third, we enhance the difficulty of the task (§9.3). The scene contains now two overlapping objects. Now, we would like to determine to what extent the object recognition still works, and, if it is possible, to recognize the "frontier" between both objects.

multiple scene objects and object separation

A last problem to discuss is how to manage the merge of several lists of hypotheses on (partial) solution so that multiple objects in the scene can be separated and recognized (§9.4).

To terminate we have drawn some conclusions.

9.1 Multiple reference objects, single scene object

The aim is to finally find, in the first position of the final solution list, the correctly recognized reference object with correctly established matches.

We use again the scene data examples as in part VII and match them one by one with the reference objects, applying always the same set of paramcters for weighting and pruning purposes. Then, we compare the solutions for the match of a single scene object with different reference objects and order them following the global cost.

9.1.1 Example "PYRAM2"

We use the parameter set we used finally in chapter 7, shown in table 9.3.

node dissimilarity weight	α	=	0.00000E+00
arc attribute dissimilarity weight	β	=	0.00000E+00
inserted arc weight	γ	=	1.00000E-02
deleted arc weight	δ	=	1.00000E-02
rotation fitting error weight	ε	=	8.00000E-01

$TH_{\Omega} = 20°$	$TH_{maxdevel} = \infty$	$TH_{minlevel} = 3$	$TH_{ND} = 6$
$TH_{cost} = 0.1$	$TH_{node} = 1$	$TH_{arcatt} = 1$	$TH_{rotfit} = 1$

table 9.3: parameter set

```
   1              ; solution_number
pyram2f.39.4.1.1.1
   278                    ; state_nb
    5          ; nb of nodes matched
    7            ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
3.21032E-01                ; node diss
7.75557E-01          ; arc attr diss
0.00000E+00          ; arc stru diss
6.00942E-03                ; rot diss
 0.644  0.765 -0.013     ;R matrix
-0.637  0.547  0.543
 0.423 -0.341  0.839
1.53714E-02                    ; cost
 8   5 sce / ref              ;matches
 6   4 sce / ref
 7   1 sce / ref
 4   2 sce / ref
 5   6 sce / ref
*
5 faces matched of  6 visible
```

```
   2              ; solution_number
pyram2f.39.12.2.2.1
   370                    ; state_nb
    5          ; nb of nodes matched
    7            ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
9.88582E-01                ; node diss
1.05503E+00          ; arc attr diss
0.00000E+00          ; arc stru diss
8.96565E-03                ; rot diss
 0.971 -0.241 -0.010     ;R matrix
0.211  0.828  0.519
-0.117 -0.506  0.854
2.65716E-02                    ; cost
 6   3 sce / ref              ;matches
 4   1 sce / ref
 8   4 sce / ref
 7   5 sce / ref
 5   6 sce / rel
*
5 faces matched of  6 visible
```

```
   3              ; solution_number
pyram2f.39.9.1.1
   351                    ; state_nb
    4          ; nb of nodes matched
    5            ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
1.08943E+00                ; node diss
6.23813E-01          ; arc attr diss
0.00000E+00          ; arc stru diss
7.25380E-03                ; rot diss
 0.253 -0.967 -0.007     ;R matrix
0.770  0.197  0.607
-0.586 -0.160  0.795
2.77262E-02                    ; cost
 8   3 sce / ref              ;matches
 4   5 sce / ref
 7   4 sce / ref
 5   6 sce / ref
*
4 faces matched of  6 visible
```

fig 9.4: results for scene "PYRAM2" and references "PYRAM2_REF" and "CUBE1_REF"

Discussion

Ordering the solutions from both inexact matches leads to a list showed partially in figure 9.4. For the ten first solutions found, the match between the scene object "PYRAM2" and its corresponding reference object "PYRAM2_REF" delivers always a lower global cost than the match with the

cube, the second reference object.: the object "PYRAM2" has been recognized.

9.1.2 Example "CUBE1"

Now we check the scene object "CUBE1" against the two reference objects "PYRAM2_REF" and "CUBE1_REF".

```
    1                ; solution_number
cube1f.5.8.1
   136                    ; state_nb
    3          ; nb of nodes matched
    3           ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 1.02960E+00              ; node diss
 8.87839E-02          ; arc attr diss
 0.00000E+00          ; arc stru diss
 2.10173E-05              ; rot diss
  0.812  0.583 -0.023     ;R matrix
 -0.290  0.437  0.852
  0.506 -0.685  0.524
 2.01258E-02                 ; cost
 26   3 sce / ref           ;matches
 15   2 sce / ref
 29   1 sce / ref
*
 3 faces matched of  3 visible
```

```
    2                ; solution_number
cube1f.5.6.1
   108                    ; state_nb
    3          ; nb of nodes matched
    3           ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 7.71061E-01              ; node diss
3.36468E-01          ; arc attr diss
 0.00000E+00          ; arc stru diss
 2.11830E-05              ; rot diss
 -0.812 -0.583 -0.022     ;R matrix
  0.290 -0.437  0.851
 -0.506  0.685  0.524
 2.40730E-02                 ; cost
 26   4 sce / ref           ;matches
 15   5 sce / ref
 29   1 sce / ref
*
 3 faces matched of  5 visible
```

```
    3                ; solution_number
cube1f.15.8.1
   149                    ; state_nb
    3          ; nb of nodes matched
    3           ; nb of arcs matched
    0       ; nb of sce arcs inserted
    0       ; nb of sce arcs deleted
 1.07675E+00              ; node diss
 2.05798E-01          ; arc attr diss
 0.00000E+00          ; arc stru diss
 2.13030E-05              ; rot diss
 -0.023 -0.812 -0.583     ;R matrix
  0.851  0.290 -0.437
  0.524 -0.506  0.685
 2.48121E-02                 ; cost
 26   2 sce / ref           ;matches
 15   1 sce / ref
 29   3 sce / ref
*
 3 faces matched of  3 visible
```

fig 9.5: results for scene "CUBE1" and references "PYRAM2_REF" and "CUBE1_REF"

Discussion

Here, the result is even clearer: the match between the scene object "CUBE1" and the reference object "PYRAM2_REF", using the parameter set from before, did not deliver any solution. Hence, the best solution corresponds to the best solution of match "CUBE1" / "CUBE1_REF" and is, this following the experiences made before, correct. The object "CUBE1" has been recognized.

9.2. Multiple separated scene objects, single reference object

In the following, a scene "CUPYRS" (fig 9.6), composed of two non-overlapping objects, is to be treated. The data has been created artificially, concatenating the two scenes "PYRAM2" and "CUBE1".

9.2.1 Scene data

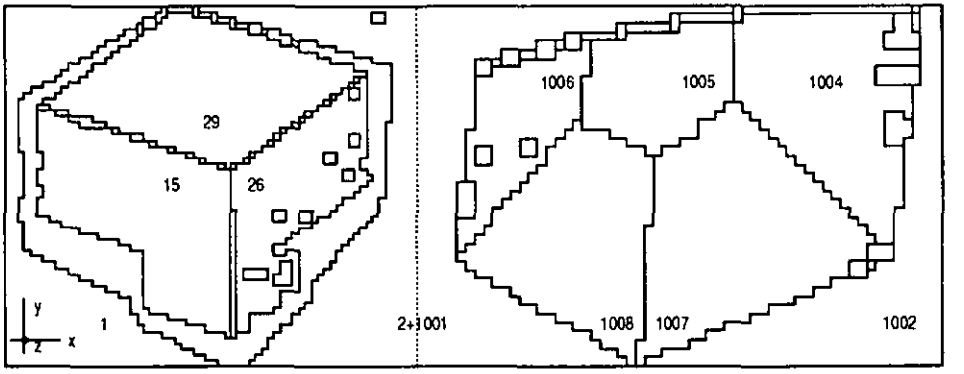


fig 9.6: acquired scene "CUPYRS", two separate objects

Figure 9.6 above shows the acquired scene after region growing, figure 9.5 below the corresponding high level representation in the form of an attributed relational graph.

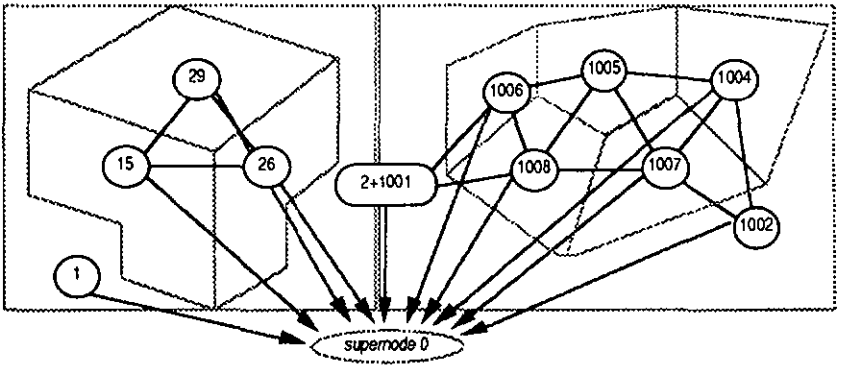


fig 9.7: high level scene representation "CUPYRS"

We apply the matching process on the high level scene representation "CUPYRS" (fig 9.7), first with reference object "PYRAM2_REF", then with reference object "CUBE1_REF".

9.2.2 Results for separated scene objects, reference object "PYRAM2_REF"

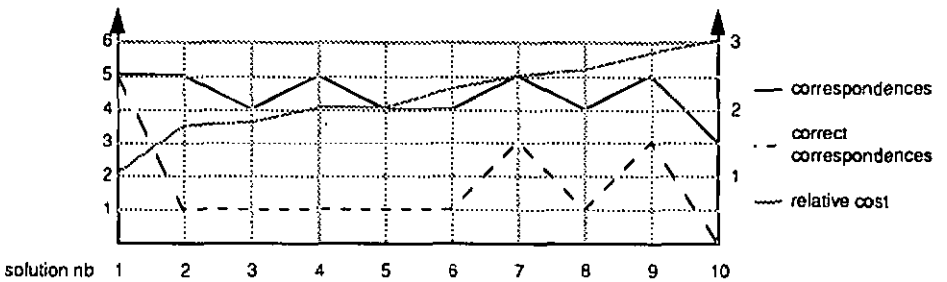
```
    1                   ; solution_number
cpp .63.4.1.1.1
   271                  ; state_nb
     5          ; nb of nodes matched
     7          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 3.21032E-01            ; node diss
 7.75557E-01        ; arc attr diss
 0.00000E+00        ; arc stru diss
 6.00942E-03            ; rot diss
  0.644  0.765 -0.013   ;R matrix
 -0.637  0.547  0.543
  0.423 -0.341  0.839
 1.53714E-02                ; cost
1008   5 sce / ref         ;matches
1006   4 sce / ref
1007   1 sce / ref
1004   2 sce / ref
1005   6 sce / ref
*
 5 faces matched of  6 visible
```

```
    2                   ; solution_number
cpp .63.12.2.2.1
   348                  ; state_nb
     5          ; nb of nodes matched
     7          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 9.88582E-01            ; node diss
 1.05503E+00        ; arc attr diss
 0.00000E+00        ; arc stru diss
 8.96565E-03            ; rot diss
  0.971 -0.241 -0.010   ;R matrix
  0.211  0.828  0.519
 -0.117 -0.506  0.854
 2.65716E-02                ; cost
1006   3 sce / ref         ;matches
1004   1 sce / ref
1008   4 sce / ref
1007   5 sce / ref
1005   6 sce / ref
*
 5 faces matched of  6 visible
```

```
    3                   ; solution_number
cpp .63.9.1.1
   323                  ; state_nb
     4          ; nb of nodes matched
     5          ; nb of arcs matched
     0      ; nb of sce arcs inserted
     0      ; nb of sce arcs deleted
 1.08943E+00            ; node diss
 6.23813E-01        ; arc attr diss
 0.00000E+00        ; arc stru diss
 7.25380E-03            ; rot diss
  0.253 -0.967 -0.007   ;R matrix
  0.770  0.197  0.607
 -0.586 -0.160  0.795
 2.77262E-02                ; cost
1008   3 sce / ref         ;matches
1004   5 sce / ref
1007   4 sce / ref
1005   6 sce / ref
*
 4 faces matched of  6 visible
```

fig 9.8: solutions for reference object "PYRAM2_REF"

Statistics



number of states investigated	:	630	584
CPU time on MicroVAX 3400 [s]	:	207.8	178.8
number of solutions	:	78	78
best solution at state nb	:	271	278
cost ratio solution 2 / solution 1	:	1.7	1.7

Discussion

With the usual parameter set we get the results shown in figure 9.8. In the statistics, the left column indicates the results for the "CUPYRS" scene, and the right column the results for the pyramid alone.

The result is ok, the best match is with the pyramid and the established correspondences are still all correct. Thanks to the pruning criterions "rigidity" and "connectivity", there are no cross-matches to the cube either.

9.2.3 Results for separated scene objects, reference object "CUBE1"

Using the same parameter set again, we match the high level scene representation with the reference object "CUBE1_REF".

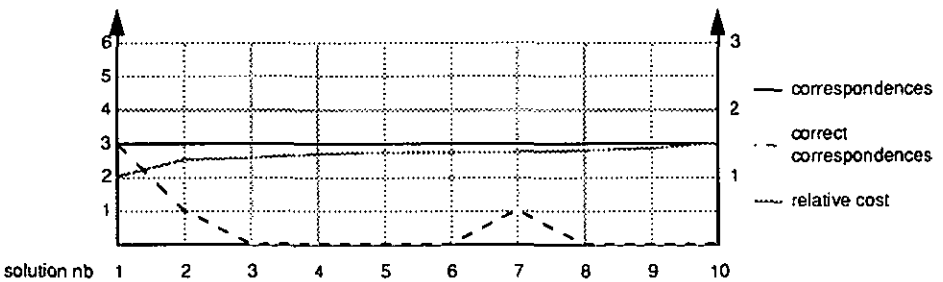
```
    1                ; solution_number
cpc .5.8.1
   343               ; state_nb
     3       ; nb of nodes matched
     3        ; nb of arcs matched
     0     ; nb of sce arcs inserted
     0     ; nb of sce arcs deleted
 1.02960E+00          ; node diss
 8.87839E-02       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.10173E-05          ; rot diss
  0.812  0.583 -0.023      ;R matrix
 -0.290  0.437  0.852
  0.506 -0.685  0.524
 2.01258E-02              ; cost
 26   3 sce / ref         ;matches
 15   2 sce / ref
 29   1 sce / ref
.
 3 faces matched of  3 visible
```

```
    2                ; solution_number
cpc .5.8.1
   283               ; state_nb
     3       ; nb of nodes matched
     3        ; nb of arcs matched
     0     ; nb of sce arcs inserted
     0     ; nb of sce arcs deleted
 7.71061E-01          ; node diss
 3.36468E-01       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.11830E-05          ; rot diss
 -0.812 -0.583 -0.022 ;R matrix
  0.290 -0.437  0.851
 -0.506  0.685  0.524
 2.40730E-02              ; cost
 26   4 sce / ref         ;matches
 15   5 sce / ref
 29   1 sce / ref
.
 3 faces matched of  5 visible
```

```
    3                ; solution_number
cpc .27.8.1
   372               ; state_nb
     3       ; nb of nodes matched
     3        ; nb of arcs matched
     0     ; nb of sce arcs inserted
     0     ; nb of sce arcs deleted
 1.07675E+00          ; node diss
 2.05798E-01       ; arc attr diss
 0.00000E+00       ; arc stru diss
 2.13030E-05          ; rot diss
 -0.023 -0.812 -0.583 ;R matrix
  0.851  0.290 -0.437
  0.524 -0.506  0.685
 2.48121E-02              ; cost
 26   2 sce / ref         ;matches
 15   1 sce / ref
 29   3 sce / ref
.
 3 faces matched of  3 visible
```

fig 9.9: solutions for reference object "CUBE1_REF"

Statistics



number of states investigated	:	709	225
CPU time on MicroVAX 3400 [s]	:	208.9	28.6
number of solutions	:	78	60
best solution at state nb	:	343	136
cost ratio solution 2 / solution 1	:	1.2	1.2

Discussion

The result confirms the statements of the discussion before: the result is again ok, but the difference in relative cost has become much smaller and, therefore, the risk of confusion between the solutions is higher.

9.3 Multiple non-separated scene objects, single reference object

The present test is much more delicate. The scene "CUPYRO" (fig 9.10), containing a cube which occludes partially a pyramid, has to undergo the object recognition process.

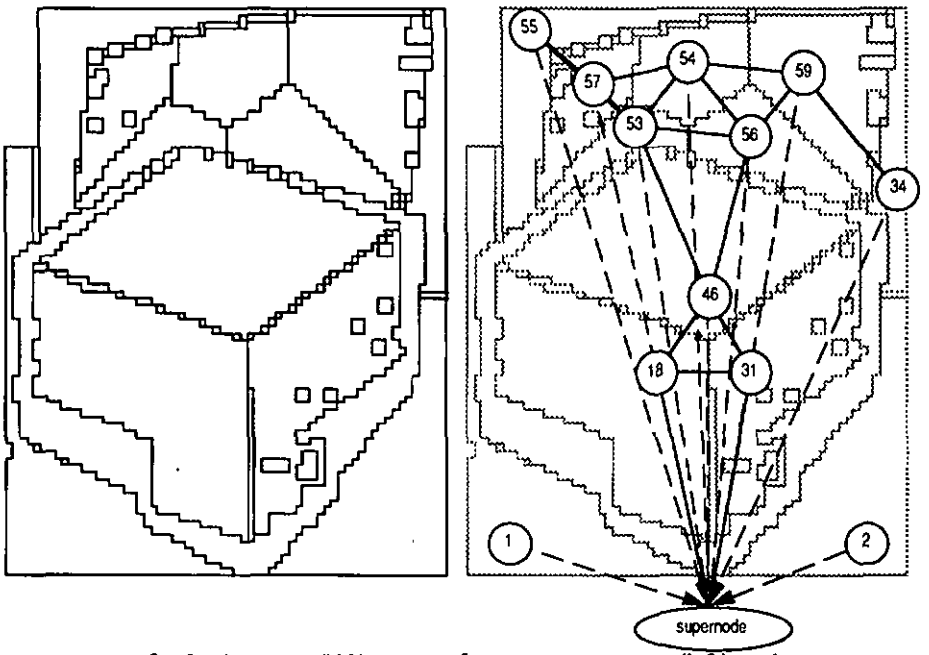


fig 9.10: scene "CUPYRO" after region growing (left) and the extracted high level representation (right)

9.3.1 Scene vs. reference object "CUBE1_REF"

First, we apply the reference objects "CUBE1_REF", using the usual parameter set.

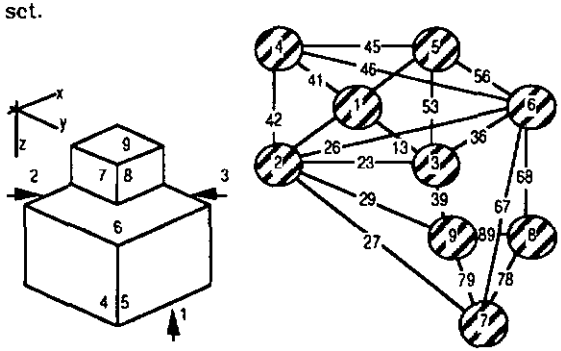


fig 9.11: reference object "CUBE1_REF"

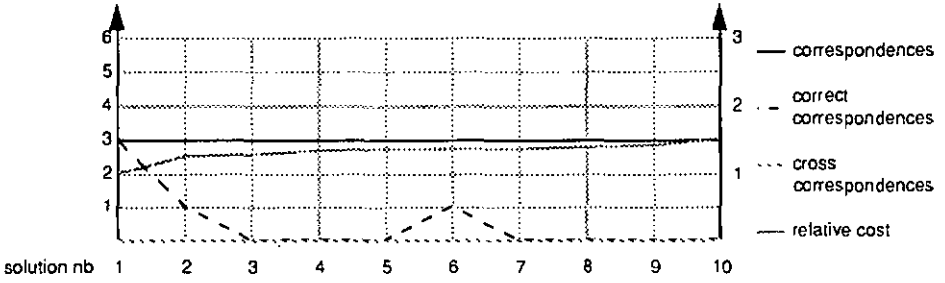
```
    1                ; solution_number
cupyr_c.6.14.1
    289              ; state_nb
    3          ; nb of nodes matched
    3          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
1.01357E+00          ; node diss
8.16662E-02      ; arc attr diss
0.00000E+00      ; arc stru diss
8.65213E-06          ; rot diss
 0.812  0.583 -0.022   ; R matrix
-0.289  0.435  0.853
 0.507 -0.686  0.522
1.96176E-02          ; cost
 31   3 sce / ref     ;matches
 18   2 sce / ref
 46   1 sce / ref
.
3 faces matched of  3 visible
```

```
    2                ; solution_number
cupyr_c.6.10.1
    209              ; state_nb
    3          ; nb of nodes matched
    3          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
7.55167E-01          ; node diss
3.29350E-01      ; arc attr diss
0.00000E+00      ; arc stru diss
 9.05468E-06         ; rot diss
-0.812 -0.583 -0.022  ;R matrix
 0.299 -0.435  0.853
-0.507  0.686  0.522
2.35672E-02          ; cost
 31   4 sce / ref     ;matches
 18   5 sce / ref
 46   1 sce / ref
.
3 faces matched of  5 visible
```

```
    3                ; solution_number
cupyr_c.30.16.1
    303              ; state_nb
    3          ; nb of nodes matched
    3          ; nb of arcs matched
    0      ; nb of sce arcs inserted
    0      ; nb of sce arcs deleted
1.06403E+00          ; node diss
2.12572E-01      ; arc attr diss
0.00000E+00      ; arc stru diss
8.64158E-06          ; rot diss
-0.023 -0.812 -0.583  ; R matrix
 0.853  0.289 -0.436
 0.522 -0.507  0.686
2.48222E-02          ; cost
 31   2 sce / ref     ;matches
 18   1 sce / rel
 46   3 sce / rel
.
3 faces matched of  3 visible
```

fig 9.12: results "CUPYRO" vs. "CUBE1_REF"

Statistics



number of states investigated	:	829
CPU time on MicroVAX 3400 [s]	:	358
number of solutions	:	104
best solution at state nb	:	289
cost ratio solution 2 / solution 1	:	1.2

Discussion

The topmost hypothesis shows matches exclusively with the cube in the scene and the hypothesis with the lowest cost is again the correct one. Having a look at the statistics, we can see that, for the first ten hypotheses, no cross- matches appear, this thanks to the rigidity constraint.

A look at the number of matched and non-matched scene faces allows to establish the hypothesis that other objects are present in the scene.

9.3.2 Scene vs. reference object "PYRAM2_REF"

Now, we apply the reference objects "PYRAM2_REF", using the same parameter set.

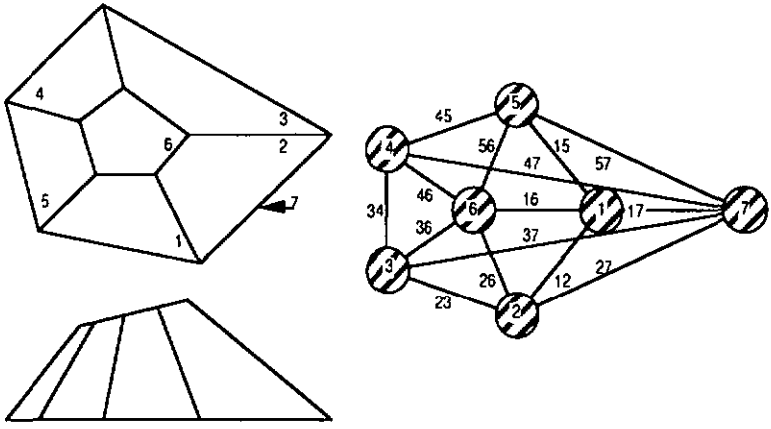


fig 9.13: reference object "PYRAM2_REF"

```
    1                ; solution_number
cupyr_p.68.3.1.1.1
    312                      ; state_nb
     5          ; nb of nodes matched
     7            ; nb of arcs matched
     0        ; nb of sce arcs inserted
     0        ; nb of sce arcs deleted
 1.46342E+00              ; node diss
 1.34588E+00          ; arc attr diss
 0.00000E+00          ; arc stru diss
 7.02529E-03               ; rot diss
 0.652  0.758 -0.017      ; R matrix
-0.635  0.558  0.534
 0.414 -0.337  0.845
 3.51257E-02                  ; cost
 53   5 sce / ref             ;matches
 57   4 sce / ref
 56   1 sce / ref
 59   2 sce / ref
 54   6 sce / ref
*
 5 faces matched of   6 visible
```

```
    2                ; solution_number
cupyr_p.68.11.1.1
    316                      ; state_nb
     4          ; nb of nodes matched
     4            ; nb of arcs matched
     1        ; nb of sce arcs inserted
     0        ; nb of sce arcs deleted
 1.45639E+00              ; node diss
 4.97001E-01          ; arc attr diss
 1.00000E+00          ; arc stru diss
 1.09376E-02               ; rot diss
-0.724 -0.681 -0.109      ; R matrix
 0.490 -0.619  0.614
-0.485  0.391  0.782
 3.55908E-02                  ; cost
 56   3 sce / ref             ;matches
 57   2 sce / ref
 59   4 sce / ref
 54   6 sce / ref
*
 4 faces matched of   6 visible
```

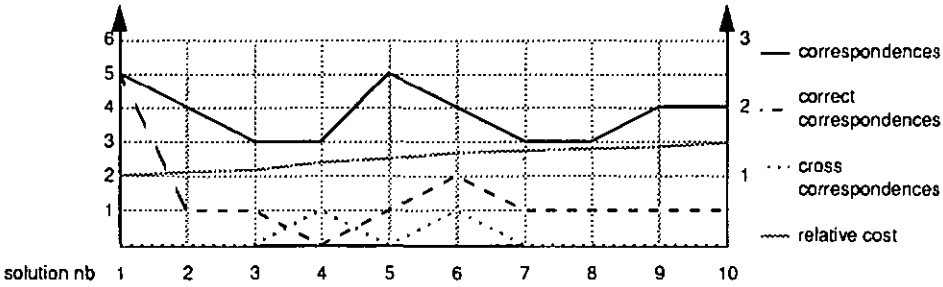
```
    3                ; solution_number
cupyr_p.68.17.2
    451                      ; state_nb
     3          ; nb of nodes matched
     3            ; nb of arcs matched
     0        ; nb of sce arcs inserted
     0        ; nb of sce arcs deleted
 1.15201E+00              ; node diss
 4.42384E-01          ; arc attr diss
 0.00000E+00          ; arc stru diss
 1.59872E-02               ; rot diss
-0.994  0.059 -0.093      ; R matrix
-0.098 -0.861  0.499
-0.051  0.505  0.862
 3.87425E-02                  ; cost
 53   2 sce / ref             ;matches
 57   1 sce / ref
 54   6 sce / ref
*
 3 faces matched of   6 visible
```

fig 9.14: results "CUPYRO" vs. "PYRAM2_REF"

Statistics

number of states investigated	:	688
CPU time on MicroVAX 3400 [s]	:	242.7
number of solutions	:	104
best solution at state nb	:	289
cost ratio solution 2 / solution 1	:	1.02

Statistics (continued)



Discussion

Since the object is partially occluded by the cube, the matching results are poorer, but still, the correct match is on the first place. The number of cross matches is again very small.

A new problem is now how to combine the results of both single-reference-object/multiple-scene-objects matchings. Pure ordering of the hypotheses following the global cost will not be sufficient.

9.4 Multiple scene objects and object separation

A next problem to resolve is how to manage the merge of several lists of hypotheses on (partial) solution so that multiple objects in the scene can be separated and recognized.

terms

solution $Sol_i = \{M_i, Rot_{Soli}, Cost_{Soli}, ...\}$

A solution is defined as a record, containing the set of matched nodes $M_i = \{..., NC_{Ri}, NC_{Sj}, ...\}$, the best fitting rotation Rot_{Soli}, the global matching cost $Cost_{Soli}$ and other results of the inexact matching process.

list of solutions $L_k = \{..., Sol_{ik}, ...\}$

A list of solutions L_k contains the different hypotheses on solutions for the inexact match k between a reference representation and a scene representation.

merged list of solutions $ML = \cup L_k$

A merged list ML of solutions contains all the hypotheses on solutions for the inexact matches done between a scene representation and a set of reference representation.

set of compatible solutions $SCS_q = \{..., Sol_{nm}, ...\}$ where $Sol_{nm} \in ML$

A set of compatible solutions SCS_q contains sets of solutions compatible Sol_{nm}, issued from one or more lists of solutions.

9.4.1 Compatibility of partial solutions

compatibility of partial solutions in one-one scene reference representation matches

First, we limit ourselves to just one list of solutions. Let us recall a point discussed before in § 8.1.3.2, the criterion of compatibility between partial solutions out of one and the same list of solutions L_k. The compatibility has to be understood in the following sense:

> If no matched node neither of the reference nor the scene graph appears simultaneously in solutions Sol_A, Sol_B, ... and the corresponding rotations Rot_{SolA}, Rot_{SolB}, ... are similar, the solutions Sol_A, Sol_B, ... are said to be compatible and can be merged.

The compatible solutions can be replaced in the ordered list of solutions by one merged solution. This case stands for partial submatches, resulting in subsolutions which can be explained by non-idealities in the scene graph, leading to situations where the connectivity condition was not fulfilled: the scene graph, representing a single object, had been separated (fig 9.15).

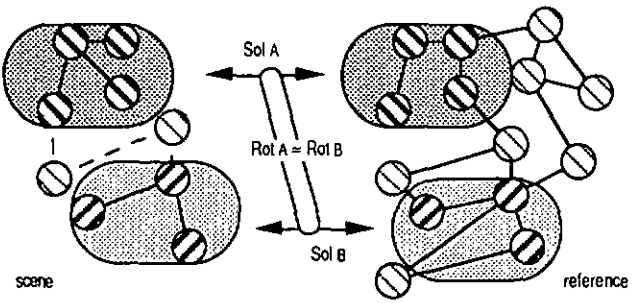


fig 9.15: compatible solutions A and B (matched nodes dotted othe

multiple object scene, single reference object

In the case of a scene with multiple objects, the matching with a single reference object will generally be done by using the ideas from above. Clearly, multiple, valuable solutions could occur, using each, partially or completely, the same reference model elements for the establishment of the correspondences.

9.4.2 Extraction of compatible solutions

Having merged partial solutions on the single reference object level, we can merge now the lists of solutions L_k issued from different reference models resulting in a "lowest-cost-first" ordered merged list of solutions ML.

Again, we extract compatible sets of solutions which can be interpreted as multiple partial matches with one or several reference objects.

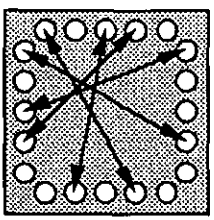


fig 9.16: graph of incompatibility between partial solutions

The result can be represented in a graph of incompatibility (fig 9.16) which represents mutual exclusions of solutions The algorithm for the test of incompatibility can roughly be described as follows :

1. take the "best" solution
   A certain number of scene faces will be matched and, since the matches are assumed to be correct, are no more available for further matches.

2. take a second solution
   If one of the correspondences leads to a scene element already used before, interpret solutions 1 and 2 as incompatible
   else
      the solutions are compatible
      if one of the correspondences leads to a reference element already used before, conclude on the presence of a second object of the same shape
      else
         interpret it as a match with a second reference object.

By this procedure, sets of compatible solutions can be extracted even for multiple object scenes.

Ordering of sets of compatible solutions

The best set of compatible solutions is the one containing the best, "lowest cost" solution. If the number of such sets is bigger than 1, the next best solution in each of the sets will be used in the ordering criterion and so on.

9.4.3 Example "CUPYRO"

We matched the PFRG of scene "CUPYRO" with the reference PFRGs "CUBE1_REF" and "PYRAM2_REF".

The approach described before and using the graph of incompatibility, delivers the desired result, represented in figure 9.17 The "best" set of compatible solutions contains the correct match between the reference object "CUBE1_REF" and the part of the scene "CUPYRO" representing the cube, as well as the reference object "PYRAM2_REF" and the part of the scene representing the pyramid.

```
   1                  ; solution_number
cupyr_c.6.14.1
   289                ; state_nb
   3          ; nb of nodes matched
   3          ; nb of arcs matched
   0        ; nb of sce arcs inserted
   0        ; nb of sce arcs deleted
 1.01357E+00          ; node diss
 8.16662E-02        ; arc attr diss
 0.00000E+00        ; arc stru diss
 8.65213E-06          ; rot diss
  0.812  0.583 -0.022    ; R matrix
 -0.289  0.435  0.853
  0.507 -0.686  0.522
 1.96176E-02            ; cost
 31   3 sce / ref       ;matches
 18   2 sce / ref
 46   1 sce / ref
.

 3 faces matched of  3 visible
```

```
   1                  ; solution_number
cupyr_p.68.3.1.1.1
   312                ; state_nb
   5          ; nb of nodes matched
   7          ; nb of arcs matched
   0        ; nb of sce arcs inserted
   0        ; nb of sce arcs deleted
 1.46342E+00          ; node diss
 1.34588E+00        ; arc attr diss
 0.00000E+00        ; arc stru diss
 7.02529E-03          ; rot diss
  0.652  0.758 -0.017    ; R matrix
 -0.635  0.558  0.534
  0.414 -0.337  0.845
 3.51257E-02            ; cost
 53   5 sce / ref       ;matches
 57   4 sce / ref
 56   1 sce / ref
 59   2 sce / ref
 54   6 sce / ref
.

 5 faces matched of  6 visible
```

fig 9.17: extracted set of compatible solutions, "CUPYRO" vs. "CUBE1_REF" and "PYRAM2_REF"

9.5. Conclusions

In the present chapter, we have shown how to handle the problem of object recognition, treating first the cases single/multiple scene objects and single/multiple reference object separately.

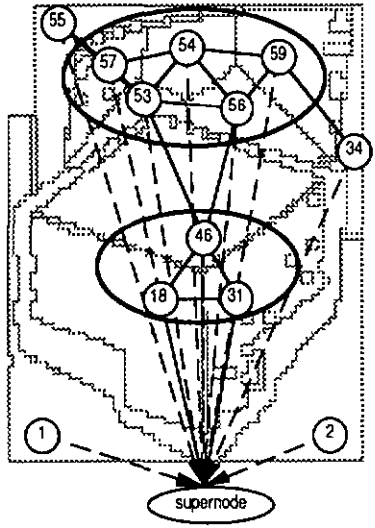


fig 9.18: prediction of the frontier between matched scene objects

The result shows that for non-occluded scene objects, the correspondences between scene objects and references have been established correctly. Combining the results of the matches with all reference models while checking for compatibility allows to recognize multiple objects in a scene, allowing also to separate the scene objects (fig 9.18).

With our choice of high level representation and correspondence search criterions, the reliability of the correspondences decreases rapidly in the case of partially occluded objects. The results must be enhanced, either by further (low level) verification or by the use of richer data in both high level representation and correspondence search criterions.

10. Conclusion and outview

10. Conclusion and outview

Conclusion

In the present work, a 3D object recognition system has been presented which acquires raw 3D data, extracts a dense high level representation and matches it with models contained in a reference library. The results are partially verified hypotheses on the identity of the objects in the scene and their respective orientation.

The implementation of the system has been discussed module by module, according to the frame shown in figure 10.1. In its present form, it is restricted to objects delimited by planar surfaces, this according to the problem definition of § 1.3.

Most of the non-overlapping parts have been recognized correctly. Attempts to use the system for object separation and recognition in multiple scene environments have been made with success.

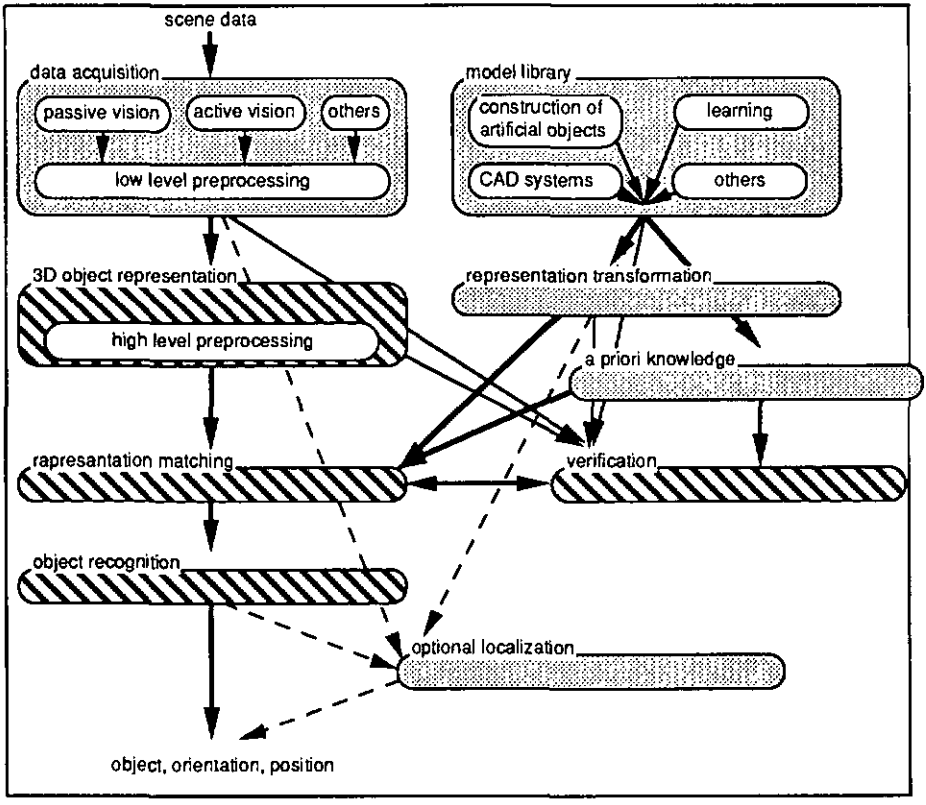


fig 10.1: a frame for an object recognition system

The most important contributions of the author are in the following two domains:

i) "inexact matching", development of the method including heuristics, and
ii) implementation of a complete 3D object recognition system including data acquisition, data preprocessing, high level representation matching and verification.

Outview

Despite the good results the researeh cannot be considered as terminated and a lot of features must be subject to further investigations. According to the frame of fig 1.10, we will pass through the different steps and discuss points which should be improved or newly considered in a future work.

data acquisition

A first aim is to create less ambiguous input data. We have seen before that the quality of the data has a big importance on the results, but also on the choice of attributes available for search tree pruning and distinction of hypotheses on solutions in our matching problem. The following enhancements should be considered:

i) extension to higher order surfsces

An extension of the surface smoothing method from planes to curves of higher order allows to use the order and type of surface as attributes in the subsequent representation and, therefore, also for the matching [Bes88], [Tie90].

ii) reconstruction of borders from smoothed surfsces

The possibility to describe surfaces by curves of higher degree allows also the reconstruction of region borders and hence a more precise determination of region attributes (e.g. surface area). Additionally, real borders can be distinguished from borders which are due to occlusions, an important indication when determining the reliability of the attributes extracted (e.g. border length, surface area, ...) [Osh87].

high level representation

The matching results obtained with the dense high level representation PFRG are rather good. Nevertheless, for complex scenes and less reliable input data, it would be interesting to extract some additional attributes from the lower level representation, following the ideas mentioned. A measure of reliability, mentioned before, could be integrated, too, being useful in the later matching steps.

dissimilarity measure

The high level representation matching leads generally to a set of (verified) hypotheses and not to a unique solution. Therefore, the measure of dissimilarity has, beside the searchtree pruning criterions, a second important application, the one of the selection of the best matching hypothesis.

Therefore, the rules for the computation of dissimilarities, but also their combination to a global dissimilarity value, needs some more investigations. Of particular interest are i) the dissimilarity measures themselves, ii) the combination rules, iii) the weightings and iv) criterions which allow to judge "locally" if a measure is useable or not (confidence).

matching

With the results shown, we estimate that the basic inexact matching method does not need to be changed. The related aspects "compatibility of partial solutions" and "combination of partial solutions" are of much more interest since they allow to recover from more important acquisition and matching non-idealities. Some propositions for future work have been made in paragraph 8.1.

verification

We limited the verification to data of the highest level of representation. Using the PFRG representation, further work could be done on a medium (e.g. smoothed surfaces) and low level (e.g. depth map) of representation. This allows, beside the verification of the hypotheses, to realize the localization task.

object separation

In chapter 9, "recognition", we have seen that multiple scene objects can be separated and recognized despite a partial occlusion. An only partially resolved problem is the comparison of the sets of hypotheses we get from the matching step.
For the most forward object, we will have good results, where for partially occluded objects the dissimilarities determined will rapidly grow. The main problem is to distinguish between incremented dissimilarities due to bad matching and incremented dissimilarities due to partial occlusion. This underlines the importance of a measuring of reliability on the attributes, proposed before in section "data acquisition" and "high level representation".

reduction of the computational effort

Despite the big advancements made in computer technology, computation time is still crucial, specially in search algorithms. The following points would merit further investigations: i) the algorithm for the search of inexact, structural matches, ii) the tree pruning criterions (reduction of the number of states to check), iii) the internal data structure and iv) the computation rules applied (e.g. determination of global dissimilarity, hypotheses on rotations, ...).

final conclusion

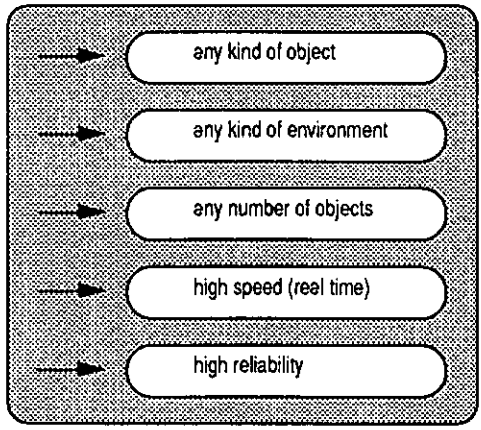


fig 10.2: the final goal in 3D object recognition

As a final conclusion, we can state that the domain of 3D object recognition and inexact matching is very complex, offering no general solution capable of handling very different cases. A big amount of work is still necessary in order to achieve the final goal of 3D object recognition (fig 10.2).

11. Acknowledgments

11. Acknowledgments

Many people have helped me to realize this PhD thesis and provided advice and encouragement. I would like to express my gratitude to the following persons:

- Heinz Hügli, my supervisor, for his generous support and guidance throughout the years I worked within his group. His advice, clear vision and patience were invaluable.

- Fausto Pellandini and Thierry Pun for their efforts and contributions in the author's doctoral committee.

- Gilbert Maître for his direct assistance in this research project, essentially in the domain of data acquisition, and all the time he has invested in long discussions on many aspects of this work.

- François Corthay and Philippe Thévenaz for serving as valuable sources for comments and help.

- All the collaborators for the good ambience at the Institute of Microtechnology of the University of Neuchâtel. A special thank to all of them, and especially to Daniel Glauser, who have invested plenty of time to make the complex infrastructure of the institute running.

- Elsbeth Sommer for the correction of this dissertation.

- My parents who offered me the possibility to choose this way.

I am also very grateful to all the people who offered me friendship and distraction, making me remember that there are more important points in life than writing a PhD thesis.

This work has been supported by the Swiss National Foundation for Scientific Research under project numbers FN2265, FN2000-5.138 and FN 20-25572.

12. References

12. References

[Ama86a] H. P. Amann
"Reconnaissance d'objects 3D: principes de vision et développement d'un système de vision stéréoscopique"
IMT Uni Neuchâtel, report 194 EC 08/86 (in french)

[Ama86b] H. P. Amann
"Reconnaissance d'objets 3-D: acquisition d'images et mise en correspondance: algorithmes"
IMT Uni Neuchâtel, report 204 EC 11/86 (in french)

[Ama87a] H. P. Amann
"Reconnaissance d'objets 3D: recherche de facettes dans un ensemble de points 3D acquis"
IMT Uni Neuchâtel, internal report 213 IC 06/87 (in french)

[Ama87b] H. P. Amann
"Papers in the domain of 3D Object Representation and Recognition"
IMT Uni Neuchâtel, internal report 229 IC 12/87

[Ama88a] H. P. Amann
"Papers in the domain of 3D Object Recognition, Recognition Planning and 3D Object Representation"
IMT Uni Neuchâtel, internal report 236 IC 04/88

[Ama88b] H. P. Amann, H. Hugli
"Stereo by Dynamic Programming boosted using Binary Images"
Proc. EURASIP 88, Grenoble, 1988

[Ama88c] H. P. Amann
"Graph Isomorphism: An Overview"
IMT Uni Neuchâtel, internal report 248 IC 11/88

[Ama88d] H. P. Amann
"ISOMOR: A Program for Subgraph- and Graph-Isomorphism"
IMT Uni Neuchâtel, internal report 251 IC 12/88

[Ama89] H. P. Amann
"Matching the representations of 3D Objects: An Overview"
IMT Uni Neuchâtel, internal report 255 IC 2/89

[Arn80] R. D. Arnold, T. O. Binford
"Geometric constraints in Stereo Vision"
SPIE Vol 238, Image Processing for Missile Guidance, 1980

[Aru87] K. S. Arun, T. S. Huang, S. D. Blostein
"Least-Squares Fitting of Two 3-D Point Sets"
IEEE PAMI-9, No 5, Sept 1987

[Aya87a] N. Ayache
"Construction, appariement et fusion de cartes visuelles bruitées"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Aya87b] N. Ayache, F. Lustman
"Un algorithme rapide et fiable de stéréovision passive trinoculaire"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Bai84] H. S. Baird
"Model-Based Image Matching Using Location"
MIT Press, Cambridge Mass. , USA, 1984

[Bak81] H. Baker, Th. Binford
"Depth Edge and Intensity Based Stereo"
Proc 7th Joint Conference on AI, Aug 1981

[Bal82] D. H. Ballard, C. H. Brown
"Computer Vision"
Prentice-Hall, Englewood Cliffs, NJ, USA, 1982

[Bau72] B. G. Baumgart
"Winged edge polyhedron representation"
STAN-CS-320, AIM-179, Stanford AI Lab, 1972

[Ber73] A. T. Bertziss
"A Backtrack Procedure for Isomorphism of Directed Graph"
ACM, Vol 20, No 3, Jul 73, pp 365-377

[Bes85] P. J. Besl, R. C. Jain
"Three Dimensional Object Recognition"
ACM Computing Surveys,17, No 1, 1985

[Bes86] P. J. Besl, R. C. Jain
"Invariant Surface Characteristics for 3D Object Recognition in Range Images"
Computer Vision and Image Processing, Jan 1986

[Bes88] P. J. Besl, R. C. Jain
"Segmentation through variable-order surface fitting"
PAMI-10, No 2, March 88

[Bey87] H. A. Beyer
"Some Aspects of the Geometric Calibration of CCD-Cameras"
ISPRS Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, June 1987

[Bey89a] H. A. Beyer
"Real Time Photogrammetry in High-Speed Robotics"
in Proc. on "Optical 3-D Meaasurement Techniques", Vienna, Austria, Sept 1989

[Bey89b] H. A. Beyer
"Calibration of CCD-Cameras for Machine Vision and Robotics"
SPIE Vol. 1197, Automated Inspection and High Speed Architectures III, 1989

[Bha82] B. Bhanu
"Surface Representation and Shape Matching of 3-D Objects"
Proc IEEE Pattern Recognition and Image Processing, 1982

[Bha84] B. Bhanu
"Representation and Shape Matching of 3-D Objects"
IEEE PAMI-6, Vol 3, May 1984

[Bha87a] B. Bhanu, C. C. Ho, T. Henderson
"3D Model Building for Computer Vision"
Pattern Recognition Letters 5, 1987

[Bha87b] B. Bhanu, C. C. Ho
"CAD-Based Object Representation for Robot Vision"
Computer, Aug 1987

[Bit75] J. R. Bittner, E. M. Reingold
"Backtrack Programming Techniques"
ACM, Vol 18, No 11, Nov 75, pp 651-656

[Bol87] R. C. Bolles, P. Horaud
"3DPO: A Three-Dimensional Part Orientation System"
in "Three-dimensional Machine Vision", ed T. Kanade, Kluwer Academic Press, 1987, pp399

[Boy87] K. L. Boyer, A. C. Kak
"Modeling and calibrating of a structured light scanner for 3-D robot vision"
Proc IEEE Int Conf on Robotics and Automation, 1987

[Bow88] K. Bowyer, J. Stewman, L. Stark, D. Eggert
"ERRORS-2: A 3D Object Recognition System Using Aspect Graphs"
9th Int. Conf on Pattern Recognition, pp 6, Nov 1988,

[Bra87] E. Le Bras-Mehlman
"Représentation de données stéreo par la triangulation de Delaunay"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Bra88] J. P. Brady, N. Nandahakumar, J. K. Aggarwal, K. Bowyer, J. Stewman, L. Stark, D. Eggert
"Recent Progress in the Recognition of Objects from Range Data"
Proc. 9th Int. Conf on Pattern Recognition, pp 85, Nov 1988

[Bron73] C. Bron, J. Kerbosch
"Algorithm 457: finding all cliques in an undirected graph"
ACM, Vol 16, No 9, Sept 73, pp 575-577

[Bro80] I. N. Bronstein, K. A. Semendjajew
"Taschenbuch der Mathematik"
Verlag Harri Deutsch, Thun, 1980, p 268

[Bro84] Ph. Brou
"Using the Gaussian Image to Find the Orientation of Objects",
The Int. Journal of Robotics Research, Vol 3, No 4 Winter 1984

[Cas85] S. Castan, J. Shen
"A New Fast Algorithm of Stereo Vision"
SPIE 595, Computer Vision for Robots, 1985

[Che81] J. K. Cheng, T. S. Huang
"Image Recognition by Matching relational Structures"
Proc IEEE Pattern Recognition and Image Processing, 1981

[Che82] J. K. Cheng, T. S. Huang
"Recognition of Curvilinear Objects by Matching Relational Structures"
Proc IEEE Pattern Recognition and Image Processing, 1982

[Chi86] R. T. Chin, C. R. Dyer
"Model Based Recognition in Robot Vision"
Computing Surveys, Vol 18, No1, March 1986

[Cor70] D. G. Corneil, C. C. Gotlieb
"An efficient Algorithm for Graph Isomorphism"
ACM, Vol 17, No1, Jan 1970, pp 51-64

[Coh80] E. Cohen et al.
"Discrete B-Splines and Subdivision Techniques in Computer Aided Geometric Design and Computer Graphics"
Computer Graphics and Image Processing, Oct 1980

[Der90] R. Deriche
"Fast algorithms for low-level vision"
PAMI Vol 12, No 1, Jan 1990, p78-87

[Dye84] C. R. Dyer
"A quad tree translation algorithm"
Computer Science Technical Report, Univ. Wisconsin, 1984

[Esh84] M. A. Eshera, K. S. Fu
"A similarity measure between Attributed Relational Graphs for Image Analysis"
7th Conf Pattern Recognition, Montreal 1984

[Esh86] M. A. Eshera, K. S. Fu
"An Image Understanding System Using Attributed Symbolic Representation and Inexact Graph Matching"
IEEE PAMI-8, Vol5, Sept 1986

[Even79] S. Even
"Graph Algorithms"
Computer Science Press, 1979

[Fau84] O. D. Faugeras
"New Steps toward a flexible 3-D Vision System for Robotics"
7th Conf Pattern Recognition, Montreal 1984, p 796-805

[Fau87a] O. Faugeras et al.
"Calcul du mouvement et de la structure à partir de points et de droites"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Fau87b] O. D. Faugeras, M. Hebert
"The Representation, Recognition and Positioning of 3D-Shapes from Range Data"
in "Three-Dimensional Machine Vision", ed T. Kanade, Kluwer Academic Press, 1987, pp301

[Fen88] J. Feng, J. F. Boyce
"Object Recognition using Relational Clique and Cycle Mappings"
Proc. 9th Int. Conf on Pattern Recognition, pp 313, Nov 1988

[Gag87] A. Gagalowicz
"Un nouvel algorithme de mise en correspondance de régions dans des paires d'images stéréo"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Gmu89] E. Gmür
"Ein Roboter-Sichtsystem basierend auf CAD-Modellen"
thesis, University of Berne, Switzerland, 1989 (in german)

[Gri84] W. E. L. Grimson, T. Lozano-Perez
"Model Based Recognition and Localization from sparse range or tactile data"
Int Joint Robotics Research, Fall 84

[Gri86] W. E. L. Grimson
"From images to surfaces"
MIT Press, Massachusetts, USA, 1981, 1986

[Gri87] W. E. L. Grimson, T. Lozano-Perez
"Recognition and Localization of Overlapping Parts From Sparse Data"
in "Three-dimensional Machine Vision", ed T. Kanade, Kluwer Academic Press, 1987, pp 451

[Gun87] K. T. Gunnarson, F. B. Prinz
"CAD Model-Based Localization of Parts in Manufacturing"
Computer, Aug 87

[Ham87] L. N. Hambrick, M. H. Loew, R. L. Carroll
"The Entry-Exit Method of Shadow Boundary Segmentation"
IEEE PAMI-9, No 5, Sept 1987

[Har79] R. Haralick, G. L. Elliott
"Increasing tree-search efficiency for constraint satisfactory problems"
Proc 6th IJAI, Aug 79, pp 356-364

[Her87] J. Hervé
"Détection de lignes de crêtes et de vallées dans les images multiniveaux"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Hen85] T. Henderson, S. Shanu
"Intrinsic characteristics as the interface between CAD and machine vision systems"
Pattern Recognition Letters 3, 1985

[Hon87] T. Hong, M. Shneier
"Rotation and Translation of objects represented by octrees"
Proc IEEE Int Conf Robotics and Automation, April 1987

[Hor74] S. L. Horowitz, T. Pavlidis
"Picture segmentation by directed split-and-merge procedure"
Proc 2nd IJCPR, Aug 74

[Hor84a] B. K. Horn
"Extended Gaussian Images"
Proceedings of the IEEE, Vol 72, No 12, Dec 1984

[Hor84b] B. K. Horn, K. Ikeuchi
"Mechanically manipulating randomly oriented parts"
Scientific American, vol 251, no2, p100-113, Aug 1984

[Hor87] R. Horaud, T. Skordas
"Model-Based Strategy Planning for Recognizing Partially Occluded Parts"
Computer, Aug 87

[Hou62] P. V. C. Hough
"Method and Mean for Recognizing Complex Patterns"
US Patent 3069654, 1962

[Hug86] H. Hugli
"Tendances et problèmes en visionique"
IMT Uni Neuchâtel, report 195 EC 08/86 (in french)

[Hug88a] H. Hugli, G. Maitre
"Generation and Use of Color Pseudo Random Sequences for Coding Structured Light in Active Ranging"
Proc. SPIE, Vol 1010, 1988

[Hug88b] H. Hugli, G. Maitre
"Generation and Use of Color Pseudo Random Sequences for Coding Structured Light in Active Ranging"
IMT Uni Neuchâtel, report 244 EC 09/88

[Hug89] H. Hugli
"Color Coding structured light for computer vision: existence of constrained N-unique sequences of maximum length"
IMT Uni Neuchâtel, report 257 EC 4/89

[Ike83] K. Ikeuchi
"Determing Attitude of Object From Needle Map Using Extended Gaussian Image"
A. I. Memo No 714, April 1983, MIT

[Ito86] M. Ito, A. Ishii
"Three-View Stereo Analysis"
IEEE PAMI-8, Vol 4, July 1986

[Jac80] C. L. Jackins, S. L. Tanimoto
"Ocitrees and Their Use in Representing Three-dimensional objects"
Computer Graphics and Image Processing,14, 1980

[Kan89] I. L. Kantor
"Hypercomplex numbers: An elementary Introduction to Algebras"
Springer, New York, 1989

[Kno68] W. Knödel
"Bestimmung aller maximalen, vollständigen Teilgraphen eines Graphen nach Stoffers"
Computing, Vol 3, No 3, 1968, pp 239-240 (in german)

[Kur85] S. Kuroe et al.
"Segmented-region based stereo reconstruction"
SPIE Vol 595, Computer Vision for Robots, 1985

[Lin87] C. C. Lin, R. Chelappa
"Classification of partial 2-D Shapes Using Fourier Descriptors"
IEEE PAMI-9, No 5, Sept 1987

[Lit83] J. J. Little
"An iterative method for reconstructing convex polyhedra from extended gaussian image"
Proc Nat Conf on AI, Washington, USA, 1983

[Lu88] S. W. Lu, A. K. C. Wong
"Synthesis of attributed hypergraphs for knowledge representation of 3D-Objects"
Int. Conf. on Pattern Recognition, Cambridge, March 1988

[Lum85] R. Lumia
"Representing Solids for a real-time Robot sensory system"
Proc. PROLAMAT, Paris, June 85

[Lum86] R. Lumia
"Rapid Hidden Feature Elimination Using an Octtree"
IEEE, Int Conf Robotics and Automation,1986

[Mai87] H. Maître, L. Pocztar
"Restitution d'un flot de particules par mise en correspondance spatio-temporelle d'images"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Mai88] G. Maître, H. Hugli
"Réalisation d'un dispositif de mesure 3D utilisant de la lumière structurée"
IMT Uni Neuchâtel, report 250 EC 12/88 (in french)

[Mai89] G. Maître, H. Hugli
"Extraction de surfaces par croissance de régions sur un modèle d'objet $2^1/2D$ "
report 259 EC 05/89 (in french), IMT University of Neuchâtel, Switzerland, 1989 (in french)

[Mai90] G. Maître, H. Hugli
"Segmentation des images de profondeur par diffusion"
report 276 EC 03/90 (in french), IMT University of Neuchâtel, Switzerland, 1990 (in french)

[Mil80] D. L. Milgram, C. M. Bjorklund
"Range Image Processing: Planar Surface Extraction"
Proc 5th Conf Pattern Recognition, Miami Beach, Dec1980

[Min97] H. Minkowski
"Allgemeine Lehrsätze über die konvexen Polyeder"
Nachrichten von der Königlichen Gesellschaft der Wissenschften, Göttingen, Germany, 1897 (in german)

[Mon87] O. Monga, B. Wrobel-Dautcourt
"Une méthode simple de contrôle de croissance de régions par les contours"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Nev80] R. Nevatia, K. R. Babu
"Linear feature extraction and description"
Computer Graphics and Image Processing, Vol 13, 1980

[Oht85] Y. Ohta, T. Kanade
"Stereo by Intra- and Interscanline Search using Dynamic Programmation"
IEEE PAMI-7, No 2, March 1985

[Osh83] M. Oshima, Y. Shirai
"Object Recognition Using Three-Dimensional Information"
IEEE PAMI, Vol 5, No 4 July 1983

[Osh87] M. Oshima, Y. Shirai
"An Object Recognition System Using Three-Dimensional Information"
in "Three-Dimensional Machine Vision", ed T. Kanade, Kluwer Academic Press, 1987, pp 355

[Phi86] T. Phillips, R. Cannon, A. Rosenfeld
"Decomposition and Approximation of Three-Dimensional Solids"
Computer Vision, Graphics, Image Processing 33, Academic Press,1986

[Ric87] M. Richetin and al.
"Attitude spatiale des courbes vues en projection"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Ros76] A. Rosenfeld, A.C. Kak
"Digital Picture Processing"
Academic Press, New York, 1976

[Rut87] W. Rutkowski and al
"Model Driven Determination of Object Pose for a visually servoed robot"
Proc IEEE Int Conf Robotics and Automation, April 1987 (in french)

[Sad80] F. Sadjadi, E. l. Hall
"Three-Dimensional Moment Invariants"
IEEE PAMI, Vol 2, No 2 March 1980

[Sak78] H. Sakoe, S. Chiba
"Dynamic Programming Algorithm Optimization for Spoken Word Recognition"
IEEE Trans on ASSP, Vol ASSP-26, No 1, February 1978

[San83] A. Sanfelieu, K. S. Fu
"A distance measure between attributed relational graphs for pattern recognition"
IEEE Trans Syst. , Man, Cybern. , May-June1983

[Sha85] L. Shapiro, R. M. Haralick
"A Metric for Comparing Relational Descriptions"
IEEE PAMI-7, No 1, January 1985

[Shn87] M. O. Shneier, R. Lumia, M. Herman
"Prediction-Based Vision for Robot Control"
Computer, Aug 87

[Sil84] T. Silberberg, L. Davis, D. Harwood
"An Iterative Hough Procedure For Three-Dimensional Object Recognition"
Pattern Recognition, Vol 17, No 6, 1984

[Sko87] Th. Skordas
"Groupement robuste de caractéristiques image"
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)

[Ste87] C. N. Stevenson
"Model-Based Programming and Control of Robot Manipulators"
Computer, Aug 87

[Str86] T. M. Strat, M. A. Fischler
"One-Eyed Stereo: A General Approach to Modeling 3-D Scene Geometry"
IEEE PAMI-8, No 6, Nov 1986

[Tie90] F. Tièche
"Segmentation et modélisation d'objets 3D selon la courbure de la surface"
diploma work, University of Neuchâtel", Switzerland, 1990 (in french)

[Tur85] J. L. Turney, T. N. Mudge, R. A. Volz
"Recognizing Partially occluded Parts"
IEEE PAMI-7, No 4, July 1985

[Ull76] J. R. Ullmann
"An Algorithm for Subgraph Isomorphism"
ACM, Vol 23, No 1, Jan 76, pp 31-42

[Wal81] T. Wallace, O. R. Mitchell, K. Fukunaga
"Three-Dimensional Shape Analysis Using Local Shape Features"
IEEE PAMI, Vol 3, No 3 May 1981

[Wen79] M. J. Wenninger
"Polyhedron Models"
Cambridge University Press, Cambrige UK, 1979

[Won85] A. K. C. Wong, M. You
"Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition"
IEEE PAMI-7, No 5, Sept 1985

[Wro87] B. Wrobel-Dautcourt
"Surfaces tridimensionelles obtenues par appariement stéréoscopique de régions "
AFCET-INRIA, Congrès RFIA, Antibes, Nov 87 (in french)