Flexible Object Recognition
Based on
Invariant Theory And Agent Technology

Thorsten Graf

December 2000

Flexible Object Recognition Based on Invariant Theory And Agent Technology

Dissertation

submitted in partial satisfaction of the
requirements of the degree of Dr.-Ing.

Thorsten Graf

Faculty of Technology
University of Bielefeld

December 2000

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706

Acknowledgments

This thesis is a result of my work as a member of the postgraduate research unit “Aufgabenorientierte Kommunikation” (task-oriented communication) at the University of Bielefeld which was funded by the Deutsche Forschungsgemeinschaft (DFG). First of all, I would like to thank my adviser Prof. Alois Knoll, whose suggestions were very helpful over the last years. I am also grateful to the other members of my thesis committee, Wolfram Burgard, Ipke Wachsmuth, and Stefan Kurtz.

Furthermore, I am also grateful to the members of the research group *Technical Informatics* at the University of Bielefeld. Especially, I would like to thank Christian Scheering, who has developed and implemented the agent library that has been extended in this thesis. Additionally, I would like to thank Yorck von Collani, Markus Ferch, and Torsten Scherer for answering my various questions. Special thanks go to André Wolfram, a former member of the research group *Technical Informatics*, for his friendship and for the fruitful collaboration which results in one of developed recognition methods.

Furthermore, I am grateful to Karsten Loer and Angelika Deister, who have carefully read this thesis and have helped to improve the English as well as to make it much more comprehensible.

Finally, I would like to thank my wife Silke as well as my children Jeannine and Marlon. Without their loving support this thesis would not have been realized.

Thorsten Graf
December 2000

Contents

List of Figures

List of Tables

1

Introduction

This chapter describes the subject of the thesis. It motivates the work and mentions several aspects that will be discussed within the thesis. Furthermore it provides a summary of the content.

1.1 Motivation

The capability of observing the world based on visual information is an essential requirement in robotic applications with increasing importance since tasks handled within the robotic scenarios are getting more complex and fewer restrictions are imposed to the environmental conditions. Especially systems acting in a natural dynamic environment cannot be provided with a complete description of the world. Due to unexpected incidents, like occurring obstacles or execution errors, these systems have to rely on the observed information.

Therefore, modern complex robotic applications impose several requirements to a vision system concerning both: reliability and flexibility. Due to external influences, e.g. partial occlusions of objects and illumination changes, as well as to internal influences, e.g. noisy imaging hardware, inaccurate measurements and quantisation effects, a vision system has to cope with incomplete, uncertain and inaccurate information. Furthermore, the system must often be able to handle a variety of vision tasks originated in different research areas, like object recognition, scene reconstruction and object tracking. Since the observed information may come from different visual sensors, different (competitive) information must be handled.

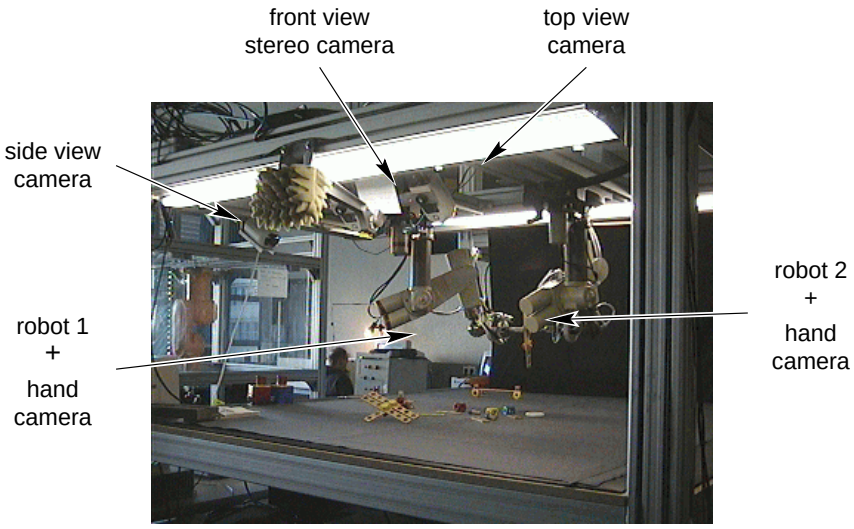


Figure 1.1: Example for a complex robotic setup

An example for such a robotic setup is shown in Fig. 1.1, which is part of the research project SFB 360 “Situated Artificial Communicators” [RICKHEIT and WACHSMUTH 1996] supported by the German Research Foundation (DFG). The goal is to develop an intelligent assembly cell which is controlled and assisted by a human instructor using natural spoken language and gestures. The particular assembly task of the system is to build up complex toy objects, such as airplanes, using the Baufix construction kit (see Appendix B).

As shown in Fig. 1.1, the assembly cell is composed of two robot arms in order to perform the construction as well as several cameras to solve the various sub-tasks that may arise during the assembly process: The overhead cameras are mainly used for recognising and locating the objects visible in the working space to validate given instructions and to provide a coarse-positioning of the robot arms. These tasks can be supported by a side-view camera which provides redundancy, that can be used to make the recognition results more robust. Furthermore, the assembly cell contains a front-view stereo camera rig to gain a 3D description of the scene to guide the assembly process as an additional source. Finally, the robot arms are equipped with hand cameras, which are mainly used for visual guidance, i.e. the fine-positioning process to find an optimal grasping position. The hand cameras, of course, might also be used for active vision to inspect the scene in more details.

Although considerable effort in the research area of computer vision has been spent on image processing, the potential of utilising visual information has only been scratched. Generally, the developed vision algorithms tend to be special purpose algorithms requiring environmental conditions that cannot or should not be fulfilled in complex robotic scenarios. Furthermore, vision systems generally follow system architectures which make it difficult to comply with the flexibility requirements of complex robotic setups.

Therefore, the development of a vision system suitable for such robotic applications should comprise both, the development of flexible and robust vision algorithms as well as of a flexible system architecture.

1.2 Contribution

Since this thesis cannot investigate all of the different vision tasks and aforementioned aspects that must be handled within a complex robotic scenario, it concentrates on the development of a flexible object recognition system. The underlying scenario is the assembly cell shown in Fig. 1.1. However, it must be noted, that many of the basic ideas and proposed approaches are not restricted to object recognition systems solely, but can also be applied to various other vision tasks. Especially, the developed system architecture provides systems that can be easily expanded by adding appropriate processing modules to provide new functionality, like active-vision components or 3D scene reconstruction modules. This flexibility is achieved by a combination of up-to-date object recognition methods and agent technology in a beneficial way.

The recognition methods developed in the thesis are mainly based on invariant theory. This theory provides mechanisms to generate object descriptions that remain unaffected by intrinsic and extrinsic camera parameters, i.e. invariant descriptions are object properties which do not change under imaging.

Two different recognition methods based on invariants are investigated: a geometric-based and an appearance-based recognition method. The former utilises geometric structures to build up the invariant descriptions. The main advantage of these invariants is that geometric information is observable very accurately, even under changing illumination conditions. Unfortunately, they cannot be applied to all different types of objects because the object structure must meet some requirements.

The latter recognition method utilises appearance-based invariants. These invariants are measured for segmented image patches and take the whole image patch information into account. Since both recognition methods rely on different image information and work in completely different ways they also differ in some of their properties; e.g. contrary to the former one the appearance-based method can be applied to a greater variety of objects, but is more affected by changing illumination conditions.

Nevertheless, these methods share the following features, which appear to be suitable for complex robotic scenarios:

- Cumbersome and error-prone camera calibration processes are avoided.
- Once an object has been acquired, it can be recognised from various viewpoints using different cameras (also non-static cameras, like the hand cameras of the assembly cell).
- Objects can be recognised even though they are partially occluded.

The flexibility of the recognition methods is further improved by the developed system architecture. The basic idea of this architecture is to model a vision system as a society of autonomous agents, where each agent is responsible for specific vision tasks, the control strategy of a vision system is decentralised, and agents communicate using a flexible but easy understandable communication language. This directly leads to self-organising vision systems, which accomplish vision tasks by goal-driven communication processes.

As a result of this architecture the developed object recognition system gain the following features:

- The recognition system can be easily integrated into other complex applications (like the assembly cell).
- The system is robust to breakdowns of single processing modules.
- It has the ability to adapt dynamically to different tasks and environmental conditions.
- Different and competitive information can be handled.
- The system is expandable by adding new agents which provide new functionalities.
- The system can perform resource management automatically.

1.3 Outline

The structure of the thesis can be decomposed into two main parts in order to reflect that two different research areas are combined in a beneficial way to support each other: The first part, Chap. 2 – Chap. 4, investigates flexible object recognition methods based on invariants. It provides an introduction into invariant theory and develops two different recognition systems that utilise either geometric invariants as well as pattern invariants. The second part, Chap. 5 – Chap. 6, concentrates on multi-agent system architectures dedicated to build computer vision systems. It introduces some basic aspects of agent technology and proposes a new multi-agent system architecture that provides vision systems with a high degree of flexibility. Additionally, an agent-based recognition system is presented which integrates the complementary recognition methods developed in the first part of the thesis.

Figure 1.2 sketches the structure of the thesis in more detail and indicates how the different chapters are related. In particular, Chap. 2 provides an introduction to invariant theory which facilitates the task of recognising objects because the theory can be used to generate object descriptions which remain unaffected by intrinsic and extrinsic camera parameters. The chapter describes the basic concepts of employing invariant theory in object recognition and reviews recent work.

Chapter 3 concentrates on object recognition systems based on geometric invariants. It develops a hypothesis generation method based on fuzzy set theory, i.e. it employs fuzzy if-then classification rules and fuzzified invariant object descriptions. This method enhances

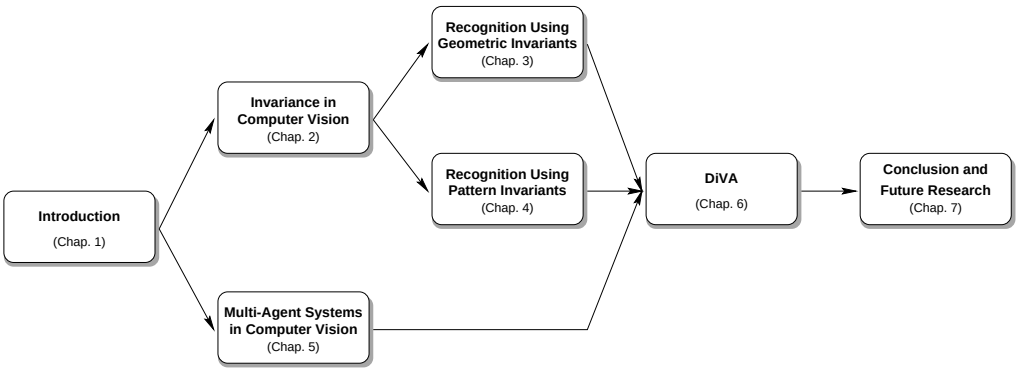


Figure 1.2: Structure of the thesis

the commonly used hypothesis generation methods based on geometric invariants and provides a great degree of flexibility. A first object recognition system based on the proposed hypothesis generation method is developed and experimental results demonstrate the performance and the flexibility of the proposed approach.

Chapter 4 develops an object recognition method that utilises pattern invariants in order to distinguish between different objects. This method is based on a classification scheme that combines the advantages of employing pattern invariant object descriptions with the classification and approximation ability provided by principle component analysis. The chapter comprehensively explains the general framework of the classification scheme and presents an object recognition system that incorporates the developed method.

Chapter 5 describes several aspects of agent technology which are necessary to build flexible computer vision systems that enhance the capabilities of conventional systems. It introduces some basic concepts of agents and multi-agent systems and discusses the main advantages of employing agent technology in computer vision. Moreover, it reviews recent work which focus on the development of multi-agent vision architectures.

Chapter 6 concentrates on the development of a new multi-agent system architecture dedicated to model computer vision systems. This architecture, called DiVA (Distributed Vision Architecture), can be utilised to enhance the flexibility and applicability of a vision system in many respects. The basic idea of the architecture is to model a vision system as a society of self-organising, autonomous agents, where each agent is responsible for specific vision tasks, the control strategy of a vision system is decentralised, and agents communicate using a flexible but easy understandable communication language. The chapter presents the general framework of the developed multi-agent system approach and presents a multi-agent recognition system employing the proposed recognition methods based on geometric (Chap. 3) and pattern invariants (Chap. 4), respectively.

Finally, Chapter 7 concludes the thesis. It summarises the main results of the thesis and provides some possible future research directions.

2

Invariance in Object Recognition: A Review

This chapter provides an introduction to the object recognition methods investigated in this thesis. These methods are mainly based on invariant theory which facilitates the recognition process of object recognition systems because it provides object descriptions which remain unaffected by intrinsic and extrinsic camera parameters. The chapter describes the basic concepts of employing invariant theory in object recognition and reviews recent work.

2.1 Motivation

Object recognition is a major research field in computer vision. Although a large amount of effort has been spent during the last decades to develop new object recognition methods, the recognition capabilities of current systems are still very limited and restricted to particular applications.

One of the fundamental difficulties in object recognition is that the appearance of objects widely varies if the objects are observed from different viewpoints. For example, the imaging process can transform the simple geometric structure of a circle to an arbitrary type of ellipse. Thus, it is generally very difficult to compare the object models stored in a model base with an unknown object observed from an unknown point of view.

To alleviate this problem, many recognition methods impose several requirements on the camera setup to reduce the degrees of freedom of the imaging process. Typically, this

is done by fixing the camera position and by assuming that the objects are located at a predefined distance to the camera. Obviously, such approaches cannot be adequately incorporated into modern complex robotic scenarios (see Sect. 1.1), since these recognition methods require a static setup and the methods must be adapted to each camera individually.

As research has indicated invariant theory provides a much more flexible and powerful approach to tackle the aforementioned vision problem. Invariant theory is an active research area and has been applied to many different vision tasks. The importance of this theory mainly results from the possibility to generate invariant object descriptions which remain unaffected under the imaging process of cameras. Such invariant descriptions can be measured directly from images without having any prior knowledge about the intrinsic parameters (e.g. focal length or aspect ratio) and extrinsic parameters (position and orientation) of a camera.

Hence, invariant object descriptions can greatly facilitate the recognition process: firstly, since the descriptions stay always the same they can be used directly as object clues by comparing the invariant descriptions observed in an image with the invariant descriptions of stored object models; secondly, cumbersome and error-prone camera calibration processes are avoided and the invariant-based recognition methods can be employed to recognise the objects in images taken from different viewpoints and cameras; and lastly, invariants reduce the manual intervention during the acquisition of new objects, because object models composed of invariant object descriptions can be principally generated from an arbitrary point of view.

Furthermore, the invariant approach provides the ability to recognise partially occluded objects. This can be done by restricting the invariant descriptions to subparts of the objects. Thus, objects can be recognised as long as single invariant descriptions can be observed in an image.

2.2 Theoretical Background

Invariant theory is a well-established mathematical theory, which provides a rich machinery to generate and analyse the characteristics of invariants. The term *invariant* refers to any quantity which remains unchanged under certain transformations. Since the thesis cannot cover all concepts of this theory it will focus on those aspects which are required to apply invariants in object recognition. For a more complete introduction to invariant theory see [WEYL 1946] or for a vision related introduction [MUNDY and ZISSERMAN 1992].

2.2.1 Transformation Groups

One of the basic concepts in invariant theory is to represent a transformation by the action of a mathematical group:

Definition 2.1: transformation group

> Let G be a group of elements acting on a vector space V, such that an element $\mathbf{g} \in G$ is a transformation $\mathbf{g} : V \to V$, together with the composition of transformations $(\mathbf{g_1 g_2})(\mathbf{x}) = \mathbf{g_1}(\mathbf{g_2}(\mathbf{x}))$, $\forall \mathbf{g_1}, \mathbf{g_2} \in G, \mathbf{x} \in V$. Then, G is called a *transformation group*.

Several transformations important to computer vision can be represented as transformation groups. For example, the following hierarchy describes a number of plane to plane transformations which are often used in vision applications based on invariants:

- *Plane projective transformation group:*
  The most general 2D linear transformation group is composed of all plane projective transformations. These transformations can be represented by homogenous non-singular matrices of the following form:

$$\mathbf{T}_{pr} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \tag{2.1}$$

  Note, that 2D projective transformations have only eight degrees of freedom (DOF). In homogeneous equations the global scaling factor t_{33} is not relevant, since the other parameters can be adjusted to yield any scaling factor (for an introduction to homogenous coordinates see [SPRINGER 1964]). Thus, t_{33} can be set to an arbitrary (non-zero) value and is often assumed to be 1.

- *Plane affine transformation group:*
  The plane affine transformation group is given by all non-singular homogeneous matrices of the following form:

$$\mathbf{T}_{af} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & t_{33} \end{bmatrix} \tag{2.2}$$

  Again, the overall scaling factor t_{33} does not matter. Thus, this transformation group has 6 DOF.

- *Plane Euclidean transformations group:*
  The 2D Euclidean transformation group represents rigid 2D motions:

$$\mathbf{T}_{eu} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

  where the upper-left (2×2)-submatrix represents a rotation and the vector $\begin{bmatrix} t_{13} & t_{23} \end{bmatrix}^t$ a translation. Since plane rotations consist of only one essential parameter, this transformation group has 3 DOF.

The importance of these transformation groups, especially of the plane projective transformation group, becomes apparent if the imaging process of cameras is analysed. Usually, this is done on the basis of the *pinhole-camera*, which is the standard model for approximating the imaging process [FAUGERAS 1993].

The following theorem expresses the relationship between plane projective transformations and the pinhole-camera model. A prove of the theorem can be found in [ROTHWELL 1995b].

Theorem 2.1:

> A pinhole perspective map between a set of homogeneous planar world points $\mathbf{x_i} = [x_i, y_i, 1]^t$ and their images $\mathbf{x_i'} = [x_i', y_i', 1]^t$ is represented by a planar projectivity.

This means, that any imaged 2D structure of an object can be approximated by transforming the original 2D structure of the object using a projective transformation.

However, in some situations the camera mappings can also be appropriately approximated by affine transformations. This can be done if the distance between the camera and the observed object is large compared with the depth of the object measured along the optical axis. Nevertheless, the imaging process is sometimes also approximated using 2D Euclidean transformations augmented by a uniform scaling factor.

2.2.2 Invariants

In invariant theory one distinguishes between *relative* and *absolute* invariants. While relative invariants may still depend on some transformation parameters, absolute invariants are completely unaffected under a particular transformation group.

Usually, in computer vision only absolute invariants are utilised in order to remove the effects of the imaging process. Thus, an invariant with respect to computer vision can be defined as follows:

Definition 2.2: invariant

> An *invariant*, $I(\mathbf{f}(\mathbf{x}))$, of a configuration described by a vector function $\mathbf{f}(\mathbf{x})$, $\mathbf{x} \in V$, subject to a transformation $\mathbf{g}$ of a transformation group G with $\mathbf{g} : V \rightarrow V$, $\mathbf{g} \in G$ acting on the coordinates $\mathbf{x}' = \mathbf{g}(\mathbf{x})$, is transformed according to $I(\mathbf{f}(\mathbf{g}(\mathbf{x}))) = I(\mathbf{f}(\mathbf{x}))$.

Note, that this definition differs from other invariant definitions used in the computer vision domain. Since geometric-based and appearance-based vision algorithms are generally investigated apart, two different types of definitions can be found: definitions for geometric-based as well as for appearance-based invariants.

Geometric-based invariants are given by the following definition [MUNDY and ZISSERMAN 1992]:

Definition 2.3: (geometric) invariant

> A (geometric) *invariant*, $I(\mathbf{x})$, of a geometric structure described by a parameter vector $\mathbf{x} \in V$, subject to a transformation $\mathbf{g}$ of a transformation group G with $\mathbf{g} : V \to V$, $\mathbf{g} \in G$ acting on the parameter vectors $\mathbf{x}' = \mathbf{g}(\mathbf{x})$, is transformed according to $I(\mathbf{g}(\mathbf{x})) = I(\mathbf{x})$.

Analogous appearance-based (or pattern) invariants are defined as [WOOD 1996]:

Definition 2.4: (pattern) invariant

> A (pattern) *invariant*, $I(p(\mathbf{x}))$, of a pattern described by a function $p(\mathbf{x})$, $\mathbf{x} \in V$, subject to a transformation $\mathbf{g}$ of a transformation group G with $\mathbf{g} : V \to V$, $\mathbf{g} \in G$ acting on the coordinates $\mathbf{x}' = \mathbf{g}(\mathbf{x})$, is transformed according to $I(p(\mathbf{g}(\mathbf{x}))) = I(p(\mathbf{x}))$.

However, the definitions of geometric and pattern invariants are very similar and can be expressed in terms of Def. 2.2.

An important aspect concerning invariants, especially geometric invariants, is the number of functional independent invariants that can be generated for a given configuration. *Functional independent* means that no function exists which transforms one invariant into another. The number of functional independent invariants can be estimated using the following counting argument [MUNDY and ZISSERMAN 1992]:

Theorem 2.2: counting argument

> Suppose a configuration is described by a vector function $\mathbf{f}(\mathbf{x}), \mathbf{x} \in V$, subjected to a transformation group G with $\mathbf{g} : V \to V$, $\mathbf{g} \in G$, which has dim G degrees of freedom. Then the number of functional independent absolute invariants is $\geq$ dim $V -$ dim G.

For example, the geometric structure of a pair of coplanar conics (10 DOF) has two functional independent invariants under plane projective transformations (8 DOF), Sect. 2.3.1.

Furthermore, invariant theory provides a rich machinery to handle various other aspects concerning invariants. For example several methods have been developed to generate new invariants, like the infinitesimal method [ARNOLD 1990], the symbolic method [ABHYANKAR 1992] or the double algebra [CARLSSON 1994]. A survey of invariants used in object recognition is given in Sect. 2.3.

2.2.3 Invariants in Object Recognition

The employment of invariants in 2D object recognition is straightforward. As suggested in Sect. 2.2.1 plane projective transformations approximate the imaging process of 2D objects. Therefore, any invariant under the plane projective transformation group evaluated for a 2D object results always in the same invariant description regardless of the particular viewpoint of the camera. This leads to efficient hypothesis generation methods because the invariant descriptions measured in an image can be directly compared with the invariant descriptions of stored object models.

For 3D object recognition the situation is somewhat more awkward. The problem is that the imaging process projects the 3D information of an object onto a 2D image plane. Such perspective transformations cannot be represented as actions of mathematical groups and hence, most of the developed methods of invariant theory cannot be applied. However, in the recent past many 3D invariants have been developed which can be evaluated for the observed 2D image information (see Sect. 2.3.1). Furthermore, 3D objects often contain (approximately) planar areas or structures, which makes the recognition of these objects by employing 2D invariants possible.

As reviewed in Sect. 2.3, various types of invariants have been developed. These invariants possess different properties and cannot be appropriately applied to every recognition task. In the following some of the main criteria for invariants used in object recognition are proposed, where a single invariant cannot comply with all of these criteria. Note, that the criteria will not be considered in order of importance, because the importance generally depend on the specific vision task (for additional information see [ROTHWELL 1995b]):

- *Completeness:*
  The completeness of invariants is a very important requirement in object recognition. It is defined as:

  Definition 2.5: complete

  Let $I(\mathbf{f}(\mathbf{x}))$ denote an invariant of a configuration described by a vector function $\mathbf{f}(\mathbf{x}), \mathbf{x} \in V$ under the transformation group G with $\mathbf{g} : V \rightarrow V, \mathbf{g} \in G$. The invariant $I(\mathbf{f}(\mathbf{x}))$ is *complete*, if:

  $$I(\mathbf{f}(\mathbf{x_1})) = I(\mathbf{f}(\mathbf{x_2})) \Rightarrow \mathbf{x_2} = \mathbf{g}(\mathbf{x_1}), \qquad \forall \mathbf{x_1}, \mathbf{x_2} \in V$$

  Obviously, different objects must result in different invariant descriptions to enable the system to discriminate between the objects. However, completeness of invariants cannot be generally achieved. This is true especially for invariants which are evaluated for subparts of an object, because many objects are locally similar or equivalent. Therefore, a recognition system based on invariants must always verify the generated object hypotheses in more detail in order to remove false positives (i.e. incorrect object hypotheses).

- *Discrimination:*
  A related requirement in object recognition concerns the discrimination ability of invariants. Due to noisy imaging hardware as well as to discretisation and quantisation effects, invariant object descriptions can never be measured exactly and are always subjected to fluctuations. Consequently, invariant descriptions belonging to different objects may be the same although the underlying invariant is complete. Again, the generated object hypotheses must be verified in order to determine the correct hypotheses.
- *Scope:*
  The scope of invariants covers the size of the object domain that can be described, the type of transformations that do not affect the invariant object descriptions, and the size of the object regions which are used to evaluate the descriptions.

  All invariants have limited scope with respect to the first two issues: invariants are designed so that specific classes of objects can be recognised assuming a particular underlying transformation group; e.g. invariants of plane geometric structures cannot be employed to recognise 2D objects composed of smooth curves nor to recognise 3D geometric structures.

  The last meaning of scope concerns the area of an object which is used for evaluation. Three different types of invariants are distinguished: local, semi-local, and global invariants. *Global invariants* are measured by taking the whole object into account. These invariants are very stable in presence of noise but cannot be measured reliably under occlusion. Contrary *local invariants* are evaluated at single distinguished points. These invariants are very sensitive to noise but can be used to recognise objects even under excessive occlusion. Finally, *semi-local invariants* are measured for a small number of proximal points and are a compromise between global and local invariants.
- *Efficiency:*
  The efficiency of invariant object descriptions is a measure for the computational cost. This measure depends not only on the effort to evaluate invariants but also must take the computational cost of required pre-processing stages into consideration. For example, in contrast to appearance-based invariants which often take the whole information of an image patch into account, geometric invariants can be computed very efficiently. However, geometric invariants require complex pre-processing stages to extract and to group geometric primitives, while appearance-based invariants require just rudimentary segmentation algorithms (see also Sect. 2.4).

2.3 Survey of Invariants

The importance of invariance to computer vision has been recognised since the origin in the field in the 1960s. However, computer vision algorithms based on invariants are getting more popular recently and are now an active research area. This section gives a survey of work done in invariant object recognition. It presents several types of invariants including both geometric as well as pattern invariants. Although this section cannot provide an exhaustive list, it indicates the diversity of invariants employed in object recognition.

2.3.1 Geometric Invariants

Geometric invariants are based on geometric structures, which can be extracted from the images of a given object. These structures are mainly generated by using the edge information provided by an edge operator, because edge information can be generally extracted very reliably even under changing illumination conditions. However, the required geometric structures may also be extracted using other preprocessing methods.

As already discussed in Sect. 2.2.3, one distinguishes between 2D and 3D invariants. Furthermore it is often suitable to distinguish between algebraic and non-algebraic invariants.

2D Algebraic Invariants

2D algebraic invariants are based on plane geometric structures of algebraic objects such as points, straight lines, and conics. These invariants can be generated for any combination of algebraic objects assuming that the resulting geometric structure is complex enough, i.e. the structure complies with the counting argument (see Theor. 2.2). Therefore, many different 2D algebraic invariants have been developed.

- *Cross-ratio:*
  The best-known (since 600 BC) and perhaps most important invariant under projective transformations is the *cross-ratio*. Although this invariant is generally not used for object recognition tasks directly, the cross-ratio is very important in projective geometry, because many invariant properties of geometric configurations can be interpreted in terms of this invariant [MUNDY and ZISSERMAN 1992].

  The cross-ratio is defined for four points lying on a straight line:

  $$I(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}) = \frac{|\,\mathbf{x_3} - \mathbf{x_1}\,|}{|\,\mathbf{x_3} - \mathbf{x_2}\,|} \frac{|\,\mathbf{x_4} - \mathbf{x_2}\,|}{|\,\mathbf{x_4} - \mathbf{x_1}\,|} \tag{2.4}$$

  where $\mathbf{x_i}, 1 \leq i \leq 4$ represent the four collinear points.

- *Invariants of a pair of coplanar conics:*
  The two functional independent projective invariants of a pair of coplanar conics are given by:

  $$I_1(\mathbf{C_1}, \mathbf{C_2}) = \frac{\text{trace}(\mathbf{C_1^{-1}C_2})\,|\mathbf{C_1}|^{\frac{1}{3}}}{|\mathbf{C_2}|^{\frac{1}{3}}} \tag{2.5}$$

  $$I_2(\mathbf{C_1}, \mathbf{C_2}) = \frac{\text{trace}(\mathbf{C_2^{-1}C_1})\,|\mathbf{C_2}|^{\frac{1}{3}}}{|\mathbf{C_1}|^{\frac{1}{3}}} \tag{2.6}$$

  where $\mathbf{C_1}, \mathbf{C_2}$ denote the conic coefficient matrices in homogeneous coordinates. Since these invariants provide a good discrimination ability between different configurations of pairs of coplanar conics, they are widely used in recognition systems based on geometric invariants.

Much work has been done concerning these invariants, addressing both object recognition applications [FORSYTH et al. 1990, QUAN et al. 1992, ROTHWELL 1995b] as well as theoretical investigations, like [MUNDY et al. 1992] which provides a geometric interpretation. Furthermore, the invariants have been extended in [QUAN and VEILLON 1998] to triplets and in [HEISTERKAMP and BHATTACHARYA 1996] to families of an arbitrary number of coplanar conics.

- *Invariants of a conic and two lines:*
  Another well-known projective invariant used in object recognition can be evaluated for the geometric structure of a single conic and two straight lines [MUNDY and ZISSERMAN 1992]:

$$I(\mathbf{C}, \mathbf{l_1}, \mathbf{l_2}) = \frac{\left(\mathbf{l}_1^t \mathbf{C}^{-1} \mathbf{l_2}\right)^2}{\left(\mathbf{l}_1^t \mathbf{C}^{-1} \mathbf{l_1}\right)\left(\mathbf{l}_2^t \mathbf{C}^{-1} \mathbf{l_2}\right)} \tag{2.7}$$

  where $\mathbf{C}$ represent the conic coefficient matrix and $\mathbf{l_1}, \mathbf{l_2}$ are the lines expressed in homogeneous coordinates. Investigations have indicated that this invariant provides only a moderate discrimination ability between different objects. However, this ability can be improved by applying the invariant to the geometric configuration of a conic and three lines $\mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}$ by determining the three functional independent projective invariants [ROTHWELL 1995b]:

$$I_1(\mathbf{C}, \mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}) = I(\mathbf{C}, \mathbf{l_1}, \mathbf{l_2}) \tag{2.8}$$
$$I_2(\mathbf{C}, \mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}) = I(\mathbf{C}, \mathbf{l_1}, \mathbf{l_3}) \tag{2.9}$$
$$I_3(\mathbf{C}, \mathbf{l_1}, \mathbf{l_2}, \mathbf{l_3}) = I(\mathbf{C}, \mathbf{l_2}, \mathbf{l_3}) \tag{2.10}$$

2D Non-Algebraic Invariants

The other type of invariants under plane transformation groups are 2D non-algebraic invariants, which are measured for non-algebraic smooth curves. In contrast to algebraic invariants which lead to single invariant values, non-algebraic invariants can often be used to generate complex *invariant signatures* as well, where invariant signatures are curves remaining unaffected under a particular transformation group.

- *Differential invariants:*
  Differential invariants are constructed by using the derivatives of a smooth curve at a single point. These invariants require higher order derivatives, e.g. affine differential invariants require derivatives of fifth order and projective invariants of seventh order. Examples for these invariants are Wilzcynski's projective invariants [WILCZYNSKI 1906] as well as the invariants described in [BRUCKSTEIN et al. 1992, WEISS 1988, WEISS 1992].

  However, using differential invariants for recognising objects is generally impractical. Due to the required high order derivatives these invariants are very sensitive to noise and cannot be measured reliably for an imaged curve [BROWN 1992].

- *Semi-differential invariants:*
  The sensitivity to noise can be reduced by employing semi-differential invariants. The basic idea of these invariants is to measure the derivatives not only at a single point but rather at various positions of the curve making a tradeoff between the number of points and the order of required derivatives.

  A typical semi-differential invariant under projective transformations which evaluates the imaged curve at two points and uses derivatives of second order is defined by [BRILL et al. 1992]:

$$I(\mathbf{x_r}, \mathbf{x}) = \frac{\left|\begin{bmatrix}\mathbf{x} & \dot{\mathbf{x}} & \dot{\mathbf{x}}_\mathbf{r}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_r} & \dot{\mathbf{x}}_\mathbf{r} & \mathbf{x}\end{bmatrix}\right|} \sqrt[3]{\frac{\left|\begin{bmatrix}\mathbf{x_r} & \dot{\mathbf{x}}_\mathbf{r} & \ddot{\mathbf{x}}_\mathbf{r}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x} & \dot{\mathbf{x}} & \ddot{\mathbf{x}}\end{bmatrix}\right|}} \tag{2.11}$$

  where $\mathbf{x_r}$ is a reference, and $\mathbf{x}$ an arbitrary point on the curve. Note, that these invariants can be used to gain complex invariant signatures by moving the point $\mathbf{x}$ along the imaged curve, where any measured valued concerning these signatures is an invariant. Other semi-differential invariants are discussed in [RIVLIN and WEISS 1993, VAN GOOL et al. 1992, WEISS 1992].

- *Canonical frames:*
  A different method for generating invariants for non-algebraic curves is to employ a canonical frame construction method. This method has been first applied to affine transformations [LAMDAN et al. 1988] but later has been extended to projective transformations as well [ROTHWELL et al. 1992].

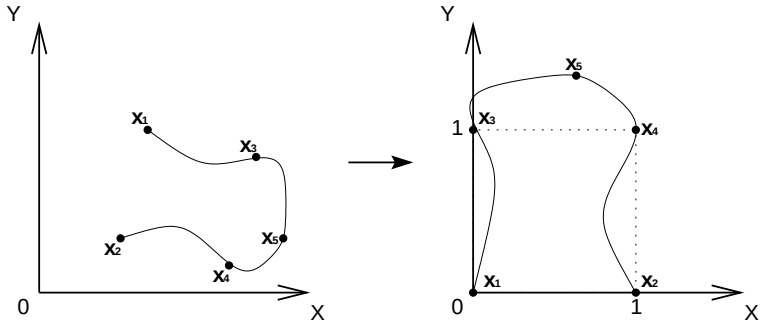


Figure 2.1: Canonical frame construction method

  In the case of plane projective transformations four distinguished points on a curve are sufficient to determine a unique projectivity between those four points and four pre-defined points in the canonical frame, like the corners of the unit square. This projection is then applied to the curve portion containing the distinguished points to yield an invariant signature in the canonical frame. An example for the canonical frame construction method is sketched in Fig. 2.1.

  The canonical frame construction method has been further investigated in [CARLSSON et al. 1996, ROTHWELL et al. 1995b, ZISSERMAN et al. 1992].

- *Fitting of algebraic curves:*
  Since invariants for algebraic curves are well established, it is natural to exploit them for approximating non-algebraic curves. The main problem of this approach is to fit algebraic curves such that the fit is invariant with respect to particular transformation groups. This approach has been investigated for affine transformations [KAPUR and MUNDY 1992] as well as for projective transformations [CARLSSON 1992, FORSYTH et al. 1990].

3D Invariants

As mentioned in Sect. 2.2.3 the generation of 3D invariants is a little bit more complicated, since the perspective transformation of a 3D world onto a 2D image plane results in a complete loss of depth information. Relying on the theorem, that no 3D invariants under such transformations exist for point sets in general position [BURNS et al. 1992], it has been argued for a long time that it is impossible to construct any 3D invariant. However, if the geometric structure of 3D objects is constrained, invariants can be found. In the recent past many invariants for various classes of 3D objects have been generated.

- *Butterfly configuration:*
  The first projective invariant of a 3D geometric configuration that has been proposed is the invariant of the *butterfly configuration* (see e.g. [ZISSERMAN et al. 1994]). Although it is a very simple 3D invariant the underlying construction method can be frequently applied to various types of 3D geometric configurations. The basic idea of this construction method is to reduce 3D geometric structures to planar or linear configurations, for which invariants can be easily measured.

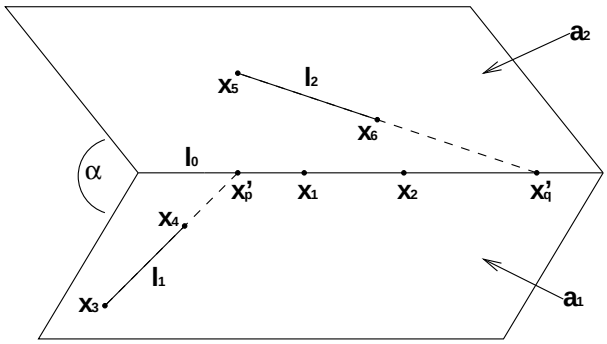


Figure 2.2: Butterfly configuration

The butterfly configuration, shown in Fig. 2.2, is composed of six points making up two four point groups $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}\}$ and $\{\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_5}, \mathbf{x_6}\}$, where the points in each group are coplanar and two points $\mathbf{x_1}, \mathbf{x_2}$ are shared between the groups. As indicated, these six points can be used to construct a linear configuration of four points

$\mathbf{x_1}, \mathbf{x_2}, \mathbf{x'_p}, \mathbf{x'_q}$, for which the cross-ratio can be computed. Nevertheless, the cross-ratio of the six points can also be measured directly using the following algebraic expression [ROTHWELL and STERN 1995]:

$$I(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \mathbf{x_6}) = \frac{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_3} & \mathbf{x_4}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_5} & \mathbf{x_6}\end{bmatrix}\right|} \frac{\left|\begin{bmatrix}\mathbf{x_2} & \mathbf{x_5} & \mathbf{x_6}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_2} & \mathbf{x_3} & \mathbf{x_4}\end{bmatrix}\right|} \tag{2.12}$$

- *Rotationally symmetric objects:*
  The calculation of invariants for generalised curved 3D objects is very difficult. However, if the objects are rotationally symmetric, cross-ratios can be measured using the aforementioned construction method.

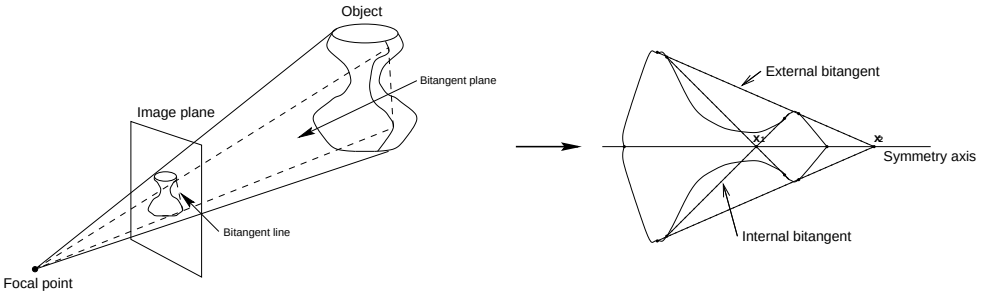


Figure 2.3: Invariants for rotationally symmetric objects

Figure 2.3 indicates the construction process where e.g. bitangents of the imaged curve can be used to generate points on the symmetry axis of the object shape. More details about invariants for rotationally symmetric objects are given in [FORSYTH et al. 1992, FORSYTH et al. 1994].

- *Trihedral polyhedra:*
  The construction method of the butterfly configuration has also been applied to trihedral polyhedra [ROTHWELL et al. 1993]. Trihedral polyhedra are composed of vertices that are always defined by the intersection of three different planes. A simple example for such polyhedra are cube-like objects for which the following three independent cross-ratios can be measured:

$$I_1(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \mathbf{x_6}, \mathbf{x_7}) = \frac{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_3} & \mathbf{x_5}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_4} & \mathbf{x_7}\end{bmatrix}\right|} \frac{\left|\begin{bmatrix}\mathbf{x_2} & \mathbf{x_4} & \mathbf{x_7}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_2} & \mathbf{x_3} & \mathbf{x_5}\end{bmatrix}\right|} \tag{2.13}$$

$$I_2(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \mathbf{x_6}, \mathbf{x_7}) = \frac{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_4} & \mathbf{x_6}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_2} & \mathbf{x_5}\end{bmatrix}\right|} \frac{\left|\begin{bmatrix}\mathbf{x_3} & \mathbf{x_2} & \mathbf{x_6}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_3} & \mathbf{x_4} & \mathbf{x_5}\end{bmatrix}\right|} \tag{2.14}$$

$$I_3(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, \mathbf{x_5}, \mathbf{x_6}, \mathbf{x_7}) = \frac{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_2} & \mathbf{x_7}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_1} & \mathbf{x_3} & \mathbf{x_6}\end{bmatrix}\right|} \frac{\left|\begin{bmatrix}\mathbf{x_4} & \mathbf{x_3} & \mathbf{x_6}\end{bmatrix}\right|}{\left|\begin{bmatrix}\mathbf{x_4} & \mathbf{x_2} & \mathbf{x_7}\end{bmatrix}\right|} \tag{2.15}$$

where $\mathbf{x_i}, 1 \leq i \leq 7$ denote seven visible points of a polyhedron.

In [ROTHWELL and STERN 1995] and [ROTHWELL and STERN 1996] these invariants are described in great detail and it is demonstrated how different types of objects can be recognised by representing the objects as polyhedral cages.

- *Model-based invariants:*
  A quite different kind of 3D invariants has been proposed in [WEINSHALL 1993, WEINSHALL 1994]. Contrary to other invariants presented here, the proposed ones are model-based invariants, i.e. the invariants are generated for each object individually using a uniform construction method which theoretically can be applied to any rigid 3D object.

  The model-based affine invariants are constructed using five 3D model points $\mathbf{p_i}, 1 \leq i \leq 5$. It is assumed without loss of generality that the first four points are not coplanar. Thus, the fifth model point can be expressed in affine invariant coordinates $\mathbf{b}$ as $\mathbf{p_5} = b_1(\mathbf{p_2} - \mathbf{p_1}) + b_2(\mathbf{p_3} - \mathbf{p_1}) + b_3(\mathbf{p_4} - \mathbf{p_1})$. This vector representing the object model is incorporated into an invariant, which is evaluated for five 2D points $(x_i, y_i), 1 \leq i \leq 5$ measured in an image assuming w.l.g. that $x_1 = 0, y_1 = 0$:

$$I_{\mathbf{b}}(\mathbf{x}, \mathbf{y}) = \frac{\left| x_5 - \sum_{i=2}^{4} b_i x_i \right|}{|\mathbf{x}|\,|\mathbf{b}|} + \frac{\left| y_5 - \sum_{i=2}^{4} b_i y_i \right|}{|\mathbf{y}|\,|\mathbf{b}|} \tag{2.16}$$

  where $\mathbf{x}$, $\mathbf{y}$ are given by $\mathbf{x} = (x_2, x_3, x_4, x_5)$ and $\mathbf{y} = (y_2, y_3, y_4, y_5)$.

- *Quasi-invariants:*
  Another type of invariants which can be used for 3D object recognition are quasi-invariants. Actually, quasi-invariants are no invariants in sense of Def. 2.2, since these invariants are not completely unaffected under a particular transformation group, but remain stable for a wide variety of transformations.

  Many quasi-invariants are based on angles or ratios of lengths of straight lines. A theoretical investigation of quasi-invariants is proposed in [BINFORD and LEVITT 1993] and recognition applications are described in [OLSON 1994, OLSON 1995].

2.3.2 Appearance-Based Invariants

Appearance-based or pattern invariants are evaluated for pre-segmented image patches and often take the whole image patch information into account. These invariants can be applied to any type of images such as grey-scale images, edge images, and power spectra.

In the following sections some of the most important methods for invariant pattern recognition are proposed including moment invariants and integral invariants. However, invariant object recognition methods based on structured neural networks will not be discussed, because these recognition methods do not rely on invariant object descriptions but obtain invariance by incorporating the effects of group transformations into the network structure. Additional information about such neural networks as well as on some other invariant recognition methods can be found in [WOOD 1996].

Moment Invariants

Moment invariants are obtained by combining the moments of an image pattern in an adequate way. A moment is a sum of all pixels of the image pattern weighted with polynomials of the pixel positions. Several types of moments have been developed. In the context of this thesis the following section will just describe invariants based on the well-known regular and Zernike moments.

- *Regular moments:*
  The most popular type of moments are *regular moments*. In the discrete form they are defined as:

$$m_{pq} = \sum_{j=1}^{N} \sum_{k=1}^{N} x_j^p \; y_k^q \; f(x_j, y_k) \tag{2.17}$$

  where p, q are non-negative integers and $f(x, y)$ is the input pattern.

  These moments are often used to determine transformations which normalise the input pattern to gain invariance with respect to a particular transformation group. For example, the *centroid* (x_0, y_0) of an image, given by $x_0 = \frac{m_{10}}{m_{00}}, y_0 = \frac{m_{01}}{m_{00}}$, can be used to generate a translation invariant pattern: $f_c(x, y) = f(x + x_0, y + y_0)$. The same method can be applied to provide invariance according to rotations and scaling as well.

  A different way of performing invariant pattern recognition based on regular moments is to combine these moments adequately. In contrast to the aforementioned method it is not required to transform the input pattern, because the moments are used directly. Based on the *central moments*:

$$\mu_{pq} = \sum_{j=1}^{N} \sum_{k=1}^{N} (x_j - x_0)^p \; (y_k - y_0)^q \; f(x_j, y_k) \tag{2.18}$$

  seven different moment invariants have been developed [HU 1962], which are translation, scale and rotation invariant. An example for such an invariant is:

$$I = \frac{\mu_{20} + \mu_{02}}{\mu_{00}^2} \tag{2.19}$$

  Furthermore, a method has been proposed [FLUSSER and SUK 1993, FLUSSER and SUK 1998] to construct invariants under arbitrary affine transformations, which are much more complex. For example the simplest affine moment invariant is given by:

$$I = \frac{1}{\mu_{00}^{10}} \left(\mu_{30}^2 \; \mu_{03}^2 - 6 \; \mu_{30} \; \mu_{21} \; \mu_{12} \; \mu_{03} + 4 \; \mu_{30} \; \mu_{12}^3 + 4 \; \mu_{03} \; \mu_{12}^3 - 3 \; \mu_{21}^2 \; \mu_{12}^2 \right) \tag{2.20}$$

- *Zernike moments:*
  Another type of moments are *Zernike moments*. These moments generally out-perform others, especially in the presence of noise.

Zernike moments are based on *Zernike polynomials* :

$$R_{dm}(r) = \sum_{k}^{(d-|m|)/2} (-1)^k \frac{(d-1)!}{k!(\frac{d+|m|}{2}-1)!\,(\frac{d-|m|}{2}+1)!}\, r^{d-2k} \tag{2.21}$$

and are defined as:

$$Z_{dm} = \frac{d+1}{\pi} \iint\limits_{x^2+y^2\leq 1} f(x,y)\, k(d,m,x,y)\, dx\, dy \tag{2.22}$$

where $d-|m|$ must be even and positive. d represents the order and m the repetition of the moment. The kernel $k(d,m,x,y)$ expressed in polar coordinates is given by:

$$k(d,m,\rho,\varphi) = R_{dm}(\rho)\, e^{-im\varphi} \tag{2.23}$$

The modulus of Zernike moments is rotation invariant. As suggested in [KHOTANZAD and HONG 1990] additional translation and scale invariance can be achieved by normalising the pattern first using regular moments:

$$f'(x,y) = f\left(x\sqrt{\frac{m_{00}}{\alpha}} + \frac{m_{10}}{m_{00}},\; y\sqrt{\frac{m_{00}}{\alpha}} + \frac{m_{01}}{m_{00}}\right) \tag{2.24}$$

Integral Invariants

Invariants of this type are computed by employing integrals in different ways. Integral invariants are mainly based on integral transformations like the Fourier transformation. However, other methods have been developed like the averaging techniques and the image filtering methods.

- *Invariants based on Fourier transformations:*
  One of the best-known methods to generate pattern invariants is based on Fourier and similar transformations, like the Fourier-Mellin transformation.

  For example, in its basic form the two-dimensional continuous Fourier transformation is given by the equation:

$$\mathcal{F}(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-2\pi i(xu+yv)}\, dx\, dy \tag{2.25}$$

  It is well-known that the modulus of $\mathcal{F}$, i.e. the power spectrum, is invariant under translations of function f. Additionally, the power spectrum has the following two properties: firstly, a rotation of function f through an angle α results in a power spectrum which is also rotated through the same angle; and secondly, a scaling of function f by a factor λ leads to a scaled power spectrum by a factor $\frac{1}{\lambda}$. Thus, as first mentioned in

[CASASENT and PSALTIS 1976], rotations and scaling can be mapped to translations by applying a log-polar transformation to the power spectrum so that the modulus of a second Fourier transformation provides invariance according to translations, rotations, and scaling.

Many different recognition methods employing invariants based on Fourier transformations and its variants have been proposed; e.g. see [CASASENT and PSALTIS 1976, DAVOLI et al. 1999, FONGA 1996].

- *Invariants based on averaging techniques:*
  Pattern invariants under a particular transformation group can also be constructed by calculating appropriate averages of an image pattern. This is done by integrating a function $f(p(\mathbf{x}))$ of the image pattern $p(\mathbf{x})$ over the whole transformation group [SCHULZ-MIRBACH 1994]:

$$I_f(p(\mathbf{x})) = \int_G f(p(\mathbf{g}(\mathbf{x}))) \, d\mathbf{g} \tag{2.26}$$

  where $\mathbf{g}(\mathbf{x})$ denotes a transformation of group G. Then, the resulting feature $I_f(p(\mathbf{x}))$ is invariant under the transformation group. Obviously, some restrictions must be imposed on G in order to ensure the convergence of the integral. As proved in [HURWITZ 1897], this can be guaranteed if the group G is locally compact.

  A very simple example for such an invariant under the plane Euclidean transformation group employing the function $f(p(\mathbf{x})) = p(0,0)$ is given by:

$$I(p(\mathbf{x})) = \frac{1}{N^2} \int_{y_0=0}^{N} \int_{x_0=0}^{N} p(x,y) \, dx \, dy \tag{2.27}$$

  This is nothing simply the mean grey value of the input pattern. More sophisticated invariants based on averaging techniques as well as object recognition applications are described in [KRÖNER and SCHULZ-MIRBACH 1995, SCHULZ-MIRBACH 1995]

- *Invariants based on image filtering:*
  In the recent past a new method has been proposed for generating the so-called affine invariant spectral signatures [BEN-ARIE et al. 1996a, BEN-ARIE et al. 1996b]. The idea of this approach is to project an image pattern onto a set of basis functions by convolving the image pattern with a set of kernels.

  Particularly, the method employs a set of derivatives of elliptical 2D Gaussian-based kernels, where the kernels vary in the order of derivatives as well as in their size and orientation. Simply speaking, the convolution of the input pattern with a kernel of the set results in a spectral image representation, where each kernel removes the effects of a particular affine transformation. Obviously, most of the kernels will lead to wrong spectral signatures and only a few of them will match with the spectral signatures of a stored object model. However, it has been demonstrated that the hypothesis generation method based on a combined hashing and voting scheme provide good recognition results even in presence of excessive noise and clutter [BEN-ARIE and WANG 1998].

2.4 Geometric vs. Appearance-Based Invariants

This section discusses the properties of geometric and pattern invariants in greater details. As already mentioned (Sect. 2.2.3) it is possible to construct geometric and pattern invariants that remain unchanged under transformation groups modelling the camera mappings. Therefore, a calibration of camera setups can be generally avoided.

However, although geometric and pattern invariants are based on the same mathematical theory they differ in many of their properties and characteristics. Each approach has its strengths and limitations, so that these methods can complement each other to result in a more robust recognition system.

In the following the main differences between the two approaches are described. These differences principally arise from the different kind of underlying object descriptions. The following discussion anticipates some of the results of Chap. 3 and Chap. 4. Some of these differences have also been proposed in [MUNDY et al. 1996].

- *Scope:*
  As already mentioned in Sect. 2.2.3 the scope of invariants covers three different aspects: firstly, the size of the object domain that can be described by an invariant; secondly, the type of the underlying transformation group; and lastly, if the corresponding invariant object descriptions are local, semi-local, or global. Generally, geometric and pattern invariants differ in any of these senses:

  Since geometric invariants are calculated for particular geometric structures, they can be employed to recognise only those objects for which the underlying geometric structures can be extracted. For example, to recognise a 2D or a 3D object using the projective invariants of a pair of coplanar conics (2.5), (2.6) it must be possible to extract at least two conics for the object lying in the same plane. In this respect pattern invariants are more universal. They can be used to recognise many different types of objects regardless of their shape and texture. The only requirement is that the objects must possess at least one (quasi-)planar area for which the pattern invariants can be computed.

  Conversely, geometric invariants are more flexible concerning the underlying transformation group. It is possible to generate geometric invariants under various types of transformations, especially for projective transformations which are often used to approximate the camera mappings. However, the geometric structures, for which the invariants should be constructed, must provide a sufficient degree of complexity, i.e. they must meet the counting argument (see Theor. 2.2). In contrast, pattern invariants are often restricted to translations, rotations, and scaling and only few of the pattern invariants remain unaffected under affine transformations.

  Furthermore, geometric invariants can be used to generate global, semi-local, and local object descriptions, where local or semi-local invariants are necessary to recognise objects under partial occlusion. Pattern invariants generally tend to be global but can be applied to subparts of an object, so that to some degree partial occlusion can also be handled.

- *Efficiency:*
  Geometric and pattern invariants also differ in their efficency, i.e. the computational cost for determining the invariant object descriptions.

  Obviously, geometric invariants can be computed very efficiently, because they are based on structures which are abstracted from the image information and are determined by a small number of parameters. Nevertheless, geometric invariants require a very time-consuming pre-processing stage containing the extraction of geometric primitives which must represent the topology of images adequately, as well as the grouping process to compose the geometric structures for which the invariants can be evaluated (see Sect. 3.3).

  Pattern invariants generally take the whole image information into account and thus, require a large amount of computations. Nevertheless, only rudimentary segmentation algorithms are needed to determine the image patches used for evaluation. As a result, pattern invariants are generally much more efficient than geometric invariants, if one takes the pre-processing stages into consideration.

- *Stability:*
  The stability of invariants is concerned with the reliability of invariant object descriptions under non-optimal imaging conditions. The imaging process is always affected by noise as well as by discretisation and quantisation effects. Additionally, the objects visible in complex scenes often partially overlap each other. This make it impossible to gain complete descriptions of the objects.

  Again, geometric and pattern invariants behave in different ways. On the one hand geometric invariants work quite well in presence of occlusion because they provide local and semi-local object descriptions. On the other hand they are sensitive to noise and clutter. The stability of geometric invariants heavily depends on the pre-processing stages of the object recognition system. If the system cannot extract the geometric features of an image robustly and reliably, the features will not describe the topology on an image adequately, so that the system will fail to recognise the corresponding objects correctly.

  The behaviour of pattern invariants is the converse. They are very stable with respect to noise as well as to quantisation and discretisation effects, because they rely on the whole information provided by an image patch, but they are susceptible to occlusion. Although pattern invariants can be limited to subparts of an object, the position and size of image patches extracted by segmentation algorithms are often affected by occlusion. Furthermore, segmentation algorithms depend also on illumination conditions and are affected by mutual shadows.

The differences between geometric and pattern invariant object descriptions naturally lead to the necessity to deal with both approaches, since they can be used to complement each other. Therefore, this thesis will investigate recognition methods based on either type of invariants. The methods will be incorporated into a single object recognition system that fuses the object hypotheses obtained by the methods in order to yield more robust recognition results.

3

Object Recognition Using Geometric Invariants

This chapter concentrates on object recognition systems based on geometric invariants. In contrast to other recognition systems of this kind the one developed in this chapter utilises a hypothesis generation method based on fuzzy set theory. This hypothesis generation method is called fuzzy invariant indexing technique (FII), since it employs fuzzy if-then classification rules and fuzzified invariant object descriptions, which can be either fuzzy invariant values or fuzzy invariant signatures. The FII-technique enhances the commonly used indexing techniques based on geometric invariants and provides a great degree of flexibility. A first object recognition system based on the proposed FII-technique is developed and experimental results demonstrate the performance and the flexibility of the proposed approach. Furthermore, some additional problems that have been encountered are discussed in more details.

3.1 Motivation

After having described some of the main aspects and features of employing invariants in computer vision, this chapter concentrates on object recognition systems that are based on geometric invariants.

The structure of such systems is similar to other model-based recognition systems and can be divided into the following processing steps: At first the input images are pre-processed to remove noise and clutter. Then, the geometric features are extracted, which are used in the

following step to generate invariant object descriptions. In the hypothesis generation stage these invariant object descriptions are compared with the descriptions stored in a model base and appropriate hypotheses are generated when correspondences have been found. Finally, the generated object hypotheses are verified in order to suppress false positives, i.e. incorrectly generated hypotheses.

Although the invariant approach facilitates the generation of object hypotheses, some care must be taken. In contrast to the theoretical expectations of invariant theory every object recognition system based on invariants has to cope with fluctuating invariant object descriptions, mainly caused by noisy imaging hardware, quantisation and discretisation effects as well as by inaccurate feature extraction. Usually, this problem is handled by generating object hypotheses not only for invariant descriptions observed in an image matching exactly the pre-defined descriptions of stored object models, but also for invariant descriptions which are slightly different.

In particular, most recognition systems employ hypothesis generation methods based on indexing techniques, which can only utilise invariant values. In these techniques the invariant values of geometric structures observed in images are used to hash into a discrete index space. To overcome the fluctuation all points of the index space belonging to an object are marked as well as their proximate points. Indexing techniques like these are used in the system based on the geometric hashing technique [LAMDAN and WOLFSON 1988, WOLFSON 1990], the LEWIS-system [ROTHWELL 1995b] and the MORSE-system [MUNDY et al. 1994a, MUNDY et al. 1995].

The nature of such hypothesis generation methods entails some drawbacks: The first problem arises, when the invariant values of geometric structures lie at the boundaries within the areas of marked points. In these cases object hypotheses will be generated. However, if the geometric structures are slightly different the corresponding invariant values might leave the areas and no object hypothesis will be generated any further, although the underlying geometric structures are very similar to each other. Such a behaviour is generally dubious and undeliberate. Another problem is concerned with the discrimination ability of the recognition system. Since these hypothesis generation methods produce object hypotheses with equal weights, the possibility to identify the most likely hypotheses is diminished. Finally, the traditional indexing methods can employ invariant values only. Therefore complex invariant signatures must be reduced to single invariant values which result in a great loss of information.

In contrast, this chapter proposes a hypothesis generation method based on fuzzy set theory, which avoids sharp boundaries and produces object hypotheses with different credibilities. This method is called *fuzzy invariant indexing (FII)* technique, since it employs fuzzy if-then classification rules and fuzzy invariant object descriptions, which can be either fuzzy invariant values as well as fuzzy invariant signatures. Additionally, the FII-technique can be extended by adding new attributes (also non-invariant attributes like colour). The following chapter presents the general framework of the fuzzy invariant indexing technique as well as its applicability to object recognition systems, where some of the items concerning the FII-technique have already been proposed in [GRAF et al. 1998b, GRAF et al. 1998c, GRAF et al. 1998a, KNOLL et al. 2000].

3.2 Fuzzy Invariant Indexing (FII)

This section describes in great detail the general framework of the fuzzy invariant indexing technique, which is based on fuzzy invariant object descriptions and fuzzy if-then classification rules (for a good introduction to fuzzy sets and fuzzy logic, see [KLIR and YUAN 1995]). The section shows how fuzzy sets can be utilised to model the uncertainty of invariant object descriptions and how to perform the generation of object hypotheses by applying fuzzy classification rules.

3.2.1 Fuzzy Classification Rules

In the FII-technique object hypotheses are generated by evaluating fuzzy if-then classification rules, which take the invariant object descriptions observed in an image as input and produce object classes with corresponding credibilities as output. These fuzzy classification rules have the general form:

$$\text{IF } i^k_{m1} = \widetilde{I}^k_{m1} \text{ AND } \cdots \text{ AND } i^k_{mN^k_m} = \widetilde{I}^k_{mN^k_m} \text{ THEN } o^k_m = \widetilde{O}^k \tag{3.1}$$

$$\begin{array}{rrll} & k & = & 1, 2, \ldots, K & \text{(object classes)} \\ \text{with} & m & = & 1, 2, \ldots, M^k & \text{(sub-rules for class } k\text{)} \\ & n & = & 1, 2, \ldots, N^k_m & \text{(conditions of a sub-rule)} \end{array}$$

where i^k_{mn} denotes the n-th input variable of sub-rule m for the k-th object, $\widetilde{I}^k_{mn}$ the corresponding fuzzy invariant object description (see Sect. 3.2.2), o^k_m the output variable of sub-rule m and $\widetilde{O}^k$ the k-th object class modelled as a fuzzy singleton. The total amount of conditions N^k_m of a rule depends on the number of functional independent invariants of the underlying geometric configuration (see Sect. 2.2) of sub-rule m for object k and the total amount of sub-rules M^k depends on the number of different geometric configurations for object k.

In order to generate object hypotheses these fuzzy if-then classification rules are inferred in the following way:

$$\mu_{o^k_m} := \min_{1 \le n \le N^k_m} \mu_{\widetilde{I}^k_{mn}}(i^k_{mn}) \tag{3.2}$$

where $\mu_{\widetilde{I}^k_{mn}}(i^k_{mn})$ denotes the membership function of the fuzzified invariant object description $\widetilde{I}^k_{mn}$, which is evaluated for the invariant description i^k_{mn} observed in an image. The resulting membership values $\mu_{o^k_m}$ of the sub-rules of object k are combined disjunctively:

$$\mu_{o^k} = \max_{1 \le m \le M^k} \mu_{o^k_m} \tag{3.3}$$

The final result is the indexed k-th object model with the measured credibility μ_{o^k}.

3.2.2 Fuzzy Invariant Object Descriptions

The main problem in generating the fuzzy classification rules (3.1) is to find appropriate fuzzy membership functions to model the fuzzy invariant object descriptions $\widetilde{I}^k_{mn}$ for a given object. Generally, two different types of descriptions must be distinguished: fuzzy invariant values and fuzzy invariant signatures.

Fuzzy Invariant Values

The most commonly used invariant object descriptions are invariant values. These values result from evaluating invariant functions based on geometric structures which can be extracted for an object to be recognised (see Sect. 2.3).

Although systematic errors might occur during the feature extraction stage of the recognition system, the investigation of invariant values measured in different perspective views (see also Fig. 3.2) has indicated that the fluctuations can be adequately approximated by bell-shaped membership functions $\mu_{\widetilde{I}^k_{mn}}$:

$$\mu_{\widetilde{I}^k_{mn}}(u) = e^{-\frac{(u-\alpha^k_{mn})^2}{2\beta^{k\,2}_{mn}}}, \quad u \in \mathbb{R} \tag{3.4}$$

where the parameters $\alpha^k_{mn}, \beta^k_{mn}$ determine the shape of the functions. The advantage of employing these membership functions to model fuzzy invariant values is that they can be efficiently evaluated. Furthermore, the shape is determined by only two independent parameters which can be easily adjusted or learned in an acquisition process by choosing the parameters in the following way:

- The parameter α^k_{mn} determines the position of the maximum of the bell-shaped function (3.4). Therefore, this parameter should be the mean of the fluctuating invariant values: $\alpha^k_{mn} = \frac{1}{N}\sum_l I_l$, where I_l, $1 \leq l \leq N$ are the invariant values for an object taken in N different images.
- The parameter β^k_{mn} determines the position of the inflexions of (3.4), which are located at $\alpha \pm \beta$. This parameter should be the standard deviation of the invariant values: $\beta^k_{mn} = \left(\frac{1}{N}\sum_l (I_l - \alpha^k_{mn})^2\right)^{\frac{1}{2}}$.

For example, Fig. 3.1 shows the distribution of invariant values for the object *nut* of the Baufix domain (Appendix B) observed in 30 different images. These invariant values have been measured using the three functional independent invariants of the geometric structure of one ellipse and three straight lines, where the invariant values on the x-axis are calculated using Eq. (2.8), the invariant values on the y-axis are calculated using Eq. (2.9) and the invariant values on the z-axis are calculated using Eq. (2.10).

The histograms of these invariant values are depicted at the left hand side of Fig. 3.2. The use of the aforementioned construction method leads to the parameters $\alpha^k_{11} = 1.4$,

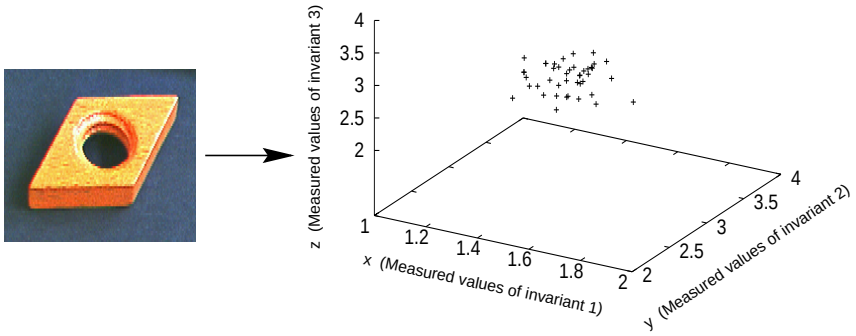


Figure 3.1: Invariant values of a conic and three lines for a *nut* observed in 30 images

$\beta_{11}^k = 0.086$ for the first fuzzy invariant value, $\alpha_{12}^k = 3.2$, $\beta_{12}^k = 0.145$ for the second fuzzy invariant value, and $\alpha_{13}^k = 2.5$, $\beta_{13}^k = 0.14$ for the third fuzzy invariant value, respectively. The corresponding fuzzy invariant values are shown on the right hand side of Fig. 3.2.

Thus, the resulting fuzzy if-then classification rule for the object *nut* can be written in a human readable form as:

$$\text{IF (inv1} \approx 1.4) \text{ AND (inv2} \approx 3.2) \text{ AND (inv3} \approx 2.5) \quad (3.5)$$
$$\text{THEN (object IS nut)}$$

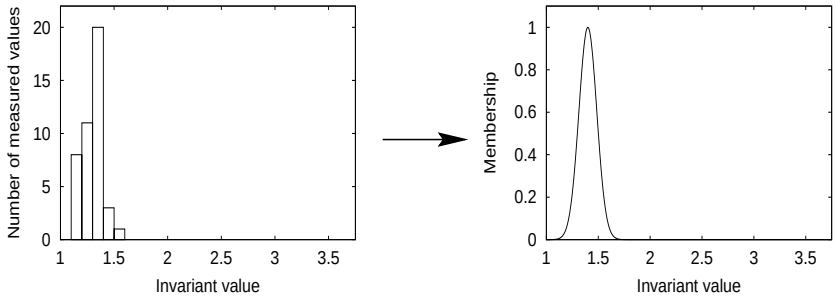
Further classification rules for several objects of the Baufix domain and their corresponding parameters that have been used to obtain the experimental results in Sect. 3.4 are listed in Appendix C.

Fuzzy Invariant Signatures

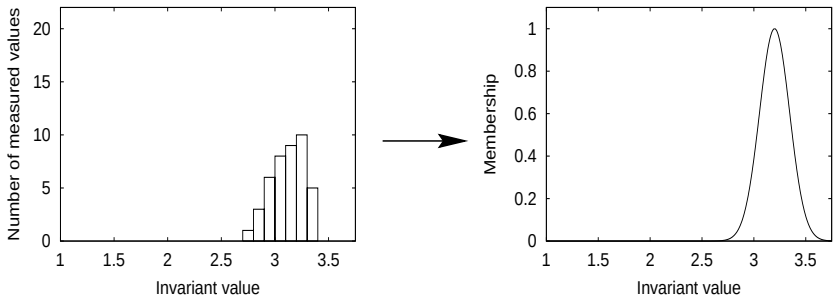
The other type of invariant object descriptions are invariant signatures which are not only single invariant values but rather complex invariant curves. Generally, these invariants are constructed by employing differential invariants or by mapping an extracted image curve into a distinguished coordinate frame using e.g. the canonical frame construction method (Sec. 2.3.1).

Invariant signatures are rarely employed in object recognition. However, in situations where object structures cannot be described adequately or reliably by simple geometric primitives, invariant signatures are more likely to be used. An example is the jigsaw puzzle mentioned in [ROTHWELL et al. 1992].

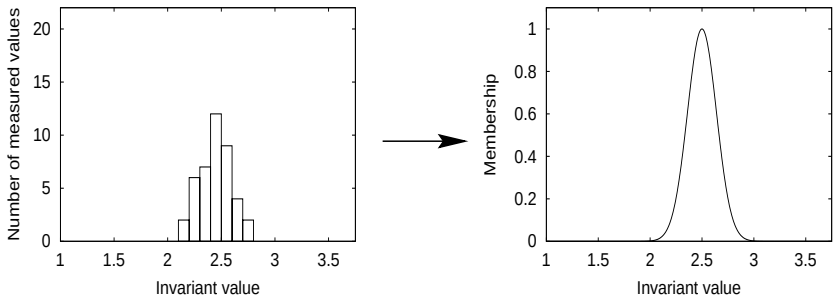
Since most of the invariant indexing techniques cannot utilise complex curves as indices, invariant signatures are reduced to single invariant values. This is done by evaluating the



(a) Histogram and resulting fuzzy invariant value for invariant 1



(b) Histogram and resulting fuzzy invariant value for invariant 2



(c) Histogram and resulting fuzzy invariant value for invariant 3

Figure 3.2: Generation of fuzzy invariant values using the measured data of Fig. 3.1

curves at some pre-defined positions or by using global measures like areas or moments. Obviously, all these methods entail a great loss of information, which may impair the discrimination ability of object recognition systems.

In contrast the FII-technique holds the potential for performing much better. It enables the use of fuzzy membership functions which take the whole invariant signatures into account. Although the following method has neither been implemented nor tested, it is proposed for the sake of completeness. Nevertheless, further sophisticated approaches may be possible.

If $\mathbf{m}(x)$ denotes the stored invariant signature of an object model and $\mathbf{i}(x)$ is an observed image signature then the method proceeds as follows:

1. Represent the observed invariant signature $\mathbf{i}(x)$ as a discrete parameterised curve $\mathbf{i}_k, 1 \leq k \leq N$, where N denotes the normalised length of the signature.
2. Compare $\mathbf{i}_k$ with the discretesized model signature $\mathbf{m}_k$ by a least square distance measure:

$$D_{\mathbf{m}}(\mathbf{i}(x)) = \frac{1}{N}\sqrt{\sum_{k=1}^{N} |\mathbf{m}_k - \mathbf{i}_k|^2} \tag{3.6}$$

3. Depending on the value of the dissimilarity measure $D_{\mathbf{m}}(\mathbf{i}(x))$, determine the credibility for the object that is described by the model signature $\mathbf{m}(x)$ (see [POPOVIĆ and LIANG 1994]):

$$\mu_{\mathbf{m}}(\mathbf{i}(x)) = \begin{cases} 1 - \frac{D_{\mathbf{m}}(\mathbf{i}(x))}{D}, & 0 \leq D_{\mathbf{m}}(\mathbf{i}(x)) \leq D \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

   where D is an upper bound for tolerated distances.

Note, that this matching algorithm can be easily applied to the FII-technique by defining the membership functions of the fuzzy invariant object descriptions $\widetilde{I}^k_{mn}$ in (3.1) as the membership functions (3.7). During the recognition process the hypothesis generation is realized in the same way as described in Sec. 3.2.1.

3.3 FII-Recognition System

The proposed fuzzy invariant indexing technique can be easily integrated into an object recognition system. As a first testbed for the FII-technique an object recognition system, of which the modular system structure is sketched in Fig. 3.3, has been developed. This structure is similar to other recognition systems based on geometric invariants, like [LAMDAN and WOLFSON 1988] and [ROTHWELL 1995b], but differs in the fuzzy rule base which is obviously present to conform with the fuzzy invariant indexing technique.

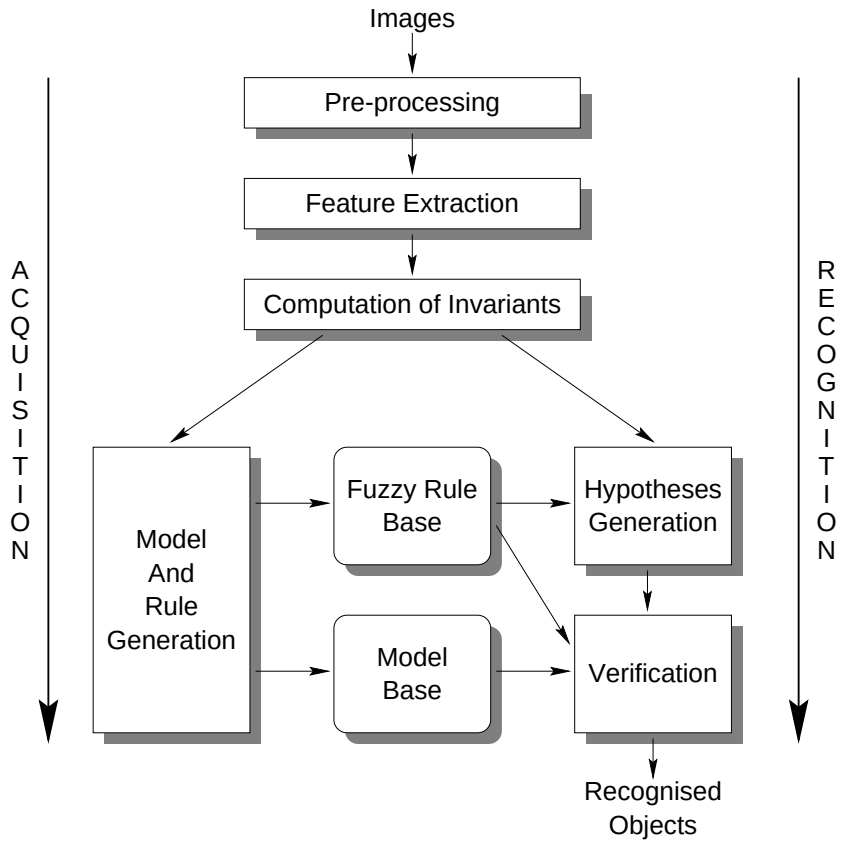


Figure 3.3: FII-recognition system

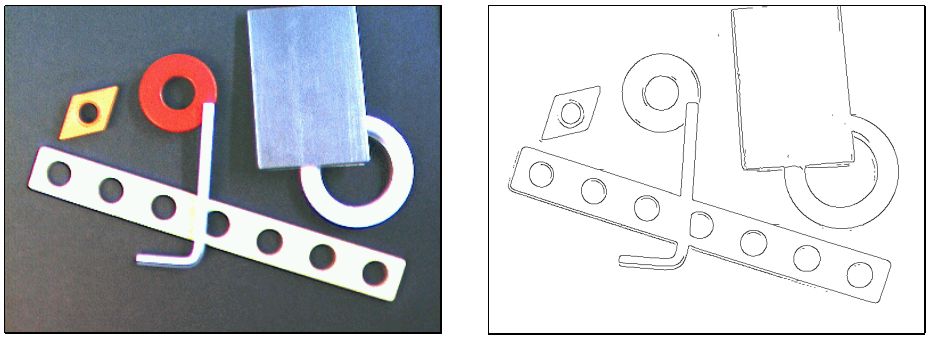
As indicated in Fig. 3.3, the system provides two different processing phases: an off-line acquisition process to learn the fuzzy if-then classification rules automatically and an on-line object recognition process.

To provide a deeper insight into the system design the different processing modules are discussed in some more detail:

- *Pre-processing:*
  In the pre-processing stage of the FII-object recognition system the edge points of an input image are extracted. This pre-processing stage directly influences the recognition performance of the whole system: if the edge points of an image cannot be extracted robustly and reliably, the fitted features of the following stage will not represent the topology of the scene adequately and the system will fail to recognise the corresponding objects.

In the implemented system the edge points are extracted by using a modified Canny edge operator [CANNY 1983]. This edge operator is based on an operator described in [ROTHWELL et al. 1994, ROTHWELL et al. 1995], which has been extended to be applied not only to grey scale but to colour images as well. The operator is composed of the following steps:

At first the edge operator determines the gradient images for the different bands of the smoothed input image. This is done by convolving the input image with the partial derivatives of a 2D Gaussian. Afterwards the gradient images are merged by applying a maximum operator, and a non-maxima suppression is performed to provide the local maxima of the resulting gradient image as possible edge points. Finally, a dynamic thresholding with hysteresis is applied, where the dynamic thresholds are determined according to the strongest edge points in a pre-defined neighbourhood.



(a) Original image (b) Edge image

Figure 3.4: Extraction of an edge image using the extended Canny operator

An example of an edge image which has been generated by using the extended Canny edge operator is demonstrated in Fig. 3.4. Figure 3.4a shows the input image and Fig. 3.4b the resulting edge image.

- *Feature Extraction:*
  In the feature extraction stage geometric primitives are fitted to the extracted edge points. Generally, the geometric primitives that are utilised depend on the object domain to be recognised, where on the one hand the primitives should be as simple as possible to be extracted efficiently and on the other hand they should be complex enough to describe the objects adequately.

  For the Baufix object domain (see Appendix B) straight lines and ellipses are suitable. Since it simplifies and speeds up the following processing stages the object recognition system is restricted to recognise only those objects which can be modelled as 2D geometric structures (i.e. *nuts*, *rims*, *tyres* as well as all types of slats). For example the *rim* and the *tyre* can be modelled as a pair of coplanar ellipses and the *3-holed-slat* can be modelled as a geometric structure of four straight lines and three ellipses.

While the fitting of straight lines to continuous edge curves is a simple task, for which several robust algorithms have been developed, the fitting of ellipses (or, more general, conics) is much more difficult. In Addition more recently proposed ellipse fitting methods, like [HO and CHEN 1995], fail, if only about 25% of the edge points are missing. Therefore, the implementation of the FII-recognition system incorporates the line fitting and the ellipse fitting algorithms of the LEWIS system, which generally produce good results, e.g. the ellipse fitting algorithm is able to fit an ellipse adequately if only about 50% of the edge points are visible. These fitting algorithms carry out a linear regression and are originally based on the fitting algorithm of Bookstein [BOOKSTEIN 1979]. For more details about the fitting algorithms see [ROTHWELL 1995b].

Due to noise and clutter as well as to occlusion the extracted edges of the last stage and the fitted features are often interrupted. Thus, the feature extraction stage provides an additional merging process to link disconnected features.

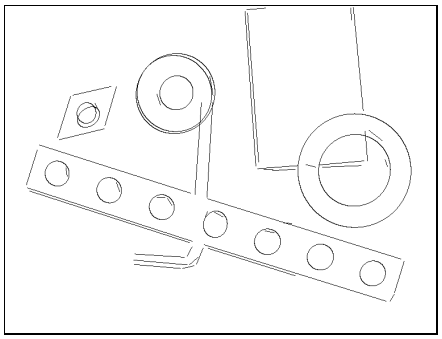


Figure 3.5: Fitted lines and ellipses to edge image of Fig.3.4b

Figure 3.5 shows the final result of the fitting process, where the features have been fitted to the edge points of Fig. 3.4b. In particular, 14 ellipses and 50 straight lines have been found. Note, that the ellipses of the *tyre* have been extracted very accurate even though the tyre is partially occluded.

- *Calculation of Invariants:*
  After having extracted the geometric primitives, the invariants can be calculated. The type of invariants that are utilised to discriminate between the different objects of the object domain, depends on the features that are used to model the objects and on the expected imaging conditions, e.g. the particular transformation group appropriate to approximate the camera mappings.

  Because the FII-system should recognise the given objects modelled as geometric structures of straight lines and ellipses under partial occlusion without making any assumptions about the intrinsic and extrinsic camera parameters, the system exploits the two semi-local projective invariants of a pair of coplanar conics (2.5), (2.6) as well as the three invariants of a conic and three lines (2.8)-(2.10). Therefore, the geometric primitives of the last processing stage must be grouped into two different

types of geometric structures: the structures composed of pairs of ellipses and of single ellipses and three lines.

Such grouping processes are a well-known problem in model-based computer vision: on the one hand it is not a priori known, which features result from the same object and should be grouped together, and on the other hand it is generally not applicable to generate all possible feature groups. For example, $\binom{n}{k} = \frac{n!}{(n-k)!\ n!}$ different geometric structures can be generated, given that n features are present in an image and k of them are used to make up a geometric structure.

Several algorithms have been developed to reduce the complexity of grouping processes, which mainly exploit local relationships between the features. Some of them, like [HUTTENLOCHER 1988] which is based on connectivity of image features, reach linear computational effort. Unfortunately, these algorithms may fail in presence of excessive occlusion, because the features of objects can be far apart so that the system might not be able to recognise such objects correctly. However, one can argue that this problem arises infrequently, because objects are modelled using several semi-local invariant object descriptions.

In particular, the implemented FII-system relies on two different grouping algorithms: For the invariants of pairs of ellipses, however, all possible geometric structures are generated. This is practicable, because the number of ellipses found in images under consideration is in the order of a few tens. Furthermore, this grouping process guarantees that all possible hypotheses based on two ellipses are found.

For the invariants of single ellipses and three lines, a proximity-based grouping algorithm is used. This algorithm only groups ellipses with three straight lines making up a connected line chain. Therefore, this grouping process generates $n_E \times \binom{n_L}{3}$ in its (very unlikely) worst case, where n_E is the number of ellipses and n_L is the number of straight lines. For example, the 14 fitted ellipses and 50 lines of Fig. 3.5b are used to generate 91 invariants of pairs of ellipses and 742 invariants of single ellipses and three lines.

- *Model and Rule Generation:*
  The model and rule generation stage is responsible for learning new objects. All information that is required to recognise an object, i.e. the object model and the fuzzy classification rules, can be obtained entirely from a set of training images and hence, objects can be learned automatically. This directly results from using invariant shape descriptions to model (quasi-)planar objects which can be measured in any image without regarding intrinsic and extrinsic camera parameters.

  An overview of the implemented acquisition process is as follows: Firstly, a number of training images of an unoccluded object is taken from a variety of viewpoints. The images should only contain the new object to be learned, since the system cannot decide which part of the image corresponds to the object of interest. Next, the invariants of all images are calculated by using the processing algorithms of the pre-processing, the feature extraction, and the invariant calculation stage. Afterwards these invariants are compared with each other using a clustering process. When the number of invariants

in a single cluster exceeds a pre-defined value, new fuzzified invariant object descriptions and fuzzy classification rules are generated as described in Sect. 3.2. Since the fuzzy rules are human readable and writable, the classification rules can also be generated and adjusted manually.

Furthermore, an object model is generated by storing the name of the object as well as its features and the corresponding invariant descriptions. The features and invariant object descriptions are determined for a distinguished view from a fronto-parallel position, which generally leads to reliable results.

Using these model and rule generation methods 17 different rules have been automatically generated. These rules are shown in Appendix C, where some of them have been manually adjusted to gain a better discrimination ability between the objects. It must be noted, that these 17 rules are sufficient to enable the recognition of most of the Baufix objects (see also Sect. 3.4).

- *Hypothesis Generation:*
  In this stage the main part of the object recognition process is accomplished: the generation of appropriate object hypotheses.

  This is done by using the measured invariant values of the invariant calculation stage to evaluate the classification rules of the fuzzy rule base (as described in Sect. 3.2.1). If the credibility of an indexed object is above a threshold a new object hypothesis is generated, where the threshold is used to remove the very unlikely hypothesis only. The result of this process are object hypotheses which are composed of the object name, the credibility of the hypothesis (coming from the evaluation of the fuzzy classification rules) and the geometric structure which has been used to calculate the invariants.

  For the 833 invariants which have been measured employing the features shown in Fig. 3.5b the evaluation of the classification rules (see Appendix C) leads to 230 different hypotheses.

- *Verification*
  The last stage of the object recognition system is the verification stage. In this stage the system analyses the object hypotheses in great detail and decides whether a hypothesis should be accepted or rejected.

  The reason for performing a verification mainly arises due to: (i) the non-completeness of the employed invariants, i.e. two geometric structures may result in the same invariant description although they are not equivalent with respect to the transformation group; (ii) objects can share the same local descriptions (like the slats of the Baufix domain); and (iii) similar geometric structures indicate the same object to compensate the fluctuation of the invariant object descriptions.

  In the implemented system the verification step first checks the initial hypotheses and tries to expand them by further supporting image features. This is done in the following way:

  The object hypotheses of the previous processing stage are sorted according to their credibilities to enable the verification of the most credible hypothesis first. Next, the

system determines the transformations that project the model features to the image features stored in the hypotheses. This requires to solve a correspondence problem first, i.e. each image feature must be assigned to an object feature. Fortunately, the invariant object descriptions indicate not only the object that could have been feasibly created the observed image features but also the particular geometric structure of the object model.

Afterwards the transformations are applied to the model features to check if they coincident with the corresponding image features. This checking is performed by simple geometric tests, e.g. the angle between a backprojected model line and the corresponding image line must be below a threshold and corresponding ellipses must have a similar size and location. If a hypothesis succeeds, the whole object model is transformed into the image plane and the hypothesis will be expanded by further supporting image features. All other hypotheses that correspond to the same object model and share more than one image feature with the analysed one, are left out of consideration to speed up the recognition process.

At a second stage the object hypotheses which have been successfully analysed so far are verified against each other: First, all object hypotheses are accepted for which all model features have been found in the image (except for the objects which are just composed of two ellipses). Intersecting hypotheses, which share more than one image feature with the accepted hypotheses, are removed. Next, the remaining hypotheses are sorted according their sizes, i.e. the number of model features which have been found in the image, as well as according to their refined beliefs, i.e. the number of found image features divided by the number of model features. Again, the most likely hypotheses are accepted while intersecting ones are removed. Additionally, the system also checks relations among the hypotheses. For example, it is impossible in the setup to observe two different tyres in an image, where one tyre is twice as large as the other one.

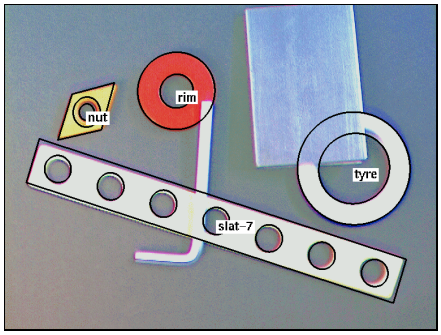


Figure 3.6: FII result for test image Fig. 3.4a

This verification process finally leads to the accepted object hypotheses. An example for a recognition result is shown in Fig. 3.6. As can be seen, the system is able to recognise all Baufix objects correctly, although some objects in the scene are partially occluded.

3.4 Experimental Results

This section demonstrates and discusses the recognition performance and flexibility of the proposed FII-technique in great detail. It shows recognition results for various test scenes of varying complexity and compares the FII-technique with a traditional invariant indexing method. Additionally, it demonstrates, how the fuzzy classification rules can be easily expanded by adding new (also non-invariant) attributes.

3.4.1 Performance of the FII-Recognition System

The recognition performance of the FII-recognition system has been inspected for a subset of the Baufix object domain (Appendix B). This subset contains objects that can be adequately modelled by planar geometric structures of straight lines and ellipses including the *rims*, *tyres*, *nuts*, *3-holed-slats*, *5-holed-slats*, and *7-holed-slats* shown in Fig. 3.7.

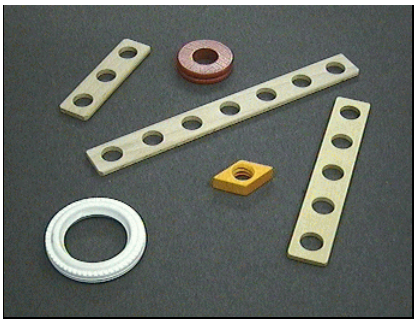
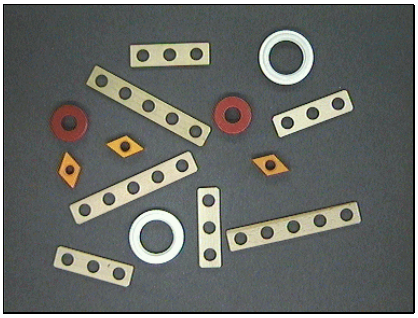


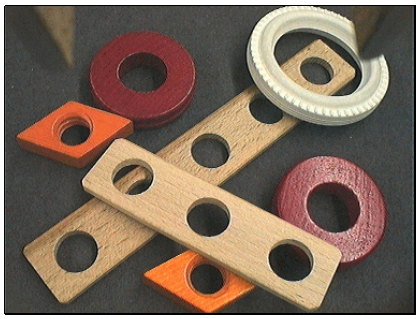
Figure 3.7: Test object domain for the FII-recognition system

Although each of the six sides of the Baufix *cubes* as well as the *screw heads* can be modelled by planar geometric structures, they are not part of the subset. Since the corners and edges of the *cubes* are very smooth, the feature extraction stage of the recognition system cannot extract the features reliably and will fit either ellipses or straight lines depending on the viewpoint. While a similar problem arises for the angular *screw heads*, the geometric structures that can be fitted to the round *screw heads* do not comply with the counting argument of invariant theory (see Sect. 2.2.2). That means, it is impossible to construct any 2D projective invariant which can be utilised for recognition.

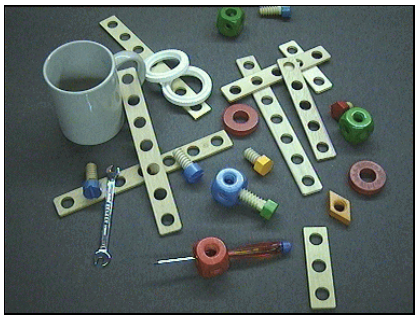
The subset of the Baufix domain has been observed in 360 test images taken with three different cameras of the assembly cell shown in Fig. 1.1 including (i) the top-view camera which always takes images perpendicular to the object surfaces, (ii) the front-view camera taking pictures at an angle of about 45°, and (iii) a hand-camera which has been moved into different positions and often produces images with great perspective distortions. The complexity of the test images varies from test scenes, which are only composed of unoccluded



(a) Top-view image



(b) Hand-camera image



(c) Front-view image

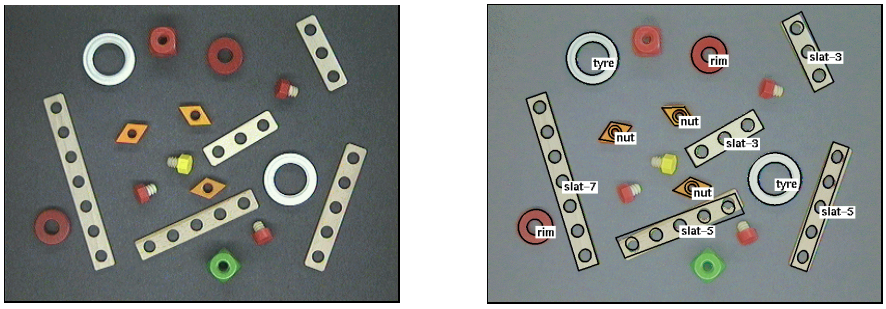
Figure 3.8: Images of varying complexity taken with different cameras

known objects, to scenes including also partially occluded and unknown objects. Figure 3.8 provides an impression of the diversity of the test scenes.

Results for Test Scenes Containing Only Unoccluded Objects

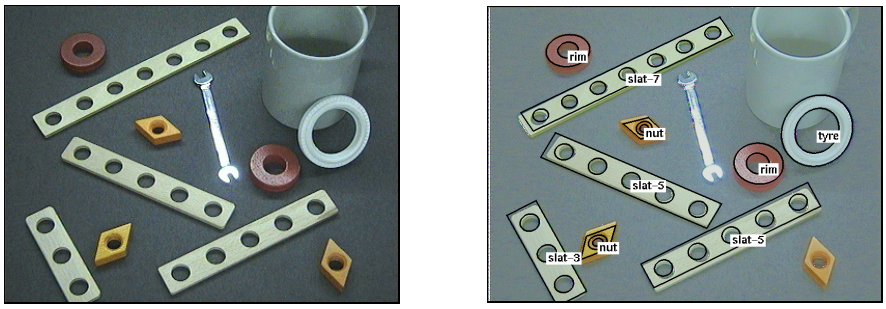
First, the FII-recognition system is applied to a set of test scenes, which are composed of unoccluded known and unknown objects. The unknown objects present in the scenes include Baufix *cubes* and *screws*, which have not been learned in an acquisition process, as well as other objects like a mug, a screw driver, keys, and a pen. Although the test scenes have been taken with different cameras from various viewpoints, the implemented recognition system generally produces good results.

For example, Figs. 3.9 – 3.11 show recognition results for test scenes which originate from the top-view camera (Fig. 3.9), from the front-view camera (Fig. 3.10), and from a hand-camera (Fig. 3.11). Note, that the viewpoint between the test scenes has changed with the result of different perspective distortions; e.g. the size of the objects visible in the images differs in a scaling factor of about $3 \sim 4$. Nevertheless, the system is able to recognise nearly all of the known objects correctly.



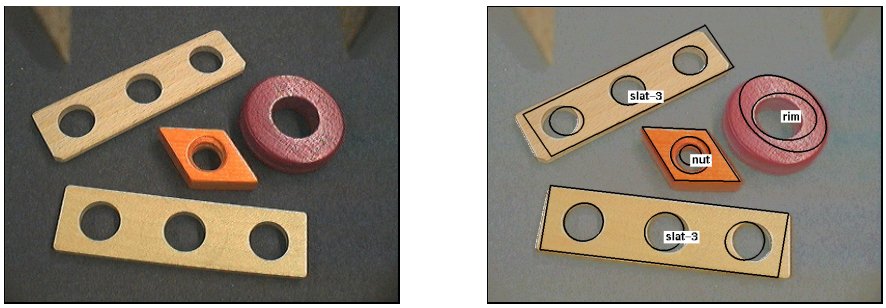
(a) Original image (b) Result

Figure 3.9: FII result: unoccluded objects taken with a top-view camera



(a) Original image (b) Result

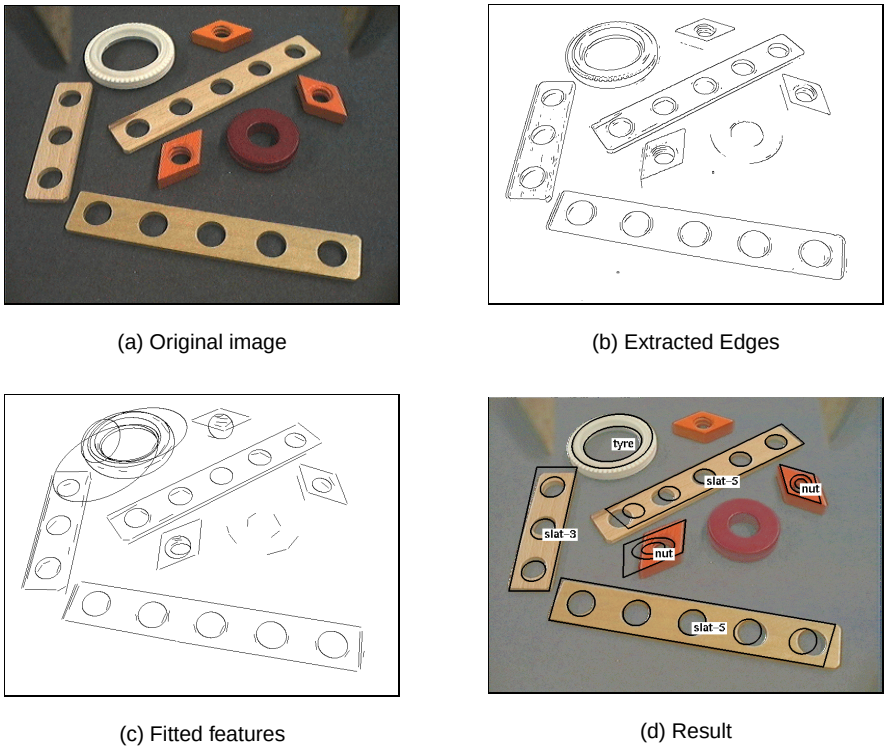
Figure 3.10: FII result: unoccluded objects taken with a front-view camera



(a) Original image (b) Result

Figure 3.11: FII result: unoccluded objects taken with a hand camera

In some situations the recognition system fails to recognise all objects present in an image, which either leads to *false negatives*, i.e. unrecognised objects, or to *false positives*, i.e. incorrect object instances.



(a) Original image

(b) Extracted Edges

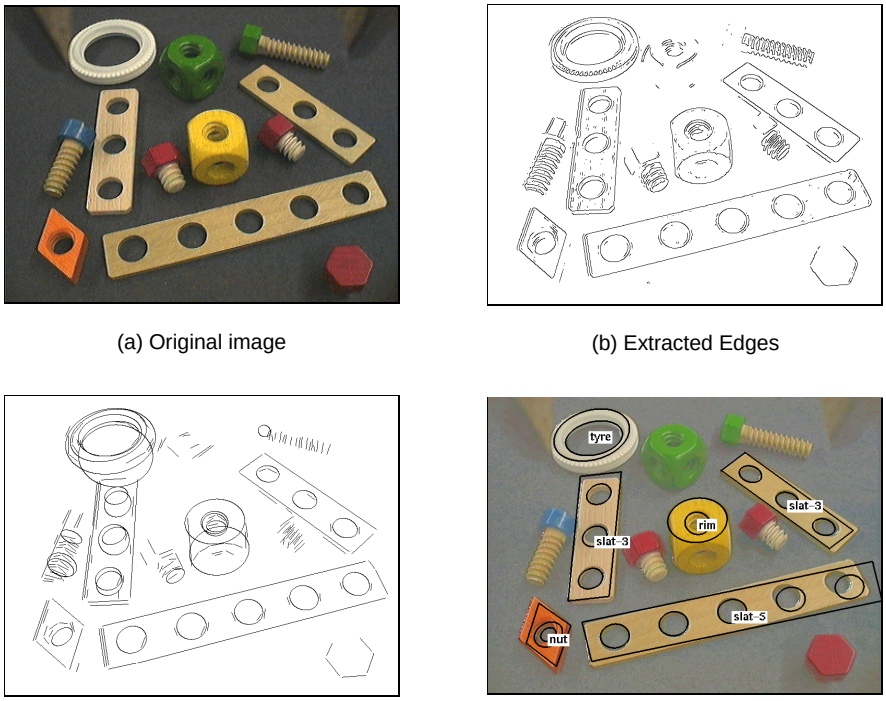
(c) Fitted features

(d) Result

Figure 3.12: FII result: unrecognised *nut* and *rim*

Figure 3.12, which has been taken using a hand-camera, shows an example for false negatives. In this example, the recognition system does not manage to detect the *rim* as well as one of the *nuts*. The problem of this deficiency is a consequence of an inaccurate edge extraction (due to low contrast) and feature fitting. As indicated in Fig. 3.12b very few edge points of the *rim* have been extracted, which prevents the fitting of any ellipse, whereas the ellipse, that has been fitted to the *nut*, does not represent the topology of the hole adequately (Fig. 3.12c). Hence, the measured invariant values differ to much from the desired values and no object hypothesis is generated.

In contrast Fig. 3.13 provides an example for a typical recognition problem that has been encountered: In this test scene the FII-recognition system has generated a false positive for a (unknown) Baufix *cube* (Fig. 3.13d). Since the feature extraction stage has fitted two ellipses to the edge points of the cube which are very similar to the pair of ellipses of a *rim* (Fig. 3.13c) and since the verification stage of the system only relies on the extracted geometric primitives, the system cannot detect the erroneously hypothesised *rim* as a false



(a) Original image

(b) Extracted Edges

(c) Fitted features

(d) Result

Figure 3.13: FII result: incorrectly recognised cubes

positive. In other test images, especially in those taken with the top-view camera, the system incorrectly recognises *cubes* as *nuts*. In these cases the feature extraction stage has fitted straight lines and ellipses which are projectively equivalent or similar to the geometric primitives of a *nut*. Thus, the verification stage fails to remove those false positives resulting in incorrectly recognised *nuts*.

Table 3.1 summarises the recognition results for the test scenes that contain only unoccluded known and unknown objects. As can be seen, the FII-recognition system, which avoids further constraints to the camera setup, generally provides good results. Only for the *rims* and *nuts* the recognition rate is below 90% resulting mainly from the aforementioned inaccurate edge extraction and feature fitting. However, it must be noted that many false positives (34%) have been generated. In most of the cases, the system has incorrectly recognised the unknown *cubes* as *rims*, *tyres*, or *nuts*. To overcome this problem two different approaches may be possible: (i) the verification stage of the FII-recognition system can be enhanced to analyse the test scenes in great detail by utilising additional information like colour and texture, or (ii) an additional system capable to recognise the *cubes* reliably can be provided, so that the *cubes* will be recognised correctly. Finally, it should be mentioned, that the best recognition rates are gained for test scenes taken with the top-view camera.

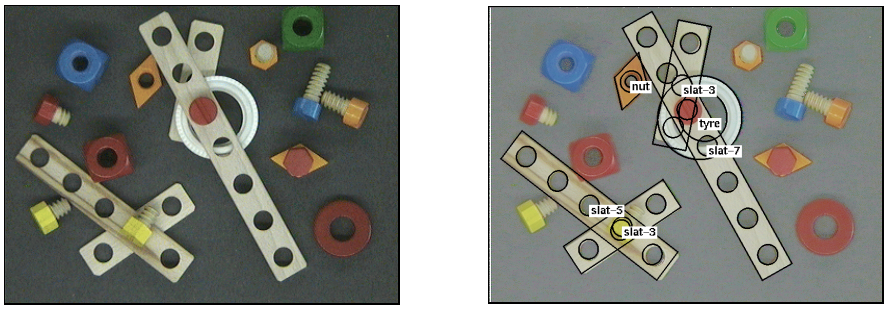
Table 3.1: FII results for unoccluded test scenes

	present	false negatives		false positives		correct	
nut	410	65	(16%)	1	(0%)	345	(84%)
rim	179	52	(29%)	0	(0%)	127	(71%)
slat-3	252	7	(3%)	6	(2%)	240	(95%)
slat-5	162	6	(4%)	9	(6%)	150	(93%)
slat-7	105	0	(0%)	9	(9%)	99	(94%)
tyre	136	2	(1%)	1	(1%)	133	(98%)
cube	(396)			133	(34%)		
screw	(691)			7	(1%)		
$\sum$	1244	132	(10%)	166	(13%)	1094	(88%)

Although the invariant object descriptions are independent of the viewpoint, the feature extraction stage is affected by the particular camera position. For example, when the objects are observed under a low viewing angle, some of the fitted ellipses are squashed because they are fitted to edge points coming from the top as well as from the bottom of an object. Therefore, these ellipses do not represent the topology of the objects and the verification stage of the recognition system may produce false negatives or false positives.

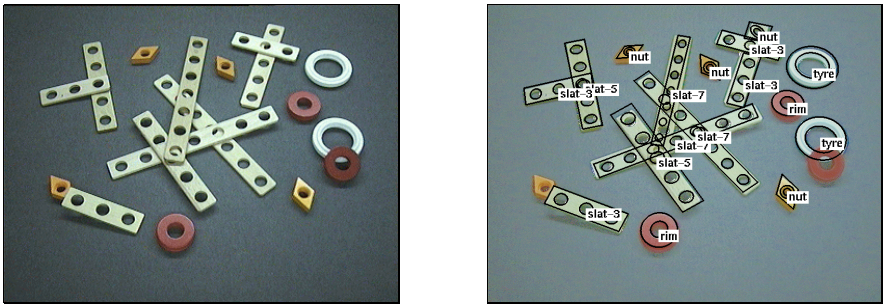
Results for Test Scenes Containing Partially Occluded Objects

This section discusses the much more difficult task of recognising partially occluded objects in test scenes that contain unknown objects as well.



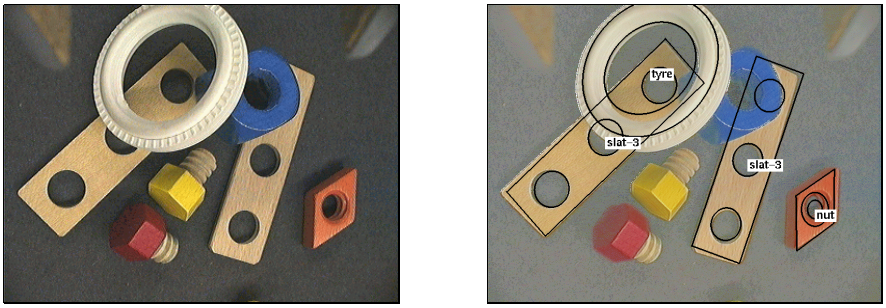
(a) Original image (b) Result

Figure 3.14: Result: partially occluded objects taken with a top-view camera



(a) Original image (b) Result

Figure 3.15: Result: partially occluded objects taken with a front-view camera

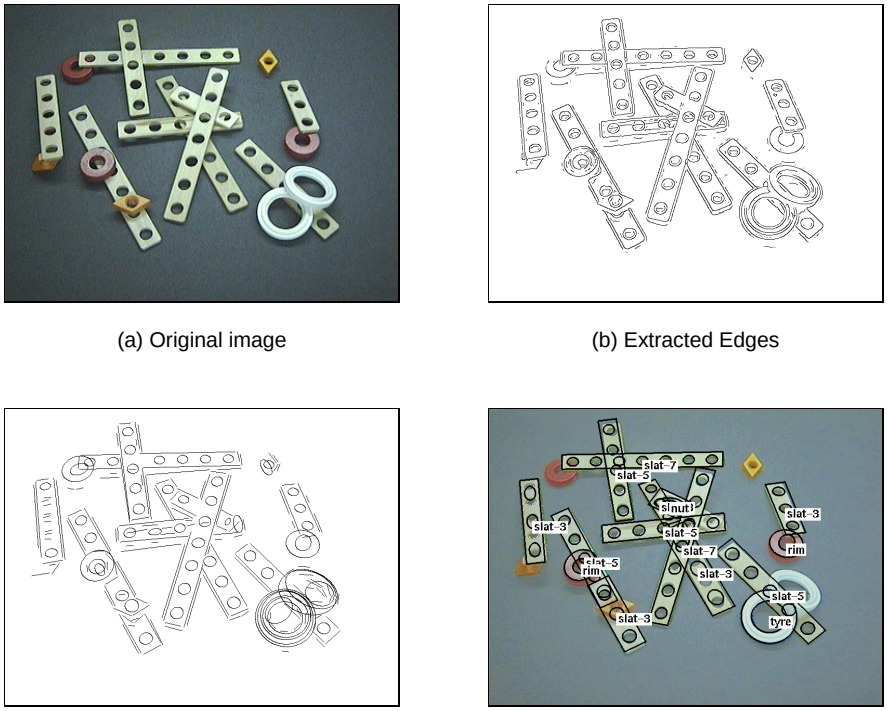


(a) Original image (b) Result

Figure 3.16: Result: partially occluded objects taken with a hand-camera

Nevertheless, as demonstrated in Figs. 3.14 – 3.16, which have been taken using the top-view (Fig. 3.14), the front-view (Fig. 3.15), and a hand-camera (Fig. 3.16), the system has still the capability to provide very good recognition results and to describe the topology of the scenes adequately.

However, in many situations the recognition system detects incorrect *slat* types or multiple *slat* instances for a single one, in order to attempt to explain the features observed in the images. Such a recognition result is demonstrated in Fig. 3.17, which originate from the front-view camera. In this test scene, the system has detected three *3-holed-slats* and a *5-holed-slat* instead of two *7-holed-slats*. Furthermore a *3-holed-slat* is recognised as a *nut*. In three of these cases, the false positives have been generated using straight lines, which are a result of the occlusion (Fig. 3.17c) and have been misinterpreted as the boundaries of objects.



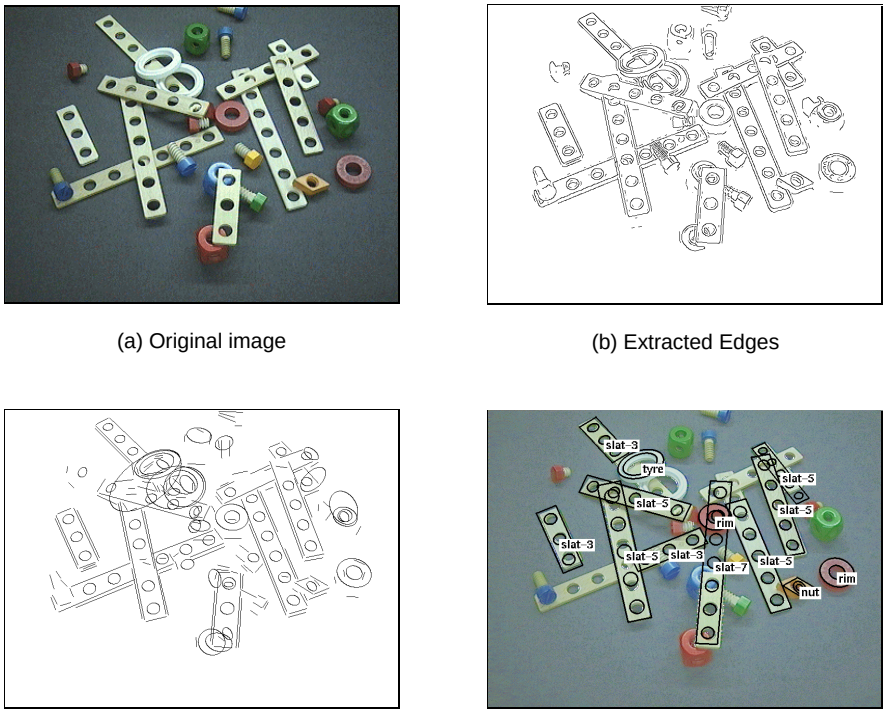
(a) Original image (b) Extracted Edges

(c) Fitted features (d) Result

Figure 3.17: FII result: incorrectly recognised slats I

A similar problem arises in Fig. 3.18. Again, the system does not manage to detect the correct *slat* boundaries, so that many false positives are generated. For example, in the case of the *3-holed-slat* that has been erroneously recognised as a *7-holed-slat* the heuristic of the verification stage employs the ellipses fitted to a *rim* and to an other *slat* as support for the incorrect hypothesis. Note, that although the heuristic fails in this situation, it often leads to the correct hypotheses for many other partially occluded objects.

Table 3.2 summarises the achieved recognition results. As expected, the recognition rates are generally lower than the recognition rates for unoccluded test scenes. On the one hand, the system produces more false negatives, because the system fails to extract the geometric primitives required to recognise the objects, and on the other hand the system generates more false positives. This is true especially for the slats, because the system generally cannot detect all features of a slat required to determine its particular type reliably (see also Sect. 3.5.2 for a more detailed discussion about the occlusion problem).



(a) Original image

(b) Extracted Edges

(c) Fitted features

(d) Result

Figure 3.18: FII result: incorrectly recognised slats II

Table 3.2: FII results for occluded test scenes

	present	false negatives		false positives		correct	
nut	394	134	(34%)	5	(1%)	255	(65%)
rim	202	118	(58%)	1	(0%)	83	(41%)
slat-3	244	47	(19%)	29	(12%)	170	(70%)
slat-5	233	35	(15%)	49	(21%)	155	(67%)
slat-7	207	18	(9%)	50	(24%)	151	(73%)
tyre	142	25	(18%)	1	(0%)	116	(82%)
cube	(539)			50	(26%)		
screw	(1032)			4	(1%)		
$\sum$	1422	377	(27%)	189	(13%)	930	(65%)

3.4.2 Comparison between Crisp and Fuzzy Invariant Indexing

So far the recognition performance of the object recognition system employing the FII-technique has been demonstrated. Now the FII-technique will be compared to conventional hypotheses generation methods based on geometric invariants, like the geometric hashing [LAMDAN and WOLFSON 1988] and the invariant indexing technique [ROTHWELL 1995b].

Since implemented recognition systems based on those techniques are not available, their behaviour will be simulated by the FII-technique using rectangular-shaped membership functions for modelling the fluctuation of invariant values. That means, a classification rule indicates the presence of an object, if the observed invariant values lie within certain intervals. In the following this technique will be called crisp invariant indexing (CII).

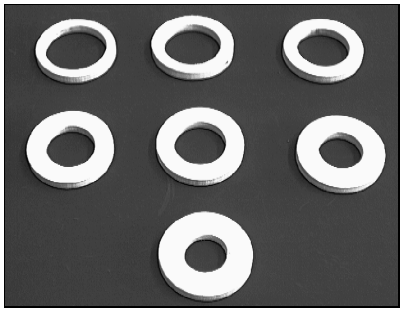
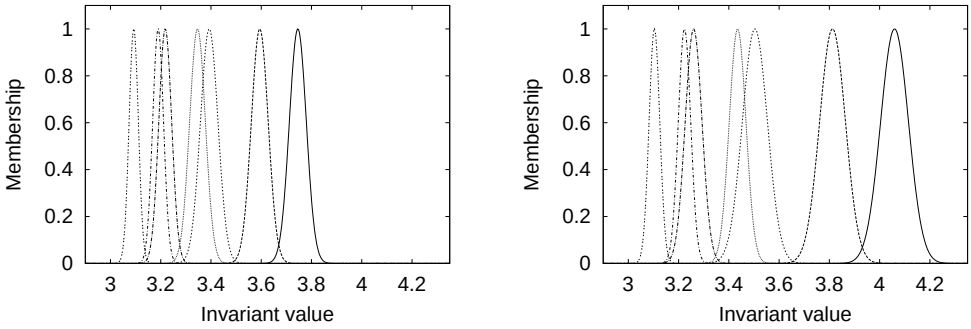


Figure 3.19: Object domain of seven similar disks



(a) Fuzzy invariant values of invariant 1

(b) Fuzzy invariant values of invariant 2

Figure 3.20: Fuzzy invariant values for objects of Fig. 3.19

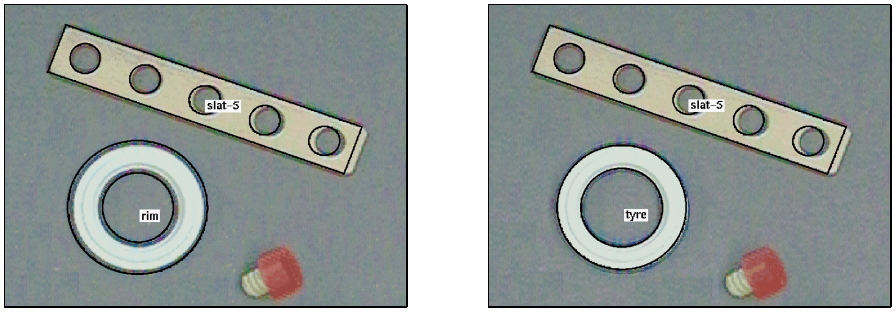
To emphasise the advantages of the proposed FII-technique we apply both methods to the difficult case of recognising very similar objects. Figure 3.19 shows the test object domain of seven similar disks, which share a constant exterior diameter of 50 mm, but varying interior diameters from 22 mm to 38 mm.

The generated fuzzified invariant values are shown in Fig. 3.20, where Fig. 3.20a corresponds to the first invariant (2.5) and Fig. 3.20b to the second invariant (2.6) of a pair of coplanar conics. From left to right the membership functions represent the objects *disk38*, *disk35*, *disk32*, *disk30*, *disk28*, *disk25* and *disk22* respectively, where the numbers denote the interior diameters of the disks. It is apparent that the fuzzy invariant values overlap each other (as well as the corresponding membership functions of the CII-technique). In two cases, for *disk28* and *disk30* as well as for *disk32* and *disk35*, the overlaps are extremely high. Especially for these cases the FII-technique should out-perform the CII-technique, since it provides not only object classes but also the believes for the classes which are used as clues to discriminate between different indexed object hypotheses.

Table 3.3: Comparison between crisp and fuzzy invariant indexing: recognition results

	CII			FII		
	corr.	false	rate	corr.	false	rate
disk22	27	3	90.0%	27	3	90.0%
disk25	17	13	56.7%	23	7	76.7%
disk28	19	11	63.3%	24	6	80.0%
disk30	6	24	20.0%	11	19	36.7%
disk32	18	12	60.0%	19	11	63.3%
disk35	2	28	6.7%	19	11	63.3%
disk38	19	11	63.3%	29	1	96.7%
$\sum$	108	102	51.4%	152	58	75.2%

The comparison is done through recognising the objects in 210 test images (i.e. 30 images per object), which are taken from different viewpoints and at different zooming levels. Table 3.3 summarises the results. As expected the FII-technique provides a better discrimination ability between the objects with the great clashes than the CII-technique; e.g. the recognition rate of the CII-technique for *disk30* is only 20% and for *disk35* as low as 6.7%. The FII-technique improves these rates to 36.7% and 63.3%, respectively. Furthermore, the FII-technique improves the recognition rate for object *disk25* as well. Only for object *disk22* the recognition rate remains unchanged. Altogether, the FII-recognition system possesses a recognition rate of about 75.2% while the CII-recognition system only reaches a rate of 51.4%.



(a) Crisp invariant indexing (b) Fuzzy invariant indexing

Figure 3.21: Comparison between recognition results of crisp and fuzzy invariant indexing

A similar result has been obtained for the *rims* and *tyres* of the Baufix domain. For example, as shown in Fig. 3.21 the CII-technique has erroneously recognised a *rim* instead of the *tyre* (Fig. 3.21a) while the FII-technique has recognised all known objects correctly (Fig. 3.21b). However, this deficiency of the CII-technique arises less frequently than for the aforementioned seven disks because the difference of the extracted geometric structures of the *rims* and *tyres* is in most of the cases sufficient.

3.4.3 Flexibility of FII-Technique

Flexibility is another important property of the FII-technique, which generally cannot be achieved by other hypothesis generation methods based on invariants. Since the FII-technique employs human read- and writable fuzzy classification rules for generating the object hypotheses, the indexing technique can be easily modified and expanded to comply e.g. with changed requirements.

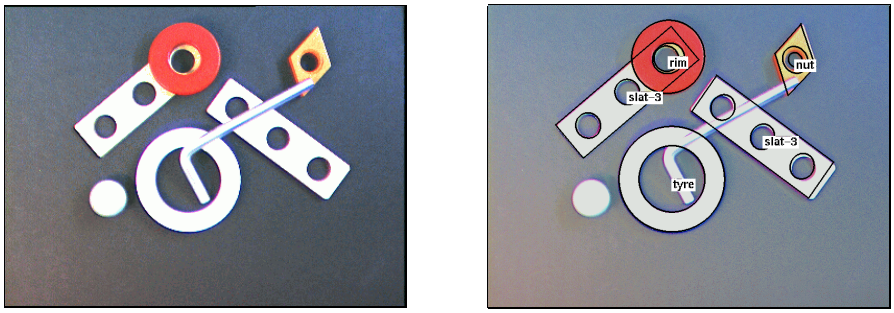
To demonstrate this flexibility the acquired fuzzy classification rules, which have been utilised to gain the recognition results in Sect. 3.4.1, are expanded by integrating an additional colour attribute. This attribute is determined by measuring the colour information of an object and is utilised by applying the following simple, straight-forward approach:

1. Determine the HSV colour histograms of the geometric primitives used to construct the fuzzified invariant object descriptions, where the colour information is measured along the primitives (in the implemented FII-system this is only done for the ellipses).
2. Calculate the mean HSV colour value for each primitive.
3. Depending on the mean saturations of the geometric primitives either utilise mean hues or the mean intensities for generating and evaluating the fuzzy rules.

For example, this approach leads to an expanded fuzzy classification rule for the Baufix *rim* that looks like:

$$\text{IF (inv1} \approx 3.8\text{) AND (inv2} \approx 4.1\text{) AND (hue is RED)}$$
$$\text{THEN (object is RIM)}$$

Experimental results, which have been gained for 30 different test scenes taken with a single static camera under constant illumination conditions, show that this extension reduces the number of generated object hypotheses by about 34%, where mainly false positives are suppressed [GRAF et al. 1998c]; e.g. for Fig. 3.22a the extended fuzzy rules decrease the number of hypotheses from 124 to 81.



(a) Original image (b) Result

Figure 3.22: FII result: employing colour information

Although these recognition results cannot be generalised to arbitrary imaging setups and conditions (see also Sect. 3.5.1), they indicate that the integration of further attributes can be utilised to enhance the performance of a recognition system in two respects: (i) to speed up the recognition process since fewer object hypotheses must be investigated in the time consuming verification stage, and (ii) to increase the robustness of the system, because fewer false positives are established.

3.5 Additional Remarks

This section provides some additional remarks and suggestions concerning the proposed FII-recognition system (Sect. 3.3) and the obtained experimental results (Sect. 3.4). In particular, it discusses the use of colour for recognising objects as well as the occlusion problem in some more detail.

3.5.1 Colour in Object Recognition

In the computer vision community it is a well-known problem to extract and classify the colours of objects robustly and reliably. Especially, the achievement of colour constancy is a difficult and challenging task. The term *color constancy* refers to the ability of humans to perceive colours very stable in real world scenes under various environmental conditions [LUONG 1993].

In general, the colour information measured by a visual sensor is affected by several factors which are concerned with the particular light sources, the type of objects present in the scene, as well as the sensor itself:

- *Light sources:*
  Light sources have a great impact on the colour information measured in an image. Often a slight change in illumination entails a drastic variation of the observed colour. Additionally, the same object produces different colour information when it is observed at different locations because the illumination conditions provided by the light sources are not spatial constant in colour and intensity.

- *Camera:*
  Since cameras differ in their sensitivity to colour they generally produce different images for the same scene. Moreover, every camera image is affected by a noisy imaging hardware at the expense of accuracy of the obtained colour information.

- *Objects:*
  Even if objects share the same colour, they may appear completely different. For example highlights and specularities visible on the surfaces of an object are influenced by the location and orientation relative to the camera, as well as by the shape and the physical structure of the surfaces.

- *Scene:*
  Finally, the scene, i.e. the relationship among objects, impairs the colour observation as an additional source. Since objects produce shadows and transmit/reflect particular portions of light, the objects in a scene affect each other.

Despite considerable effort that has been spent to study colour constancy, the developed algorithms and methods, see the recently proposed methods [FORSYTH 1990] and [FINLAYSON 1996], are still affected by many of the problems mentioned above. Therefore, the developed algorithms impose several conditions to the imaging setup which restrict the practicability of these algorithms to simplified scenes. For example, the assumptions that (i) a scene is always illuminated by indirect diffused light and/or (ii) only few colours are visible in a scene, are generally not fulfilled in real world applications.

As a result the developed algorithms for classifying colours cannot be applied robustly and reliably in the camera setups under consideration (see Fig. 1.1): These setups may be composed of several cameras located at different positions, so that the supposed imaging

conditions cannot be guaranteed. Especially, dynamic hand-cameras, which can be moved to arbitrary positions, may take pictures that show highlights and specularities on the surfaces of objects making it impossible to determine colour information correctly. Therefore, the recognition results presented in Sect. 3.4.3, which have been obtained using a single static camera, must be interpreted with some care.

In general, colour should not be used in such setups as a single measure for classifying and verifying the object hypotheses within a recognition system, but rather as an additional clue for modifying the beliefs of generated hypotheses.

3.5.2 The Occlusion Problem

Although the Baufix object domain seems to be simple to recognised, some important and interesting problems have been encountered in the experimental results in Sect. 3.4.1. The FII-recognition system sometimes recognises an incorrect slat type. In particular, the system detects (i) a shorter slat type if it is unable to find appropriate image support, and (ii) a longer slat type if it erroneously merges the features of different objects into a single hypothesis. In both cases the false positives are mainly caused due to the difficult task of handling partial occlusions.

In some situations the false positives can be suppressed by employing more sophisticated verification stages, which inspect the images in great details. In order to gain more reliable descriptions of the scene, such verification stages require complex image analysis processes, which do not only rely on the extracted features but also on additional information provided by the images. For example, when the recognition system cannot provide image support for the whole object description of a hypothesis, the verification stage must analyse whether the system has generated a false positive which should be retracted, or the absence of the image support can be explained by partial occlusion (see [ROTHWELL 1995a, ROTHWELL 1996] for such a verification process).

However, even sophisticated verification stages will not be able to solve all of the encountered problems. In contrast to other work that deals with partial occlusion [ROTHWELL 1995b, MUNDY et al. 1994a, MUNDY et al. 1995] the object domain used here contains objects (i.e. the *slats*) which are locally equivalent and can only be classified correctly if the whole object is visible. Thus, in some scenes it is impossible to obtain unambiguous recognition results.

Figure 3.23 shows an example for such a test scene. Even for humans it is a difficult task to decide whether the test scene is composed of two *7-holed-slats* or of four *3-holed-slats*. Only when the highlighted area is examined in some more details, the corner of a *3-holed-slat* becomes apparent. However, a slight modification of the position of the tyre may prevent this observation. Furthermore, vision systems will generally fail to detect such image clues, since it is very error-prone to determine whether such an observation is affected due to noise or it corresponds to an object feature.

Nevertheless, in the setups under consideration a more appropriate approach can be utilised to overcome the aforementioned problems. Since these setups are generally equipped with

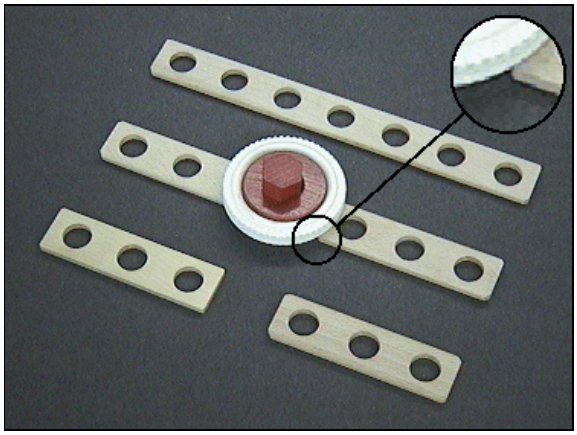


Figure 3.23: Example for an ambiguous test scene

robots, they can be employed to resolve the ambiguities. For example, if the recognition system should recognise a particular type of object, the robot can grasp the object corresponding to the most credible hypothesis and bring it into a pre-defined situation which generally leads to correct recognition results; e.g. by eliminating possible occlusions. If the first recognition result proves to be wrong other object hypothesis can be tested in the same way until a correct object has been found.

4

Object Recognition Using Pattern Invariants

This chapter concentrates on the development of an object recognition method that utilises pattern invariants in order to distinguish between different objects. This method is based on a classification scheme that combines the advantages of employing invariant object descriptions with the classification and approximation ability provided by principle component analysis. The chapter comprehensively explains the general framework of the classification scheme and presents an object recognition system that incorporates the developed method. Furthermore, many experimental results are demonstrated and discussed in detail.

4.1 Motivation

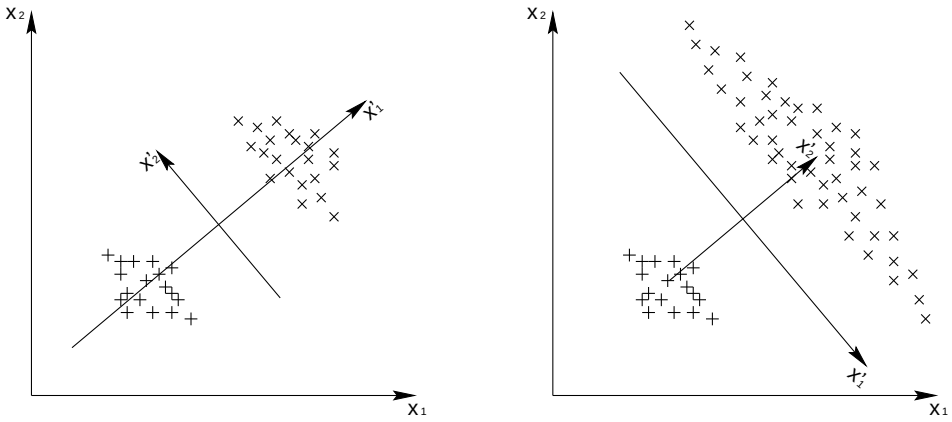
As shown in the previous chapter, the proposed fuzzy invariant indexing technique can be utilised to build flexible object recognition systems that provide the ability to recognise partially occluded objects without making any assumptions about the intrinsic and extrinsic camera parameters. Nevertheless, some deficiencies have been encountered, e.g. geometric invariants require the possibility to extract planar geometric structures that meet the counting argument, and hence, such invariants cannot be applied to arbitrary object domains (see also Sect. 3.4.1).

This chapter investigates a second object recognition method based on pattern invariants which are measured for segmented image patches and take the whole image patch information into account. Since this method relies on different image information and works in

a completely different way, it also differs in some important properties. For example, as opposed to the FII-technique this method can be applied to recognise arbitrary objects composed of at least one single nearly planar surface, but it is more affected by illumination changes and partial occlusion (see Sect. 2.4 for a more complete discussion).

The developed recognition method combines the advantages of employing invariant object descriptions with the classification and approximation ability provided by principle component analysis. This combination is performed in a beneficial way so that invariant theory and principle component analysis support each other.

Principle component analysis (PCA) is an established mathematical method for extracting the statistical relevant information of a multi-dimensional data set [HOTELLING 1993], i.e. a PCA can be used to reduce information by avoiding redundancy. In particular, principle component analysis computes the eigenvectors of a multi-dimensional data set, which forms an orthogonal basis for representing the data. An important property of these eigenvectors is: the eigenvectors which correspond to the largest eigenvalues, called *principle components* (PC), point into the directions of the greatest variances in the data set. Therefore, an information reduction can be obtained by transforming the data into an *eigenspace* which is spanned by a pre-defined number of principle components. In general, few principle components are sufficient to approximate the characteristics of a data set adequately.



(a) First PC permits classification (b) First PC prevents classification

Figure 4.1: PCA for sample points corresponding to two different classes

The principle component analysis has been widely used in appearance-based object recognition [MURASE and NAYAR 1995, MUNDY et al. 1996, PENTLAND et al. 1994]. Since PCA can be utilised to reduce information, it also facilitates the classification process of image patterns, i.e. fewer information must be investigated. The application of PCA for pattern classification is based on the assumption, that the greatest variance in the image patterns can be observed for patterns that correspond to different object classes, so that few principle components are required to classify objects correctly. An example for such a data set

is shown in Fig. 4.1a. It is apparent that the first principle component $\mathbf{x}'_1$ is sufficient to distinguish between the two classes.

However, some well-known problems are inherent to the applications of PCA: In general, it cannot be guaranteed, that (i) the images of different object classes are separated in eigenspace, nor that (ii) the greatest variance of an image set corresponds to images of different objects, because the appearance of a single object may completely change when the object is observed from different viewpoints. For example, the data set depicted in Fig. 4.1b cannot be classified correctly by its first principle component. To alleviate the aforementioned problems, different methods have been developed, e.g. [AITCHISON 1986] proposes the use of logarithmic transformations to map the data into eigenspace, while [HEIDEMANN 1998] proposes a method that first clusters the data set and then applies a PCA to each cluster individually.

In this thesis a different approach is investigated: Since invariants reduce the effects of camera transformations, the PCA is not applied to segmented image patches directly but rather to their corresponding multi-dimensional pattern invariants. As a result, it can be expected, that the obtained eigenspaces provide a better discrimination ability between different objects, because the pattern invariants of objects should result in more compact clusters of the image set.

In the following sections the general framework of the so-called principle invariant component analysis (PICA) is presented in great detail. Furthermore, this method is utilised to build an object recognition system and experimental results are demonstrated.

4.2 Principle Invariant Component Analysis (PICA)

The *principle invariant component analysis* (PICA) provides a classification scheme that integrates pattern invariants and principle component analysis. While the pattern invariants are used to diminish the effects of camera transformations of 2D image patterns, the determined principle components are mainly employed to perform an information reduction in order to facilitate the classification process.

4.2.1 Invariant Pattern Representation

Since principle invariant component analysis does not make any assumptions about the employed pattern invariants, various types of invariants can be utilised. In this thesis a pattern invariant is used which is based on Fourier transformations. In contrast to other pattern invariants, like the ones based on moments, Fourier invariants (i) can be efficiently computed using the *Fast Fourier Transform* [FRIGO and JOHNSON 1999], and (ii) are robust with respect to noise [WOOD 1996] (see also Sect. 2.3.2).

The calculation of this invariant follows the method proposed in [CASASENT and PSALTIS 1976] but integrates an additional normalisation in pattern size and brightness. In particular, the Fourier invariants of image patches are determined as follows:

First, scale invariance is obtained by a re-sampling process of the image patches which have been segmented in an input image, i.e. each image patch is mapped onto a $(N \times N)$-image array. Further, in order to reduce some of the illumination effects, these re-sampled image patches are converted into grey scale images, which are afterwards normalised with respect to brightness by performing a linear scaling of the pixel values, so that:

$$\begin{aligned} \min_{x,y} \{p_k(x,y)\} &= v_0 \\ \max_{x,y} \{p_k(x,y)\} &= v_1 \end{aligned}$$

where $p_k(.)$ denotes the resulting k-th image patch and v_0, v_1 are pre-defined intensity values.

Next, translation invariance is achieved, by applying the two-dimensional Fourier transformation:

$$\mathcal{F}_{T,k}(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p_k(x,y)\, e^{-2\pi i(xu+yv)}\, dx\, dy$$

The power spectra $|\mathcal{F}_{T,k}(u,v)|$ of these Fourier transforms are invariant according to shifts of the image patterns $p_k(.)$. This invariance can be easily verified. Suppose, that $\mathcal{F}'_T(u,v)$ is the Fourier transform of $p'(x,y) = p(x-x_0, y-y_0)$, then:

$$\begin{aligned} \mathcal{F}'_T(u,v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x-x_0, y-y_0)\, e^{-2\pi i(xu+yv)}\, dx\, dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} p(x,y)\, e^{-2\pi i[(x+x_0)u+(y+y_0)v]}\, dx\, dy \\ &= \mathcal{F}_T(u,v)\, e^{-2\pi i(x_0u+y_0v)} \end{aligned}$$

Hence, the power spectra are the same, $|\mathcal{F}'_T(u,v)| = |\mathcal{F}_T(u,v)|$.

In the following step the translation invariant power spectra of the image patterns are transformed into log-polar coordinates $|\mathcal{F}_{T,k}(r,\varphi)|$ by employing the transformation $u = e^r \cos\varphi$, $v = e^r \sin\varphi$. Then, another one-dimensional Fourier transformation of the angle φ is applied:

$$\mathcal{F}_{R,k}(r,\phi) = \int_0^{2\pi} |\mathcal{F}_{T,k}(r,\varphi)|\; e^{-2\pi i\phi\varphi}\, d\varphi$$

Since the power spectra $|\mathcal{F}_{R,k}(r,\phi)|$ of these Fourier transformations are invariant with respect to shifts in the angle φ, rotation invariance is achieved. In contrast to the invariant calculation method proposed in [CASASENT and PSALTIS 1976], the log-transformation of $|\mathcal{F}_{T,k}(u,v)|$ is not employed to obtain scale invariance (which has been already achieved),

but rather to satisfy that the location of the most significant information of a Fourier transform is in the proximity of the origin.

In general, Fourier invariants are used in classification schemes that compares the measured pattern invariants with stored model descriptions by utilising Euclidean distances like least square distance measures [DAVOLI et al. 1999]. However, it must be noted that such approaches lead to sub-optimal results: As proven in [DAVOLI and TAMBURINI 1993], the power spectrum of a Fourier transform of a $(N \times N)$-sampled real non-negative bi-dimensional function $f : [0, 2\pi[\times [0, 2\pi[$ is monotonically decreasing within the interval $[0, \sqrt{2}/8]$ measured in radial direction. This means, that the pixels of a Fourier invariant are not equally weighted in frequency domain, so that Euclidean distances cannot be used reliably.

Therefore, the pattern invariants $|\mathcal{F}_{R,k}(r, \phi)|$ are finally projected back into the spatial domain to yield translation, rotation and scale invariant images. However, it must be noted that invariance is only achieved for images patches that show 2D structures. If 3D structures of objects become visible, the pattern invariants will be generally fluctuating.

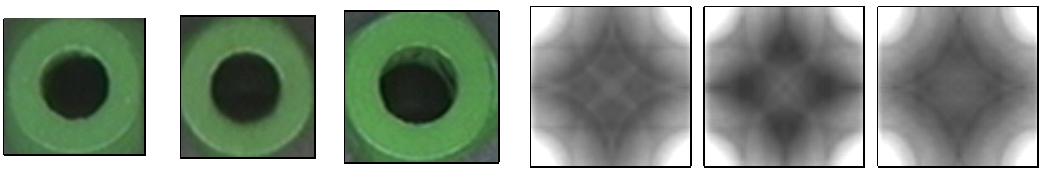
Figure 4.2 demonstrates some examples of pattern invariants obtained for different objects. The left-hand side shows the segmented image patches of the objects, while the right-hand side shows their corresponding invariant patterns. Although the viewpoint between the image patches has been changed, the main differences in the pattern invariants can be observed for invariants measured for different objects.

4.2.2 Principle Component Analysis

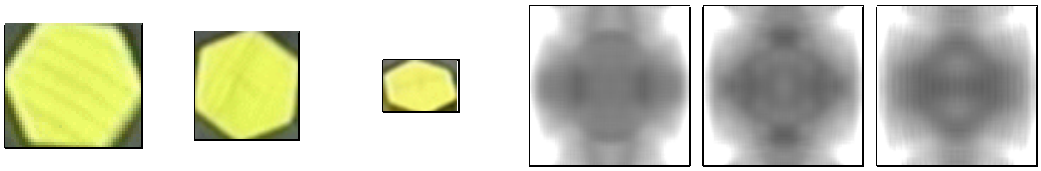
As mentioned before, the *principle component analysis* (PCA), which is also known as Karhunen-Loéve transformation, is a mathematical method for reducing the information provided by a high-dimensional data set. The information reduction is accomplished in a way, that makes the application of PCA for computer vision tasks very attractive, i.e. the PCA preserves the main characteristics of data sets and avoids redundancy. This property is achieved by constructing a low-dimensional eigenspace, of which the coordinate axes, the *principle components* (PCs), point into the directions of the greatest variances in the data set.

Usually, the PCA is employed in the acquisition phase of an object recognition system to determine the transformation that projects the training images of various objects into low-dimensional representations in eigenspace. Then, the recognition process is performed by (i) mapping the segmented image patches of an input image into the eigenspace, and (ii) comparing them with the representations of the training data.

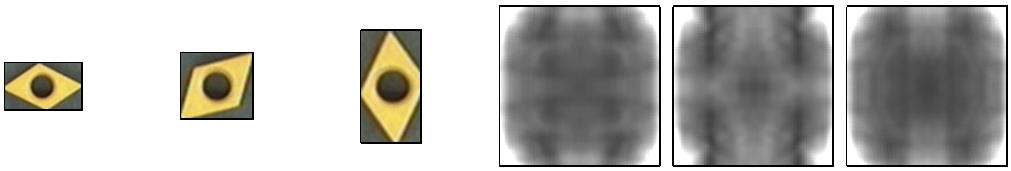
Similar, the PCA is used in this thesis to determine the eigenspace for pattern invariants measured for a set of training images. Although the proposed Fourier invariants fluctuate due to visible 3D structures, illumination changes, discretisation, and quantisation effects (see Fig. 4.2), it can be expected that the greatest differences between invariant patterns can be generally observed for different objects. Therefore, few principle components should be sufficient to distinguish between different objects correctly.



(a) Segmented image patches of a *cube* and the corresponding invariants



(b) Segmented image patches of a *screw* and the corresponding invariants



(c) Segmented image patches of a *nut* and the corresponding invariants

Figure 4.2: Examples for Fourier invariants obtained by employing the proposed method

In particular, the PCA is performed in a common way:

1. Convert the M measured $(N \times N)$-dimensional pattern invariants of a set of training images into N^2-dimensional invariant vectors $\mathbf{i_k},\ 1 \leq k \leq M$.
2. Determine the mean of the invariant vectors $\boldsymbol{\mu} = \sum_{k=1}^{M} \mathbf{i_k}$ as well as the covariance matrix $\mathbf{C} = \frac{1}{M} \sum_{k=1}^{M} (\mathbf{i_k} - \boldsymbol{\mu})(\mathbf{i_k} - \boldsymbol{\mu})^t$.
3. Determine the N^2 eigenvectors $\mathbf{e_k}$ and eigenvalues λ_k of the covariance matrix $\mathbf{M}$.
4. Construct the transformation matrix $\mathbf{T} = [\mathbf{e_1}\ \mathbf{e_2} \ldots \mathbf{e_s}]$ of the eigenvectors $\mathbf{e_k}$ that correspond to the $S\ (\ll N^2)$ largest eigenvalues $\lambda_k, 1 \leq k \leq S$.
5. Use the transformation matrix $\mathbf{T}$ and the mean vector $\boldsymbol{\mu}$ to project the invariant vectors $\mathbf{i_k}$ into the S-dimensional eigenspace: $\mathbf{i'_k} = \mathbf{T}\ (\mathbf{i_k} - \boldsymbol{\mu}),\ \ 1 \leq k \leq M$.

The invariant patterns $\mathbf{i'_k}$ are the result of this process represented in the eigenspace.

An important parameter of the PCA is the dimension S of the eigenspace, because S determines the degree of information reduction as well as the loss of information. In general,

the choice for a suitable S depends on the particular data set. Since the eigenvalue λ_k is a measure for the information content provided by the corresponding eigenvector $\mathbf{e_k}$, the ratio:

$$\frac{\sum_{k=1}^{S} \lambda_k}{\sum_{k=1}^{N^2} \lambda_k} \tag{4.1}$$

determines the degree of information preserved by the eigenspace. For example, if the eigenspace should preserve at least 90% of the information of a data set, the above ratio must be greater than 0.9.

For additional information about principle component analysis, see [MURASE and NAYAR 1995].

4.3 PICA-Recognition System

After having described the general framework as well as the underlying idea of principle invariant component analysis, it will now be utilised to build a flexible object recognition system that provides the ability to recognise objects in images taken with different cameras from a wide variety of viewpoints.

Although the principle invariant component analysis relies on pattern invariants which differ in many respects to geometric invariants, the structure of the developed PICA-recognition system is very similar to the structure of the FII-system (see Sect. 3.3), i.e. the PICA-system is composed of a pre-processing and segmentation stage, an invariant calculation stage, an acquisition stage, the hypothesis generation, and a verification stage.

Nevertheless, the implementations of both systems are completely different. In the following, the modules of the PICA-system will be described in more detail:

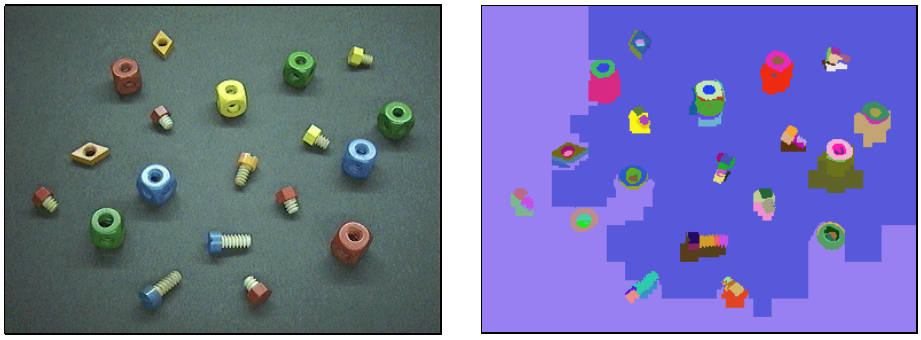
- *Pre-processing and Segmentation:*
  The first stage of the PICA-recognition system is the pre-processing and segmentation stage. In this stage input images are (i) smoothed in order to remove noise, and (ii) segmented into image patches which are used in the following stages of the recognition system to generate appropriate object hypotheses. Thus, this stage influences the recognition performance of the whole system.

  In particular, the image smoothing is accomplished by employing a 2D Gaussian smoothing filter and the segmentation process is based on a split-and-merge algorithm proposed in [DENZLER et al. 1995], that has been extended to be applied to images of arbitrary size. This split-and-merge algorithm is composed of the following steps:

  First, the input image is subdivided into an initial partition of square image regions. Then, a splitting process is accomplished: If the variance of the pixel values of an image region exceeds a pre-defined threshold, the region is splitted into four smaller square regions. This process is repeated until the variances of all square regions fall below the threshold or the regions have reached a minimum size.

Since the splitting process often cuts topologically connected regions into many small parts, an additional merging process is performed. This process merges two neighbouring image regions if the distance of the mean pixel values of the regions are below a second threshold.

In general, split-and-merge algorithms lead to reliable segmentation results, if they are applied to images composed of objects which are homogeneous in structure and colour, such as the objects of the Baufix domain (Appendix B). Furthermore, split-and-merge algorithms only rely on relative pixel (colour) information and do not require to specify e.g. the number of different image regions *a priori*.



(a) Original image (b) Segmented image

Figure 4.3: Segmented image using the split-and-merge algorithm

An example, for a segmented image which has been generated using the described split-and-merge algorithm is demonstrated in Fig. 4.3. Figure 4.3a shows the input image and Fig. 4.3b the resulting image regions, which have been marked using random colours.

- *Calculation of Invariants:*
  The calculation of the pattern invariants proposed in Sect. 4.2.1 is straightforward: At first, appropriate image patches based on the segmented image regions of the last processing stage are generated. This is done by cutting out the image patches which are determined by the bounding boxes of the extracted image regions. Next, these image patches are sub-sampled in order to obtain image patches of a pre-defined size (in the implemented system (32×32)-patches are used) and then, they are normalised w.r.t. brightness. Afterwards, the translation invariant power spectra are calculated by utilising the 2D Fast Fourier Transformation provided by the FFTW-library [FRIGO and JOHNSON 1999]. Then, these power spectra are expressed in log-polar coordinates and a second 1D Fast Fourier Transformation is applied to achieve additional rotation invariance. Finally, the pattern invariants are back-projected into the spatial domain by employing inverse Fourier transformations.

- *Model Acquisition:*
  Similar to the FII-recognition system, the PICA-system provides an acquisition stage which can be utilised to facilitate the model acquisition. However, this acquisition process requires the assistance of human, because the system cannot decide which of the extracted regions of the training images correspond to meaningful information. Nevertheless, after this pre-selection the acquisition process automatically proceeds as follows:

  The pattern invariants of the selected image patches of all objects are used to generate a universal eigenspace by applying the PCA as described in Sect. 4.2.2. The transformation, which maps the pattern invariants into the universal eigenspace, and the eigen-representations of the pattern invariants, are stored in a model base, which is used in the hypothesis generation stage to generate appropriate object hypotheses.

  Additionally, for each object the acquisition process generates an individual eigenspace, which is used for verification. In contrast to the universal eigenspace, the individual ones are constructed by employing the selected image patterns of single objects directly (instead of their corresponding invariants). Again, the transformation as well as the object representations are stored in the model base.

- *Hypothesis Generation:*
  In the hypothesis generation stage the pattern invariants which have been obtained in the invariant calculation stage are mapped into appropriate object hypotheses. This generation method is based on a nearest-neighbourhood classificator and proceeds in the following way:

  First, the pattern invariants which are likely to correspond to image patches that contain meaningless information and often lead to false positives are left out of consideration, like the invariants of very small or very dark image patches. Then, the remaining pattern invariants are projected into the universal eigenspace and a nearest neighbourhood classificator is used to classify the patterns. In particular, if the minimum Euclidean distance between a pattern invariant observed in the input image and an eigen-representation of an object is below a pre-defined threshold, the image patch is classified as the corresponding object class and an appropriate hypothesis is generated. In general, the hypotheses are composed of the object name, the image region used for the invariant calculation, as well as the minimum distance to the object representations. Note, that a single image patch may result in different hypotheses if the minimum distances of several object classes are below the threshold.

  For example, the 99 segmented image regions of Fig. 4.3b are used to generate 35 different hypotheses.

- *Verification:*
  In the verification stage of the recognition system the object hypotheses are analysed in more detail. Similar to the FII-recognition system, the verification stage is mainly present to suppress incorrectly generated object hypotheses.

  In the implemented system the verification process is accomplished in two steps: First,

the initial hypotheses of the last stage are checked individually and then, the succeeded hypotheses are verified against each other.

In particular, the (non-invariant) image patches of the initial hypotheses are directly projected into the individual eigenspaces of the corresponding objects. Then, a nearest-neighbourhood classificator determines the number of object representations that lie within a pre-defined proximity of the projected image patches, i.e. the Euclidean distances must be below a pre-determined threshold. If no or very few object representations have been found for a hypothesis, it is left out of consideration. Furthermore, the object hypotheses, which have been successfully analysed so far, are compared with one another. If two hypotheses share the same image region or the corresponding bounding boxes heavily overlap one another, the object hypothesis for which fewer representations have been found is suppressed. This verification process finally leads to the accepted object hypotheses.

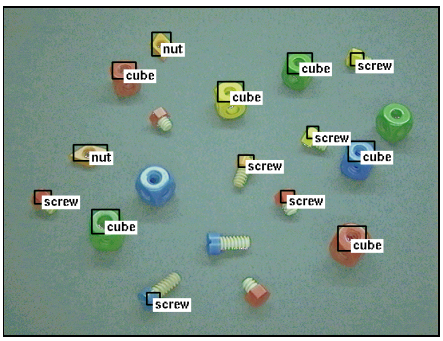


Figure 4.4: PICA result for test image Fig. 4.3a

An example for a recognition result is shown in Fig. 4.4. As can be seen, the system is able to recognise most of the objects correctly.

4.4 Experimental Results

This section discusses the experimental results that have been obtained by employing the proposed PICA-recognition method. First, the section investigates some of the properties and characteristics of the proposed method in more detail and then, it demonstrates the recognition performance of the implemented system.

4.4.1 Properties of the PICA-Recognition Method

As mentioned in Sect. 4.1, it can be expected that the proposed PICA-recognition method leads to a better discrimination ability between different objects than applying the PCA directly. Since pattern invariants remain stable under the imaging processes of cameras, the

use of invariants should result in more complex clusters of the image set and the main differences should be observed for those invariants that have been measured for different objects.

In order to investigate this property of the PICA-recognition method in more detail, it has been used to construct the universal eigenspace for a subset of the Baufix domain (Appendix B) which contains the *cubes*, *nuts*, and *screws*. In particular, the PICA-recognition method is applied to a set of 2032 training images (838 for the different coloured *cubes*, 238 for the *nuts*, and 957 for the various types of *screws*, which differ in their colours as well as in the shape of their screw heads). Since eigenspaces are usually generated by using up to 40 principle components, the investigation is very difficult.

As a first clue for the discrimination ability provided by the PICA-recognition method, the intra- and inter-class distances have been determined. These distances are given by the following equations:

$$D_{\text{inter}} = \frac{2}{K(K-1)} \sum_{\lambda=1}^{K-1} \sum_{\mu=2}^{K} \frac{1}{N_\lambda N_\mu} \sum_{l=1}^{N_\lambda} \sum_{m=1}^{N_\mu} (\mathbf{i}_{\lambda l} - \mathbf{i}_{\mu m})^t (\mathbf{i}_{\lambda l} - \mathbf{i}_{\mu m}) \tag{4.2}$$

$$D_{\text{intra}} = \frac{1}{K} \sum_{\lambda=1}^{K} \frac{1}{N_\lambda^2} \sum_{l=1}^{N_\lambda} \sum_{m=1}^{N_\lambda} (\mathbf{i}_{\lambda l} - \mathbf{i}_{\lambda m})^t (\mathbf{i}_{\lambda l} - \mathbf{i}_{\lambda m}) \tag{4.3}$$

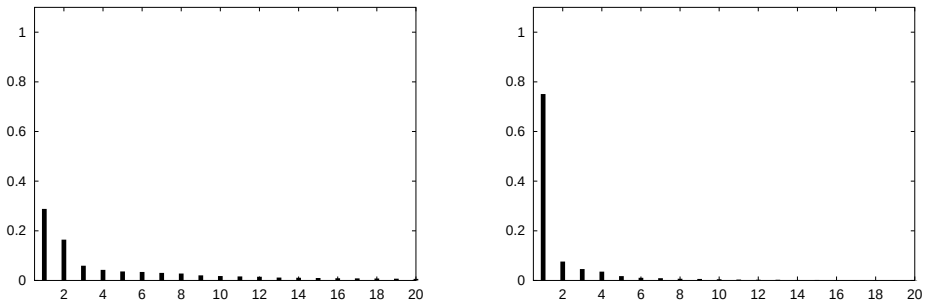
where K denotes the number of different object classes, N_λ is the number of pattern invariants measured for the training images of object λ, and $\mathbf{i}_{\lambda l}$ is the l-th pattern invariant of object λ.

Table 4.1: Comparison between PCA and PICA: class distances

	PCA			PICA		
	intra	inter	ratio	intra	inter	ratio
universal	1788810	2133110	83.9%	1779530	2141500	83.1%
cube	1304770	2010690	64.9%	1304480	2020790	64.6%
nut	465550	1901670	24.5%	441845	1904190	23.2%
screw	2396120	2486970	144.6%	3592250	2499520	143.7%

The distance measures have been applied to the image set represented in the eigenspaces obtained by employing PICA and PCA. To enable a comparison between both methods, each eigenspace is constructed so that it preserves about 98% of the image set information. The results are shown in Tab. 4.1, where the distance measures have also been applied to each object individually (the inter-class distance of a single object is measured with respect

to other objects). As apparent, the ratios of intra- and inter-class distances for the PICA-recognition method are generally lower than the ratios determined for the PCA. Therefore, the clusters of the test images represented in the PICA-eigenspace might be more compact. Unfortunately, the differences of the ratios are not significant.



(a) Information amount of PCs using PCA

(b) Information amount of PCs using PICA

Figure 4.5: Amount of information preserved by each of the first 20 PCs

However, if one compares the amount of information provided by the first principle components, the differences of both methods become more clearly. As indicated in Fig. 4.5, the normalised amount of information of the first n principle components provided by the PICA-recognition method is generally higher than the corresponding information provided by the PCA. For example, in order to preserve 98% of information the PCA requires 190 principle components while the PICA-recognition method leads to 17 components only. Similar, to preserve 85% of information the PCA requires 26 and the PICA-method only 3 principle components. Thus, the clusters in the PICA-eigenspace must be much more compact.

4.4.2 Performance of the PICA-Recognition System

The recognition performance of the PICA-system has been investigated by employing the system to recognise the subset of the Baufix domain described in Sect. 4.4.1, which is composed of *cubes*, *nuts*, and *screws*. This subset is shown in Fig. 4.6.

Although the Fourier invariants can be generally applied for recognising arbitrary types of objects, the *tyre* and the *slats* cannot be handled by the proposed PICA-recognition system. The problem is, that the implemented pre-processing stage simply cuts out image patches which are determined by the bounding boxes of segmented image regions, and therefore, often comprise a large amount of background that might diminish the possibility to identify the correct object classes. However, as shown in Sect. 3.4.1 these objects can be adequately recognised by the geometric-based FII-recognition system.

In the following the PICA-system is applied to the same set of test images that has already been used to obtain the recognition results of the FII-system, i.e. the set is composed of

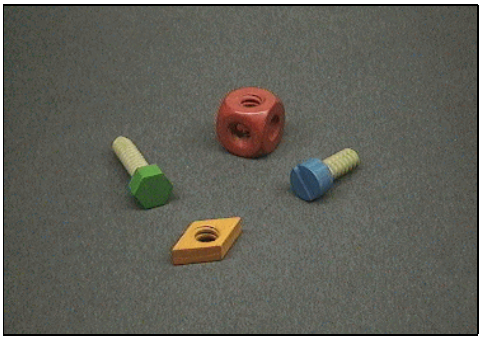
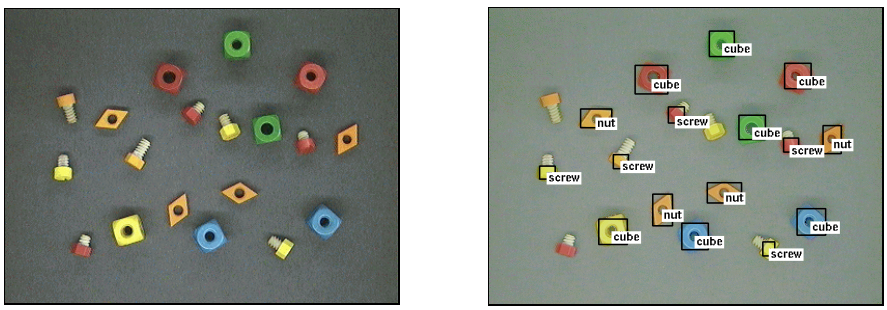


Figure 4.6: Test object domain for the PICA-recognition system

360 test images taken by different cameras from various viewpoints. The objects of the subset of the Baufix domain have been acquired by employing the aforementioned 2032 training images. Note, that this is a relative small number of training images. Many other approaches based on PCA are trained by employing similar numbers of training images, even though the approaches are restricted to single pre-defined static viewpoints, e.g. in [MURASE and NAYAR 1995] 450 images per object have been used.

Results for Unoccluded Objects

At first the PICA-system is applied to test scenes that consist of known and unknown, but unoccluded objects.

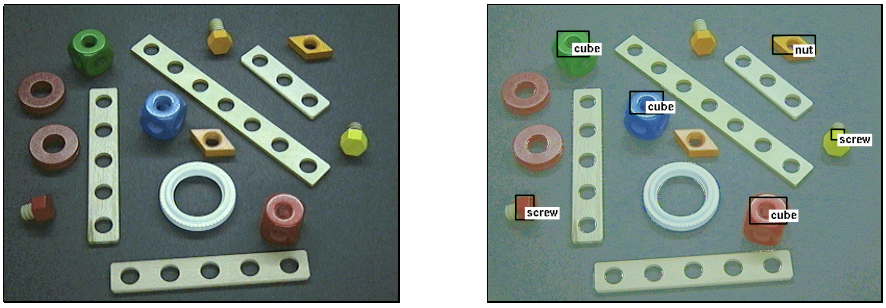


(a) Original image

(b) Result

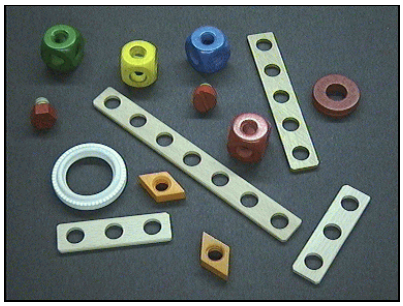
Figure 4.7: PICA result: unoccluded objects taken with a top-view camera

For example, Figs. 4.7 and 4.8 show recognition results for test images that have been taken from different viewpoints using the top-view and the front-view camera. Similar to the FII-system, the PICA-recognition system is able to recognise most of the known objects

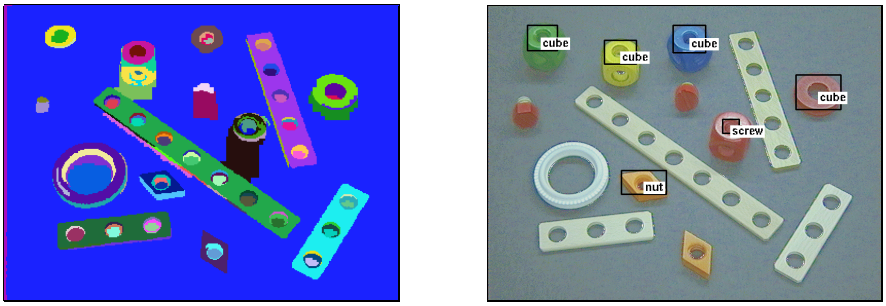


(a) Original image (b) Result

Figure 4.8: PICA result: unoccluded objects taken with a front-view camera



(a) Original image



(b) Segmented regions (c) Result

Figure 4.9: PICA result: incorrectly recognised objects

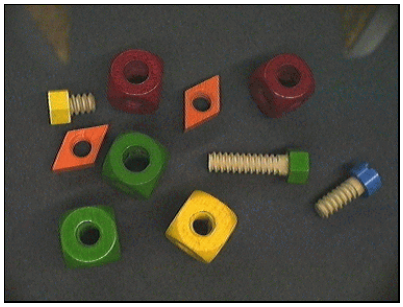
correctly. However, some of the *screws* and one of the *nuts* have not been detected. Such *false negatives* are mainly caused by the following reasons: In some situations the image segmentation process extracts image patches that do not represent the topology of the scene adequately, i.e. the surfaces of objects are splitted into many small regions (see also Fig. 4.9), so that no object hypotheses are generated. This is true, especially for the *screws*. Furthermore, the verification process has been adjusted to suppress as many false positives as possible. In the construction scenarios under consideration, the scenes are usually composed of many object instances of the same type, so that it is generally more suitable to miss some of the objects than to generate false positives. Unfortunately, such restrict verification processes reject correct hypotheses as well.

Figure 4.9 shows an example for *false positives*, where the system has erroneously hypothesised a *screw* instead of the *cube* and a *cube* instead of the (unknown) *rim*. The former false positive arises due to an inaccurate image segmentation. As shown in Fig. 4.9b, the surface of the *cube* has been splitted into different small regions, so that the *cube* cannot be recognised and the incorrect *screw* hypotheses, which corresponds to the threaded hole of the *cube*, is established. The second type of false positive, i.e. the incorrectly recognised *cube*, arises frequently in the test images. Similar to the FII-recognition system, the PICA-system cannot distinguish between the *rim* and the *cube* of the Baufix domain reliably. Even for humans, it is sometimes difficult to distinguish between the two objects, supposed that only the extracted image patches are visible. Although this problem cannot be solved, it will be drastically decreased by the multi-agent vision system proposed in Sect. 6.3.

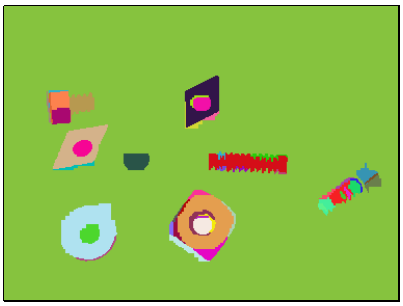
Table 4.2: PICA results for unoccluded scenes

	present	false negatives		false positives		correct	
cube	396	85	(21%)	21	(5%)	301	(76%)
nut	410	108	(26%)	12	(3%)	290	(70%)
screw	691	306	(44%)	27	(4%)	372	(54%)
rim	(179)			123	(68%)		
slat-3	(252)			2	(1%)		
slat-5	(162)			0	(0%)		
slat-7	(105)			2	(2%)		
tyre	(136)			0	(0%)		
$\sum$	1497	499	(33%)	187	(12%)	963	(64%)

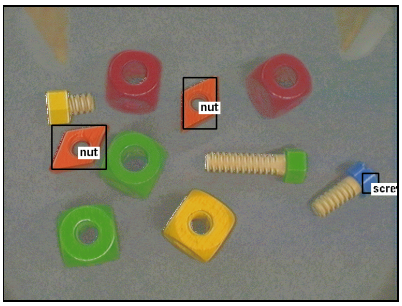
The recognition results obtained for the unoccluded test scenes are summarised in Tab. 4.2. As shown, the recognition rates are generally below the rates provided by the FII-recognition system. Additionally, many false positives are generated for the unknown *rim*. It must be



(a) Original image



(b) Segmented regions



(c) Result

Figure 4.10: PICA result: unrecognised objects in a hand-camera image

Table 4.3: PICA results for unoccluded scenes omitting the hand-camera images

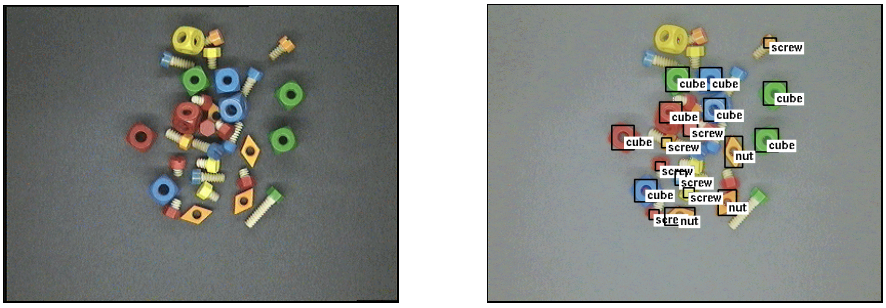
	present	false negatives		false positives		correct	
cube	317	44	(14%)	14	(4%)	269	(85%)
nut	338	70	(21%)	4	(1%)	264	(78%)
screw	544	249	(46%)	8	(1%)	290	(53%)
rim	(150)			114	(76%)		
slat-3	(210)			1	(1%)		
slat-5	(142)			0	(0%)		
slat-7	(97)			1	(1%)		
tyre	(116)			0	(0%)		
$\sum$	1199	363	(30%)	142	(12%)	823	(68%)

mentioned, that the best recognition results are achieved for the test images originated from the top-view and the front-view camera. In the hand-camera images the system often fails to detect many of the objects. This deficiency is mainly caused by inaccurate image segmentations. On the one side the hand-camera images are often over-segmented, i.e. the visible object surfaces are splitted into many small parts, and on the other side object surfaces are not detected (due to low contrast); see also Fig. 4.10.

If one removes the hand-camera images from consideration, the recognition rates are much better (except for the *screws*) and less false positives are generated; see Tab. 4.3.

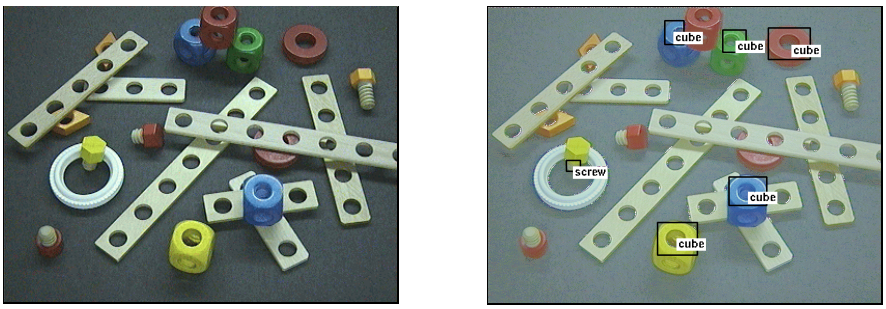
Results for Occluded Objects

In this section the PICA-system is applied to the difficult task of recognising partially occluded objects. Since the system accomplishes the recognition process by utilising only



(a) Original image (b) Result

Figure 4.11: PICA result: partially occluded objects taken with a top-view camera



(a) Original image (b) Result

Figure 4.12: PICA result: partially occluded objects taken with a front-view camera

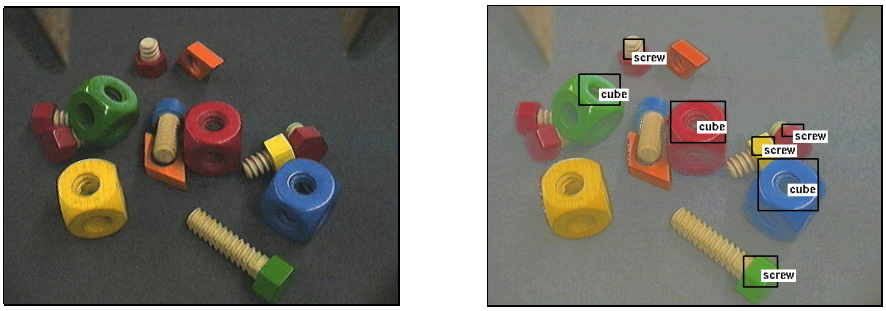
subparts of an object, i.e. its segmented surfaces, the system has the capability to recognise the objects even in presence of occlusion.

For example, Figs. 4.11 – 4.12 show recognition results for test scenes that originate from the top-view (Fig. 4.11) and from the front-view camera (Fig. 4.12). As shown, the system still detects many of the objects correctly.

Table 4.4 summarises the achieved recognition results. As expected, the recognition rates are generally lower than the recognition rates for the unoccluded test scenes. However, in contrast to the FII-recognition system, the PICA-system also generates less false positives,

Table 4.4: PICA results for occluded scenes

	present	false negatives		false positives		correct	
cube	539	175	(32%)	35	(6%)	337	(63%)
nut	394	231	(59%)	29	(7%)	135	(34%)
screw	1033	566	(55%)	27	(3%)	456	(44%)
rim	(202)			64	(32%)		
slat-3	(244)			8	(3%)		
slat-5	(233)			12	(5%)		
slat-7	(207)			5	(2%)		
tyre	(142)			0	(0%)		
$\sum$	1966	972	(49%)	180	(9%)	928	(47%)



(a) Original image

(b) Result

Figure 4.13: PICA result: partially occluded objects taken with a hand-camera

because the partial occlusions prevent the system to produce false positive as well.

Although the system can be applied to hand-camera images, see Fig. 4.13, the system often produces false negatives. Again, if these images are left out of consideration the recognition rates are much better (Tab. 4.5).

Table 4.5: PICA results for occluded scenes omitting the hand-camera images

	present	false negatives		false positives		correct	
cube	454	136	(30%)	16	(4%)	307	(68%)
nut	305	166	(54%)	9	(3%)	130	(43%)
screw	854	489	(57%)	12	(1%)	359	(42%)
rim	(165)			58	(35%)		
slat-3	(190)			1	(1%)		
slat-5	(206)			1	(0%)		
slat-7	(196)			2	(1%)		
tyre	(108)			0	(0%)		
$\sum$	1613	791	(49%)	99	(6%)	796	(49%)

5

Multi-Agent Systems in Computer Vision

This chapter describes several aspects of agent technology which are necessary to build flexible computer vision systems that enhance the capabilities of conventional systems. It introduces some basic concepts of agents and multi-agent systems and discusses the main advantages of employing agent technology in computer vision. Finally, this chapter reviews recent work which focuses on the development of multi-agent vision architectures.

5.1 Motivation

So far this thesis has concentrated on the development of object recognition methods, which have been implemented in recognition systems that follow conventional system architectures. These architectures make it difficult to incorporate the developed systems into modern complex robotic applications.

Generally, conventional vision architectures are too rigid to comply with the requirements of modern robotic applications. Vision tasks that must be handled within such applications are getting more complex and fewer restrictions are imposed to the environmental conditions, so vision systems must be very flexible. For example, (i) it must be simple to incorporate the vision system into complex robotic setups, (ii) the vision system must be able to adapt dynamically to different (possibly changing) tasks and environmental conditions, (iii) the architecture must provide the ability to handle different competitive information, and (iv) the addition of new processing modules must be possible without rebuilding the complete system. Therefore, the development of reliable and flexible computer vision systems intended

to be incorporated into modern complex robotic scenarios must also cover the investigation of appropriate system architectures.

The following chapters will concentrate on the development of multi-agent systems dedicated to solve computer vision tasks. Although the agent technology has been used successfully in many domains, little has been done to investigate multi-agent systems in computer vision. However, since many vision algorithms are developed as special purpose algorithms, it seems to be natural to model a vision system as an assembly of autonomous agents, where each agent represents an expert responsible to solve a particular vision task.

In contrast to conventional vision architectures the vision modules are implemented as agents, which perform their corresponding tasks autonomously avoiding pre-defined information flows. In an agent-based vision system the agents have to communicate in order to solve given tasks. This kind of modelling provides a number of advantages to a vision system, so that it can be adequately integrated into complex scenarios.

Before describing the developed multi-agent vision architecture in great detail in Chap. 6, the following sections will discuss some basic aspects of utilising agent technology in computer vision including some of the main concepts of agent theory, the advantages of agent-based systems, as well as recent work that has been done on agent-based vision architectures.

5.2 Theoretical Background

The field of *Multi-Agent System* (MAS) research is a branch of *Distributed Artificial Intelligence* (DAI) [BOND and GASSER 1988]. The main objective of multi-agent systems is concerned with the coordination of a collection of autonomous agents, which are capable to accomplish diverse tasks according to their own knowledge. Since the agents are independent from each other, the coordination is employed by communication processes.

Therefore, the following sections will provide a short introduction to the agent concept as well as to communication and cooperation models. For a more complete introduction to multi-agent systems see [BOND and GASSER 1988, MÜLLER 1993].

5.2.1 Agent Concept

Although the main concept in multi-agent systems is concerned with the term *agent*, a generally applicable definition of an agent does not exist. Many researchers involved in multi-agent systems have offered a variety of definitions. These definitions range from the simple to the demanding, and are often a result of the investigation of a particular domain. One of the main problems in defining the term agent is that it is still unclear how to distinguish adequately between a software module and an agent.

For example, Russell and Norvig have proposed the AIMA agent definition [RUSSELL and NORVIG 1995]:

> *AIMA Agent:*
> An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

Clearly, this agent specification is very general and depends heavily on the underlying environment as well as on the meaning of sensing and acting. In particular any entity which has a well-defined interface to its environment such as a simple software module, appears to be an agent.

A more suitable and often referred agent definition has been proposed in [WOOLDRIDGE and JENNINGS 1995] , which is also called the *Weak Notion of Agency*:

> *Wooldridge-Jennings Agent:*
> An autonomous agent is a hardware or (more usually) software-based computer system that enjoys the following properties:
>
> – *autonomy:* agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state;
> – *social ability:* agents interact with other agents (and possibly humans) via some kind of communication language;
> – *reactivity:* agents perceive their environment, and respond in a timely fashion to changes that occur in it;
> – *pro-activeness:* agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by taking the initiative.

In addition, Wooldridge and Jennings have suggested a second specification for agent, that extends the definition above by further properties which are usually applied to humans. This agent specification, called *Strong Notion of Agency*, includes mental notions such as knowledge, belief, desire, intention, obligation as well as emotional notions.

A different way of specifying and describing agents has been proposed in [FRANKLIN and GRAESSER 1997]. After having introduced various definitions of agents used in the agent-based computing community, they tried to extract the essence of those definitions to yield a more universal formalisation:

> *Franklin-Graesser Agent:*
> An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future.

As Franklin and Graesser themselves have suggested this definition of an agent is too general to be useful. Therefore, they have proposed the agent taxonomy, shown in Fig. 5.1, as a tool to classify and describe agents.

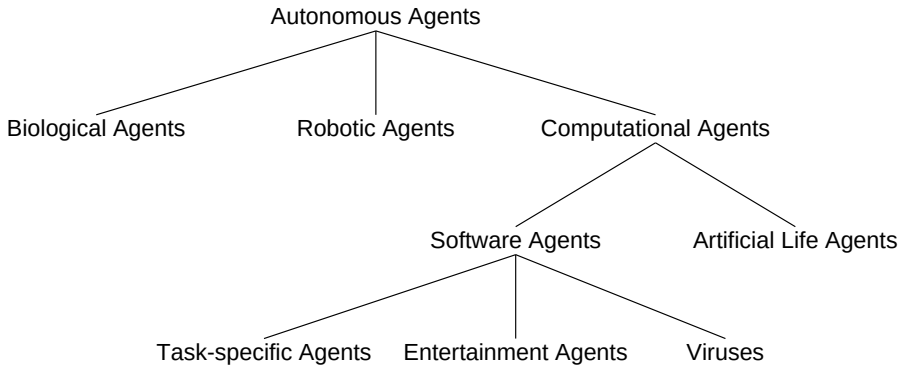


Figure 5.1: Agent taxonomy of Franklin and Graesser

This taxonomy is constructed following the form of the well-known biological taxonomy. At the kingdom level they distinguish between organisms, robotic and computational agents. At the phylum level they sub-classify computational into software and artificial life agents, and finally, at the class level software agents are subdivided into task-specific and entertainment agents as well as into viruses. However, any further classification will be very difficult, since in general it cannot be decided which properties (control structures, environments, applications, etc.) should be naturally used to advance.

5.2.2 Communication and Cooperation

The communication plays an important role in multi-agent systems. In general the agents of a society must communicate in order to achieve their goals. The two main issues for communication being (i) to coordinate the cooperation among agents, and (ii) to exchange knowledge, which might facilitate the handling of tasks.

Several methods have been developed to accomplish the required communication processes. In the context of this thesis two well-known communication methods are described, the blackboard model as well as the popular contract net approach:

- *Blackboard Model:*
  The blackboard model [CORKILL 1991] is a rather simple way to realize the communication and cooperation among the agents of a society. The underlying idea of the blackboard model is that agents communicate via a shared memory (the blackboard), i.e. each agent has the capability to read and write information from/to a specified memory which is shared among all agents. An agent generally will be activated if the information stored in the memory changes into a state which enables the agent to accomplish one of its tasks.

In particular, the communication and cooperation process is performed in the following way: In the beginning the blackboard contains just an initial description of a task. Each agent of the society continuously observes the blackboard and analyses the provided information. If an agent is able to accomplish a described task or to provide additional information, it will be activated. However, in systems where more than one agent might be able to become activated at the same time a control mechanism, which is responsible to select the appropriate agent, must be installed. This selection can be done by simply choosing the first agent that reacts (first come - first served), or by selecting the agent that might do best. Subsequently, the selected agent accomplishes its task and changes the information stored in the shared memory. This process will proceed until the initial task has been solved or no further agent can be activated.

The agents of a blackboard system are completely independent from each other and do not need to have any knowledge about the agent society. Therefore, a blackboard system can be easily expanded by adding new agents. However, all agents must represent information in exactly the same way in order to enable an information exchange via the blackboard. Furthermore, the structure of a blackboard becomes more complex, if the complexity of the tasks that must be handled increases.

- *Contract Net:*
  In contrast to the blackboard model the contract net approach provides a more natural way of communicating. In this approach the agents communicate directly with each other in order to achieve their goals.

  The communication process among the agents follows a particular scheme: First, an agent, which requires the assistance of other agents of the society, generates an appropriate request and sends it to other agents. The requesting agent is often called manager, because it controls many aspects concerning the subsequent communication. It decides whether the request should be directed to some particular agents, i.e. point-to-point or one-to-many communication, or should be sent to the complete agent society, i.e. broadcast communication. The agents, which receive the message, analyse the request and make bids if they are able to fulfil the requested task. According to these bids the manager selects the agents that should award the contract. Then, the contractors accomplish the requested task, which possibly includes further requests to other agents, and generate appropriate answers. Therefore, the roles of managers and contractors are not pre-defined in an agent society and may change during a communication process. For example, a contractor becomes a manager if the contractor must also request the assistance of other agents.

  Furthermore, agents cannot only communicate in order to request the assistance of other agents, but can also utilise communication for various issues such as knowledge exchange. The intention of a message is generally indicated by message types implementing speech acts, such as `request`, `inform`, and `answer`. For additional information about the contract net approach and speech acts see [SMITH 1981] or [WOOLDRIDGE and JENNINGS 1994].

5.3 Advantages of Multi-Agent Systems

The concept of employing agents to model complex systems provides a number of advantages which are mainly concerned with the flexibility of the resulting systems. These advantages make multi-agent system architectures attractive for computer vision, because multi-agent systems can be utilised to out-perform conventional vision architectures.

Although the following advantages are not discussed with respect to computer vision explicity, they are valid for multi-agent vision systems, too.

- *Robustness:*
  Multi-agent architectures can be utilised to model systems which are robust with respect to breakdowns of single processing modules. A number of equivalent agents can be instantiated to provide redundancy, so that the agent society will be able to perform its tasks even if some agents are unavailable. However, in situations where the remaining agents can just accomplish similar tasks, the system may produce sub-optimal results.

- *Efficiency:*
  Generally, distributed system architectures enable parallel computation to enhance the efficiency of implemented systems. In the multi-agent system paradigm parallel computation can be realized in different ways:

  Either a task requiring the same computation on a large set of data can be performed by employing a number of equivalent agents working on different subsets of data, or assuming the task can be divided into different sub-tasks which are independent from each other, then those sub-tasks can be accomplished by different agents at the same time. Additionally, the combination of both cases can also be handled.

- *Adaptability:*
  Since it is possible to build agents which always act according to their own knowledge and goals avoiding a pre-defined information exchange among the agents of a society, an agent society may be able to solve its tasks in different situations in different ways. This leads to systems which can react adequately in unexpected situations and therefore will be much more robust than traditional systems which require well-defined environments and conditions.

- *Openness:*
  Multi-agent architectures can be used to generate open systems, i.e. systems which can be expanded by adding new functionality. This expansion can be done by constructing appropriate agents. It is generally not required to have precise knowledge about the internal structure of the existing agents, because the agents of a society often communicate using an abstract communication language. Moreover, the expansion of a system may be possible at run-time.

- *Integrability:*
  An important property of an agent architecture is the possibility to integrate different

competitive processing modules into a single system. The integration can be realized in several ways and mainly depends on the concrete implementation. The main advantage of integrating different modules for similar tasks is to yield more reliable results. For example, different object recognition methods can be integrated into a single vision system to enhance the recognition performance of the system.

- *Re-usability:*
  Since agents are responsible for particular tasks and since they possess well-defined communication interfaces, agents can be easily re-used. For example an agent implementing some image pre-processing methods can be integrated in any recognition system which relies on those methods.

- *Resource Management:*
  Another property of multi-agent systems is their ability to perform resource management on their own. This property becomes very important if various agents have to share a system resource which can be controlled by only one agent at a time like a camera. A simple way to overcome such a problem is to build an agent whose only responsibility is to manage the access to a particular system resource. Nevertheless, resource management can also be utilised to increase the efficiency of computation. For example, if several agents perform the same task the fastest agents can be selected by measuring e.g. the speed, and computational load of the CPUs of the underlying computer systems.

It must be noted, that a vision system modelled as an assembly of autonomous agents does not achieve all of these properties automatically. The agent technology only provides the theoretical background for building very flexible systems. However, the particular degree of flexibility that is achieved by an agent-based vision system heavily depends on the implementation of the multi-agent system concept.

5.4 Multi-Agent Vision Systems

As already mentioned little has been done for the development of building flexible and reliable computer vision systems suitable to be incorporated into modern complex robotic applications. Especially, multi-agent systems have been rarely employed.

In the following four different multi-agent vision architectures and systems that have been proposed recently are described.

- *MAVI:*
  The MAVI system, a **M**ulti-**A**gent system for **V**isual **I**ntegration, has been proposed by Bossier and Demazeau (see e.g. [BOISSIER and DEMAZEAU 1994b, DEMAZEAU et al. 1994a, DEMAZEAU et al. 1994b]). As indicated by its name, the main goal of this system is to facilitate the integration of various image processing modules into a single computer vision system.

The agents of the MAVI system share a common agent architecture, which is provided by the underlying ASIC [BOISSIER and DEMAZEAU 1994a] multi-agent control architecture. In this architecture an agent is composed of a hierarchy of three different control layers: a decision layer, an adaption layer, and a command layer. The three layers determine the behaviour of the agent and specify how the agents react according to a given request. In particular, the decision layer specifies the goals to be satisfied, the adaption layer chooses a particular plan, and the command layer specifies the actions according to the given goals and plan. In order to enable each layer to accomplish its tasks within the architecture, each layer is provided with an inference engine, a knowledge base, and different processing states for storing various information.

The agents of a society are connected to each other in a completely connected network, i.e. each agent can interact with every agent it wants to interact with. In order to make the system converge to a solution, the agents possess a social control model which defines many aspects with respect to the message passing such as the type of provided information, the particular time a message has to be sent, and the tasks that must be accomplished by each agent. The communication among agents is performed by employing a complex interaction language, where a message specified in this language covers information about (i) the speech act of the message, i.e. request, answer or inform, (ii) the control layer the message belongs to, i.e. every message is dedicated to a particular control layer of the addressed agent, (iii) the priority of the message, and (iv) the content of a message, which depends on the application domain.

Based on this multi-agent architecture, Boissier and Demazeau have implemented a computer vision system [BOISSIER and DEMAZEAU 1994b], which is capable of performing vision tasks like the extraction of regions of interest in a given input image. The implementation makes some disadvantages of the approach apparent: the agent architecture is too complex to model such a vision system adequately, since it seems that most of the effort must be spent to implement the agent architecture. Furthermore, the communication among agents is based on a complex interaction language. This language tends to be cryptic and makes it difficult to handle the system.

- *Purposive computer vision system:*
  Bianchi and Rillo have proposed a so-called purposive computer vision system (see [BIANCHI and RILLO 1996, BIANCHI and RILLO 1997]). The basic idea of their approach is to consider vision always within the sets of particular tasks that must be accomplished in a robotic application, where the purpose of a system is decomposed into a set of behaviours and behaviours are translated into specific tasks, i.e. the purpose of a system is modelled by the society of autonomous agents, where each agent is responsible for a visually guided behaviour, and tasks are implemented as basic agents, organised in a hierarchical structure controlled by corresponding autonomous agents.

  The architecture and implementation of autonomous agents and basic agents are quite different. Autonomous agents modelling the behaviours of the system like an assembly agent or a collision avoider are connected with each other using a decentralised,

completely connected communication network. Generally, autonomous agents must communicate in order to achieve their goals. Since any autonomous agent is able to allocate system resources such as a camera or a robot, a special issue of communication is to coordinate the access to system resources. However, the resource management cannot be accomplished by the agents themselves. An additional global authority structure, which is defined for the particular society, determines which agent can allocate a system resource in a certain situation.

Furthermore, autonomous agents require the assistance of basic agents, which are responsible for specific tasks, like grabbing of camera images and recognition of objects. Contrary to the communication network of autonomous agents, basic agents are organised in a pre-defined hierarchical structure with corresponding autonomous agents on the top. The information about this structure is stored directly in the knowledge bases of the agents, i.e. each agent only knows about those basic agents it can interact with as well as the respective tasks those agents can accomplish.

As a testbed for the architecture, a purposive computer vision system performing simple visually guided assembly tasks has been implemented. Details of this system are described in [BIANCHI and RILLO 1997]. Although the architecture has been successfully tested, the applicability to more complex robotic scenarios is very doubtful. Firstly, the authority structure controlling the resource management as well as the hierarchical structure of the basic agents are defined in a statical way and depend on the specific purpose of the system. This inhibits the implementation of flexible systems, which must be able to accomplish various tasks in different situations. Moreover, the proposed system architecture provides just a very simple communication language, which makes it difficult or impossible to comply with the requirements of complex robotic setups.

- *IART:*
  Vuurpijl and Schomaker have proposed a framework called **I**ntelligent **A**gent system for pattern **R**ecognition **T**asks (IART), which has been employed to build a distributed recognition system for handwritten digits [VUURPIJL and SCHOMAKER 1998a, VUURPIJL and SCHOMAKER 1998b]. In contrast to other agent architectures dedicated to model object recognition systems, their agent society is only used as a supplementary system for performing additional verification and conflict resolution processes.

  The IART framework can be decomposed into two main parts: The first part of the system is modelled in a conventional way consisting of various processing modules which are independent from one another. In particular, a number of feature extraction modules, each using the information provided via one or more pre-processing and segmentation modules, extract different sets of features that are used by diverse classification modules to generate appropriate object hypotheses. The second part of the framework is modelled as a society of autonomous agents where the agents have to communicate with each other in order to analyse the pre-classified hypotheses in more detail and to perform an adequate conflict resolution. This is done, by utilising knowledge which may origin from different levels in the pattern recognition process.

  The global knowledge base used to store such information plays an important role in the pattern recognition architecture. It does not only contain knowledge about the

available pattern recognition modules, but also about which modules are suited for a particular pattern recognition task. The agents can employ this information to determine the processing modules which may provide additional information as well as to re-parameterise some of the modules to yield more accurate results in a specific situation.

Using this multi-agent system architecture Vuurpijl and Schomaker have implemented a recognition system for handwritten digits. The first part of this recognition system consists of several standard digit recognition methods. The second part is composed of intelligent agents, each of them designed to solve a particular conflict that may arise between two or more of the standard classifiers. That means, that the agent society only becomes activated if classifiers produce different hypotheses.

It should be mentioned, that the IART framework can be adequately applied to model recognition systems which are situated in well-defined environments such as the digit recognition system. However, in complex dynamic environments, where the systems must deal with changing requirements, different system architectures should perform better. The problem of the IART framework concerns both: firstly, the agents performing the conflict resolution in different situations must be implemented manually, and secondly, the global knowledge base will become very complex and difficult to survey.

- *Image understanding system:*
  The image understanding system proposed by Yanai and Deguchi is constructed as an assembly of agents, each of them responsible to recognise a particular kind of object [YANAI and DEGUCHI 1998, YANAI 1999]. The integration of the recognition results of different agents is not only accomplished by employing the believes of the hypothesised objects but also by incorporating additional relational knowledge concerning, e.g. the locations and sizes of objects.

  Each agent of this system consists of a recognition and a communication module. The former is designed to recognise a particular object as regions in an input image. This is done by using common object recognition methods based on segmentation and feature extraction algorithms. Depending on the support for an object, i.e. the proportion of detected image clues, the recognition module generates an appropriate object hypothesis which contains the image region as well as a score expressing the believe of the hypothesis.

  The communication module carries out the cooperation with other agents. To provide the consistency of the recognition results the communication module has relational knowledge concerning its own target object. This knowledge is stored as triplets of a source object, a relation, and a destination object, e.g. the agent responsible to recognise books has the relational knowledge ”book smaller-than table”. Based on these relations the communication module generates a second score measuring the consistency of the hypothesis with respect to the object hypotheses provided by other agents.

  The recognition process is generally the same: An input image is sent to all agents of the society where each agent starts to recognise the objects independently. If an agent has generated a new object hypothesis the corresponding information is broadcasted.

All others agent examine whether the provided information is consistent with their own hypotheses. If an agent detects an inconsistency, the agent sends an objection message to enforce a conflict resolution. In order to remove the least likely hypothesis the agents compare the recognition scores as well as the relation scores. This recognition strategy generally leads to consistent object recognition results. Examples for the recognition performance are described in more detail in [YANAI 1999].

Although this approach is based on an interesting idea, it can be expected that the efficiency of the recognition process drastically diminishes when the size of the object domain increases or when many similar objects are visible in the scene. In both cases a large amount of communication and computation must be spent to ensure the consistency of the recognition results.

6

DiVA: A Distributed Vision Architecture

This chapter concentrates on the development of a new multi-agent system architecture dedicated to model computer vision systems. This architecture can be utilised to enhance the flexibility and applicability of a vision system in many respects. The basic idea of the architecture is to model a vision system as a society of self-organising, autonomous agents, where each agent is responsible for specific vision tasks, the control strategy of a vision system is decentralised, and agents communicate using a flexible but easy understandable communication language. The chapter presents the general framework of the developed multi-agent system approach and presents a multi-agent recognition system employing the previously discussed recognition methods based on geometric and pattern invariants. Finally, it demonstrates experimental results which show the recognition performance, flexibility, and applicability of the proposed architecture.

6.1 Motivation

As mentioned in the previous chapter, agent technology can be utilised to construct computer vision systems that provide several advantages and out-perform vision systems based on conventional architectures. These advantages are mainly concerned with the flexibility, robustness and efficiency of the modelled systems (see also Sect. 5.3).

However, the multi-agent system architectures that have been developed in computer vision so far (see Sect. 5.4), generally do not make use of all of these advantages: At first, most of the developed architectures explicitly impose hierarchical structures to a vision system, such as a pre-defined static communication network or as an authority structure. This makes it

difficult to add, remove, or modify an agent in order to adapt the system to other (possibly changing) tasks and environmental conditions, because a single modification may affect many other agents of the society. Furthermore, the multi-agent system architectures employ communication languages which make it difficult to integrate the vision systems into other applications. On the one hand these languages are too simple to express complex facts and tasks, which restricts the flexibility of a system, and on the other hand they tend to be too cryptic making it difficult to specify messages correctly.

In contrast this chapter develops a multi-agent system architecture called DiVA (**Di**stributed **V**ision **A**rchitecture), which enhances the flexibility and applicability of the modelled vision systems by avoiding the aforementioned problems. Although this architecture is based on the same idea of modelling vision systems as societies of autonomous agents, each of them responsible for specific vision tasks, it differs in some important respects: (i) the proposed architecture avoids explicit hierarchical structures wherever possible; (ii) the control strategies of vision systems are decentralised; and (iii) the agents communicate using a flexible communication language, which is easy understandable. These properties directly lead to self-organising vision systems which accomplish vision tasks by goal-driven communication processes.

In the following sections this architecture will be described in great detail, where some of the items have been already presented in [GRAF and KNOLL 1999a, GRAF and KNOLL 1999b, GRAF and KNOLL 2000]. In addition the previously proposed recognition systems, i.e. the FII-recognition system (Sect. 3.3) and the PICA-recognition system (Sect. 4.3), will be modelled as assemblies of autonomous agents and are integrated into a single agent society. Many experimental results that demonstrate the recognition performance, flexibility, and applicability of this society will be discussed.

6.2 The DiVA System Architecture

In the proposed multi-agent system architecture, which is called **Di**stributed **V**ision **A**rchitecture (DiVA), a computer vision system is modelled as a society of agents, each one responsible for specific vision tasks. Since many of the computer vision algorithms are very time-consuming the architecture provides two different classes of agents: master agents and slave agents. The former accomplish all of the planning and interpretation as well as many of the required image processing tasks. The later are responsible to assist master agents in performing time-consuming tasks only, where the slave agents work in teams controlled by corresponding master agents to provide the distributed calculation of sub-results.

In the society the agents are connected to each other using a contract net, of which the topology is that of a completely connected network. The agents generally have to communicate in order to achieve their goals. Since the capability to communicate is an essential property of agents, the structure of the communication language has a great impact on the flexibility and applicability of the whole system. Therefore, the architecture incorporates a

communication language, which on the one hand can be simply employed, i.e. human readable and writable, and which on the other hand is flexible enough to describe also complex facts and tasks.

In the following this multi-agent system architecture will be explained comprehensively including the structures of master and slave agents, the communication language and network, and the interaction strategy.

6.2.1 Agent Architectures

Since master and slave agents are responsible for different purposes within a computer vision system they also differ in their architectures.

Master Agent

Master agents are modelled as autonomous agents, which generally must perform complex planning and interpretation tasks. To enable master agents to accomplish these tasks adequately, they share the agent architecture sketched in Fig. 6.1. As shown, the architecture is composed of five different modules:

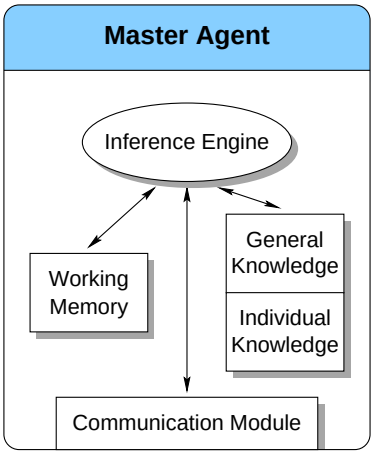


Figure 6.1: Architecture of a master agent

1. *Communication module:*
   The communication module is responsible for connecting to other agents. It contains methods for sending and receiving messages as well as functions for wrapping various data types, which are required to solve the particular tasks to which the agent belongs.

2. *General knowledge:*
   This data base is used for storing general knowledge, like basic planning strategies and the grammar of the communication language. Most of the knowledge is stored in rules and facts (see Appendix D) that are applicable in different situations and determines the general behaviour of master agents.

3. *Individual knowledge:*
   The individual knowledge base contains all knowledge concerning the particular agent, like specific planning strategies, processing functions and the provided vocabulary. Note, that it is not required that all agents share the same vocabulary. Similar to the general knowledge base the individual knowledge is stored in rules and facts and determines the individual behaviour of each agent.

4. *Inference engine:*
   Based on general and individual knowledge the inference engine accomplishes all planning and interpretation tasks of the agent. The most extensive work that must be performed by the inference engine is (i) to generate program scripts appropriate to solve requested tasks, that are specified by an abstract description, and (ii) to execute the program scripts.

5. *Working memory:*
   The working memory is used by the inference engine for storing various information, like sub-results and knowledge about the dynamic environment.

In order to perform a particular vision task which is specified by an abstract description (see Sect. 6.2.2) the responsible agent proceeds as follows: Firstly, the agent analyses the abstract task description including the instruction and destination specifications as well as the additional constraints. However, it is generally not required that the agent has to understand all of the source specifications, because the agent is able to request further assistance of the society. According to the description of the task the agent automatically generates a complex *Clips*-style [CLIPS 1998] program script, that is composed of all required processing functions as well as message passing to other agents. Lastly, the program script is executed under control of the agent, in order to enable the agent to perform an exception handling and to react to unexpected situations. This processing strategy of the agent is described in Sect. 6.2.4 in some more detail and an example for a program script is shown in Tab. 6.3.

Slave Agent

Although a computer vision system can be modelled by employing but master agents, the multi-agent system architecture provides an additional slave agent class. Since many of the image processing algorithms are very time-consuming, the slave agent class is provided as a tool that facilitates the implementation of parallel processing methods to speed up computation.

The particular purpose of slave agents is simply to assist master agents in performing time-consuming tasks, where the slave agents often can communicate with their corresponding

masters only. Slave agents are completely controlled by master agents, i.e. the master decides how many slave agents he wants to use as well as which particular processing function must be performed. The communication between a master and its slaves is reduced to the absolute minimum. Generally, a master agent only determines the processing functions and additional parameters.

Therefore, slave agents only need a simple architecture containing a communication module, processing functions and rudimentary mechanisms for interpreting messages.

6.2.2 Communication Language

As already mentioned, the communication language has a great impact on the flexibility and applicability of the whole system. However, the syntax of communication languages used in the computer vision domain generally tends to be cryptic and too simple to express complex facts and tasks; this is true especially for that part of the communication language which is relevant for the application itself.

Therefore, a more flexible communication language is developed considering the following basic requirements:

- the communication language must permit the construction of flexible and self-organising vision systems,
- it must be able to express complex facts and tasks,
- it must be simple to understand, i.e. human readable and writable, and
- efficient mechanisms for interpreting messages must be provided.

In contrast to other communication languages, which just permit messages composed of function names and parameters, the developed one can be utilised to specify a task by an abstract description which is independent of the underlying implementation. Therefore, agents can be added, replaced, and modified without having any knowledge of the internal structures of the existing agents.

The developed communication language employs message types making the intention of a message explicit:

```
<message> ::= <message-type> <message-content>
```

The allowed message types are similar to the ones used in other communication languages [BIANCHI and RILLO 1997, BOISSIER and DEMAZEAU 1994a] in that they implement speech acts but differ in some important respects:

1. *request:*
   The message type `request` is used to request the assistance of other agents. Generally, this message type indicates that the agent cannot perform a particular task on its own.

2. *answer:*
   An `answer` message is a reply to a request or script message, which can be both a result or an error message.
3. *inform:*
   The message type `inform` is used for passing additional information to other agents which is not necessarily needed for performing particular tasks. This type is also used for indicating the presence or absence of agents.
4. *script:*
   The message type `script` can be used for gaining direct access to the capabilities of an agent avoiding its interpretation mechanisms. Since master agents generate program scripts in order to perform requested tasks dynamically, programs can be passed directly. The advantage of sending scripts is that agents can be used to flexibly perform complex algorithms which are neither pre-defined nor have been implemented in the agents.

Furthermore the message content is sub-divided into a message text and additional message data:

```
<message-content> ::= <message-text> <message-data>
```

where the `<message-text>`-slot contains a message text as a string and the `<message-data>`-slot is a list capable for storing different data types like images and edges. The text string itself can be both (i) an abstract task or fact description, or (ii) a *Clips*-style [CLIPS 1998] program script:

```
<message-text> ::= <abstract-description> | <program-script>
```

Depending on the message type, the message text will be interpreted as a task or fact description (`answer`, `inform`, `request`) or as a program script (`script`).

Abstract task and fact descriptions

The common and most flexible way to perform communication in the multi-agent system architecture is provided by employing abstract, easy understandable fact and task descriptions. These descriptions are independent of the underlying implementation of the processing functions of an agent and are specified in *Clips* notation.

An excerpt of the formal grammar used for specifying such descriptions is shown in Tab. 6.1 (see also Appendix E for the complete grammar of the *recognition fusion agent*). For reasons of efficiency this grammar allows only one goal per message, where a goal is not a specific entity but rather some kind of global plan, which can be restricted by supplementary conditions. These conditions can be complex logical expressions containing variables

as well. Furthermore, the grammar can be simply expanded by allowing additional constraints or information. For example, a specification of the quality of input data such as the illumination conditions of an input image could be utilised to enhance the computed results.

Table 6.1: Excerpt of formal grammar for specifying messages

```
<abstract-description> ::= (<goal> <variable>*) <goal-condition>*

<goal>                 ::= convert | extract | introduce | ...
<goal-condition>       ::= <attr-condition> | <not-condition> |
                           <and-condition>  | <or-condition>

<attr-condition>       ::= <is-a-attr> | <has-type-attr> | ...
<not-condition>        ::= (not <goal-condition>)
<and-condition>        ::= (and <goal-condition>+)
<or-condition>         ::= (or <goal-condition>+)

<is-a-attr>            ::= (is-a <variable> <object-class>)
<has-type-attr>        ::= (has-type <variable> <type-spec>)

<object-class>         ::= feature | image | ...
```

A simple example for an object recognition task, which possibly must be solved during an assembly process in the robotic application shown in Fig. 1.1, is to extract all *slats* as well as all known red objects shown in an image taken from a camera, of which the server is called 'penelope'. This task can easily be specified using the following description:

```
(recognise ?dest ?src)

(is-a    ?dest object)
(or (has-name ?dest slats) (has-color ?dest red))

(is-a    ?src  image)
(has-source ?src camera) (has-server ?src penelope)
```

There are two important points to note here: Firstly, the interpretation of such an abstract description (see also Appendix E) can be realized in a very efficient manner using the pattern-matching facilities of expert system tools; and secondly, entities can be identified not only by their unambiguous names but also by their features and attributes. Although such querying requests are very useful and important to vision applications, they are generally not supported by other agent-based vision systems.

Program Scripts

As an additional and interesting feature, the communication language permits communication and interaction among agents not only by using abstract fact and task descriptions but also by employing complex program scripts. Since these program scripts can be directly executed by the inference engine, scripts can be utilised (i) to gain direct access to the capabilities of an agent by avoiding the mechanisms for interpreting abstract descriptions, and (ii) to flexibly accomplish algorithms which otherwise would not be supported by an agent, i.e. they are not pre-defined.

Program scripts are specified in *Clips* [CLIPS 1998] which allows the construction of complex programs including e.g. branches, loops, as well as user-defined functions. In particular, arbitrary programs can be specified, which may also include further requests to other agents. Examples for such program scripts which enable an agent to perform vision tasks that have not been pre-defined in that agent are demonstrated in Sect. 6.4.3.

However, it must also be noted that the use of program scripts entails a loss of flexibility: In general program scripts diminish the possibility of modifying agents, because scripts often rely on the implementation of an agent i.e. on its specific algorithms, so that a program script should always be sent to a particular agent.

6.2.3 Communication Network

The master agents of a society are connected with each other through a decentralised contract net, of which the topology is that of a completely connected network. No further constraints such as pre-defined hierarchical structures are imposed. That means, that each master agent can communicate with every other master agent of the society.

In contrast, since slave agents are provided as a tool to assist master agents in performing time-consuming tasks by facilitating the implementation of parallel computation, slave agents can often communicate and interact with their corresponding masters only. In particular, the communication between master agents and their slaves proceeds as follows: If a master agent requires the assistance of slave agents, it sends a connection request to a specific type of slave agents, which are able to accomplish the required task. Then, all slave agents of the specified type reply to the connection requests and the master agent selects all slaves that should form a team and should accomplish the required task. The master agent splits up the task into appropriate sub-tasks and sends each sub-task to a particular slave agent which is part of the team. After the slaves have performed the sub-tasks, the master agent merges the sub-results. Finally the master decides, whether the slave agent team should be released or if it should be used for further requests. Thus, master and slave agents are connected in flexible hierarchical structures.

An example for such hierarchical structures is schematically sketched in Fig. 6.2. The example demonstrates three different types of connections among master agents and their corresponding slave agents, which often occur during accomplishing a requested vision task: The

first type consists of a single master agent, which is able to perform all required processing functions on its own, while the second type is composed of a master agent which divides time-consuming tasks into sub-tasks in order to be solved by its corresponding slaves. The third type consists of two (or more) master agents, which are responsible for the same tasks. Again, they divide sub-tasks in order to be accomplished by their slaves, which can be dynamically shared between the master agents. Moreover, it is also possible that two different types of master agents share the same slave agents when some sub-tasks are the same or very similar.

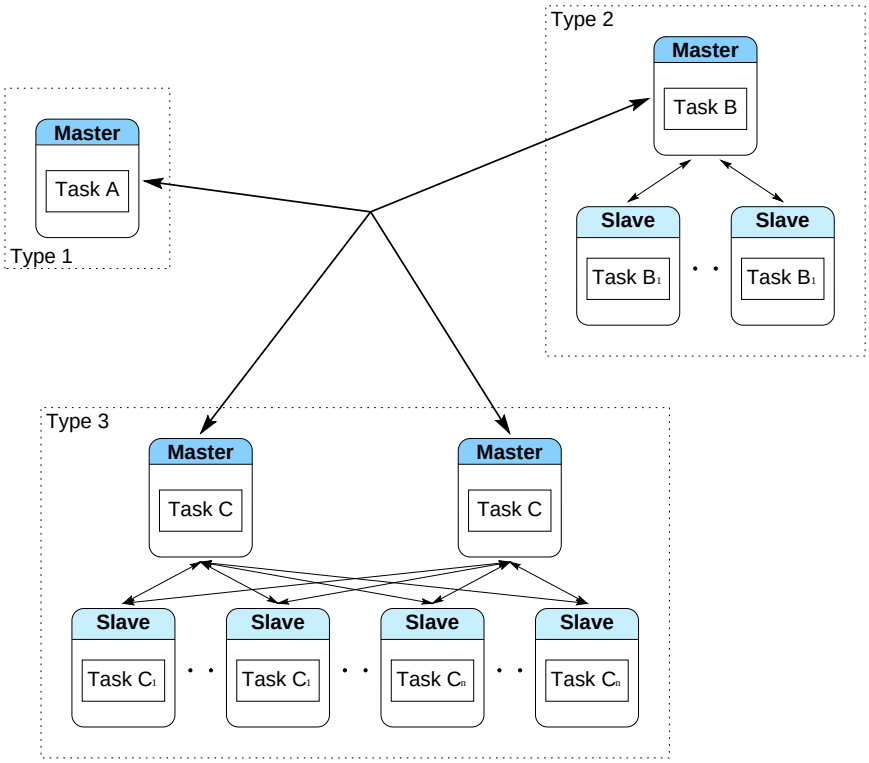


Figure 6.2: Example for possible connections among master and slave agents

It is very important to note, that the hierarchical structure between master and slave agents is not pre-defined in a statical way, but is rather a result of communication processes. The structure may dynamically change due to several reasons, e.g. if additional agents are instantiated or if existing agents are deleted at run-time.

In general, to provide a high degree of flexibility, pre-defined hierarchical structures should be avoided wherever it is possible, because any imposed hierarchical structure will make it difficult to add, remove or modify agents. For example in the purposive computer vision system (see Sect. 5.4) the modification of an agent or of a behaviour may affect many other agents entailing the modification of these agents too.

6.2.4 Interaction Strategies

Another important aspect concerning the flexibility of the agent society is the interaction strategy among master agents. In many implemented multi-agent systems the communication ability is restricted and agents can pass messages only to pre-defined subsets of the agent society. This is done to reduce the communication overhead and the computational load. However, such an interaction strategy also entails a loss of flexibility, because the modification of a single agent may affect many other agents of the society.

In the proposed multi-agent system architecture a similar problem arises if program scripts are utilised to perform the communication and interaction among agents because a program script is generally addressed to a particular agent which is capable to accomplish the script.

In contrast the use of abstract fact and task descriptions leads to very flexible vision systems. In these systems the control mechanisms are decentralised, i.e. each master agent accomplishes requested tasks according to its own knowledge and goals, where generally no further control mechanisms, like explicit hierarchical structuring, are imposed. Abstract descriptions are usually sent to the whole agent society and each agent tries to react to the specified request. Therefore, the interaction among agents is performed by communication processes which result in a self-organisation of the agent society. This self-organisation process is goal-driven and proceeds as follows:

- If a master agent requests a vision task, all master agents of the society decide if they can accomplish the given task. As mentioned before, this is done by analysing the instruction, the destination specifications, and the additional constraints. However, the source specifications of the task are often neglected.
- All master agents that are responsible for the particular task make a bid. According to these bids the master agent, which has requested the task, selects the agents that should award the contract.
- Then, the selected agents generate appropriate program scripts according to the task specifications. If the source of the task is unknown or additional assistance must be provided the selected agents generate further requests to gain a required sub-result.

A comprehensive demonstration and discussion of this self-organisation process is provided in Sect. 6.4.1.

There are some important points to note here: A drawback of this interaction strategy is that it can be established just at run-time if the vision system can accomplish a particular vision task. Furthermore, the interaction strategy produces some overhead because all master agents try to react to a request. Nevertheless, this overhead can be neglected since most of the vision algorithms are very time-consuming compared to the overhead.

The advantage of this approach is the flexibility of the resulting vision systems: agents can be added and deleted at run-time without causing any problems. Furthermore, new functionality can be provided by simply adding appropriate agents without having precise knowledge of the internal structure of existing agents.

6.3 A Multi-Agent Recognition System

In order to demonstrate the flexibility and applicability of the proposed multi-agent system architecture, the FII-recognition system (Sect. 3.3) and the PICA-recognition system (Sect. 4.3) have been modelled as assemblies of autonomous agents, which are integrated into a single agent society.

6.3.1 Generic Agent Design

The agents of the proposed multi-agent recognition system have been implemented in C/C++ using the multi-agent generation tool *MagiC* [SCHEERING and KNOLL 1998, SCHEERING 2000] and the expert system tool *Clips 6.10* [CLIPS 1998].

In particular, *MagiC* is a C++-library which facilitates the generation of multi-agent systems based on contract net approaches. It provides several classes to build different types of agents and mechanisms encapsulating all of the negotiation protocols and strategies. For example, an agent can be constructed by inheriting the basic functionality from one of the provided agent base classes and by implementing its virtual functions, which are automatically called in specific situations such as the `make_bid` function which is called when the implemented agent receives a request and the `service` function which is automatically called if the agent awards a contract. For a more comprehensive description of the multi-agent generation tool *MagiC* see [SCHEERING 2000].

The expert system tool *Clips 6.10* is a productive development and delivery tool which provides a complete environment for the construction of rule and/or object based expert systems. This tool is provided as a complex C-library, that implements a *LISP*-style programming language as well as an efficient inference engine, which can be completely controlled by C/C++-programs. Furthermore, *Clips* can be extended by embedding user-defined C-functions within its environment. This makes it easy to model and to handle the wide variety of knowledge required to build autonomous agents. A complete introduction to *Clips 6.10* can be found in [CLIPS 1998].

Figure 6.3 sketches the generic design of master agents and indicates how the different tools and modules interact with each other. For convenience the expert system tool *Clips* has been divided into the *inference engine* and the *knowledge base*.

Although master agents are constructed to solve diverse vision tasks, the information flow within a master agent generally proceeds as follows: When a master agent receives a message that may include either an abstract fact/task description or a program script, its *agent interaction interface* provided by the *MagiC*-library automatically calls appropriate virtual functions such as the aforementioned `make_bid` and `service` functions. These functions translate the received C++-message into *Clips*-notation, store it in the *knowledge base* of the agent (1), and initiate the *inference engine* (2). Then, the *inference engine* utilises the stored knowledge of the agent (3), which is provided as facts and rules, to generate an appropriate answer.

For example, if the `make_bid` function of the agent has been called, the *inference engine* generates an appropriate bid message, which will be converted afterwards by the *agent interaction interface* into a C++-message, that will be passed automatically to the requester. However, if the agent has awarded the contract and the `service` function has been called, the agent tries to accomplish the requested task. In the case of an abstract fact/task description the *inference engine* has to generate an appropriate *Clips* program script while the program provided by a `script`-message can be used directly. In both cases the program scripts may include complex user-defined *service functions* (4) such as image processing functions, which receive their parameters from the *knowledge base* (5). In some situations the *service functions* generate additional requests other agents (6). Finally, the *inference engine*, i.e. the *Clips*-interpreter, executes the *Clips* program and generates an answer, which might also be an error message, and attaches the calculated results. Again, the *agent interaction interface* takes over, converts the *Clips*-message into a C++-version, and sends it to the requester.

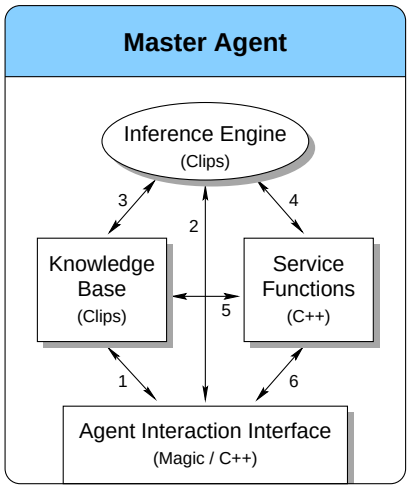


Figure 6.3: Generic agent design

In contrast to master agents, slave agents share a simpler design. Since slave agents generally assist a particular master agent in performing time-consuming tasks only, i.e. slave agents are completely controlled by corresponding master agents, they just require the *agent interaction interface* and the *service functions*. As a result the information flow within slave agents is very simple: When a slave agent has awarded the contract to accomplish a particular task, the master agent sends an appropriate request message which is composed of the function to be accomplished as well as the required parameters. Then, the *agent interaction interface* of the slave agent performs a simple pattern matching process in order to switch to the correct *service function*. Finally, the computed results are stored in an answer message, which will be resent to the master.

6.3.2 Implementation

The following section describes the implementation of an agent society, that integrates the object recognition systems proposed in Sects. 3.3 and 4.3. In this society the recognition methods are not modelled as single agents but the methods are splitted into appropriate sub-tasks which are accomplished by different agents. This is done, (i) to improve the flexibility of the resulting system, (ii) to make the system more robust, and (iii) to enable the distributed calculation of sub-results in order to speed up the recognition process.

In particular, the society is composed of eight different agents:

- *Communicator agent:*
  The communicator agent provides an intelligent graphical user interface, which facilitates the communication with the agents of the proposed society.

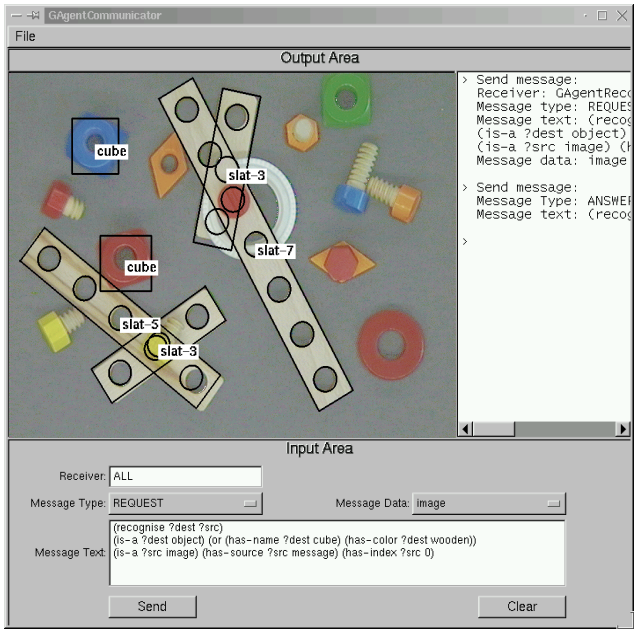


Figure 6.4: Graphical user interface provided by the communicator agent

The design of the interface is sketched in Fig. 6.4. As shown, this graphical interface is composed of an input and an output area. The former enables the user to specify an arbitrary vision task by determining an (optional) target agent, the message type and text, which can be either an abstract fact/task description or a complex program script (see also Sect. 6.2.2), as well as additional message data. The output area is responsible to display various information about the message passing process. For

example, the output area displays the incoming messages including the message texts as well as the calculated data, such as images, extracted features, or recognised object hypotheses, which are automatically converted in order to be adequately drawn in the graphic frame.

- *Master/slave image processing agents:*
  The master and slave image processing agents are responsible for low-level vision tasks, such as image smoothing, segmentation, or the extraction of edge images. Since most of these vision tasks are very time-consuming, both types of agents are provided, i.e. a master image processing agent as well as a slave image processing agent.

  The master image processing agent accomplishes the communication and interaction with the agent society and also performs some of the simpler vision tasks. Furthermore, the master agent incorporates strategies for splitting up complex image processing tasks in order to be accomplished by a dynamic team of slave agents. This slave agent team is completely controlled by the master agent: First, the master image processing agent determines the number of available slave agents that should form a team. Then, the master agent divides the given vision task into appropriate sub-tasks and sends each sub-task to a particular slave agent. After these sub-tasks have been solved, the master agent merges the sub-results of the slave agent team and generates the final result.

  Although the appropriate splitting process depends on the specific vision task, many of the low-level image processing tasks can be distributed by (i) cutting the input images into different slices, and (ii) performing the required operation on each slice individually. This splitting strategy has been used e.g. for the extended Canny edge operator (see Sect. 3.3).

- *Master feature extraction agent:*
  The feature extraction agent is mainly present to extract the geometric features that are required by the FII-recognition method. In particular the agent knows how to extract edge points from an image and how to fit geometric primitives such as straight lines and ellipses. The employed algorithms are equivalent to the ones described in the feature extraction stage of the FII-object recognition system (see Sect. 3.3).

- *Master/slave FII-recognition agents:*
  The FII-recognition agents accomplish the recognition method based on geometric invariants by implementing the FII-recognition system described in Sect. 3.3. In particular, these agents utilise extracted image features to determine invariant object descriptions, which are employed to generate appropriate object hypotheses that are verified in a final verification process.

  Since the hypothesis generation as well as the verification process are very time-consuming, master and slave agents have been constructed. Similar to the image processing agents, the master FII-recognition agent (i) forms an appropriate slave agent team, (ii) divides the task into sub-tasks, and (iii) sends them to the different

agents of the slave team. For example, the set of measured invariants is splitted into disjoint subsets in order to enable the distributed generation of hypotheses.

- *Master PICA-recognition agent:*
  This agent implements the appearance-based PICA-recognition system proposed in Sect. 4.3. Contrary to the FII-recognition method, the PICA-recognition process has much more computational efficiency, so that all required processing functions are performed by the master agent itself. That means, the master agent employs image regions, which may be segmented by the provided image processing agents, to extract appropriate image patches. Furthermore, the agent determines the corresponding pattern invariants and accomplishes the hypothesis generation as well as the verification process (see Sect. 4.3).
- *Master recognition fusion agent:*
  The recognition fusion agent has the capability to combine the recognition results obtained by different recognition agents, i.e. the agent must handle different and competitive information. In the proposed agent society the fusion agent compares the object hypotheses generated by the FII- and the PICA-recognition agents. Although the fusion agent does not have comprehensive knowledge about the different objects, it generally enhances the recognition results by selecting the most likely hypotheses. The selection process is based on simple rules; e.g. if the image regions of different object hypotheses overlap each other, the fusion agent suppresses those hypotheses that mostly differ in size to other hypotheses of the same class recognised in the image.

6.4 Experimental Results and Discussion

This section demonstrates and discusses various aspects concerning the performance, flexibility, and applicability of the proposed multi-agent vision system: First, an example for a trace of a recognition process is demonstrated. It shows how the agent society organises itself by goal-driven communication processes in order to accomplish a given vision task. Secondly, the recognition results obtained for the aforementioned test scenes (Sects. 3.4 and 4.4) are discussed; and finally, it is demonstrated how the use of program scripts can be utilised to achieve functionality that has not been pre-defined in the agent society.

6.4.1 Trace of a Recognition Process

To provide a comprehensive insight into the self-organisation process of the proposed agent society, a number of agents of each type has been run on different platforms including Linux-PCs and Sun-Solaris-Workstations. Although no explicit hierarchical structure has been imposed on the system, the agent society is capable of solving also complex vision tasks.

For example, if one requests the task of recognising all *cubes* as well as all wooden-coloured objects in an image attached to the message, which is given by the abstract task description:

```
(recognise ?dest ?src)

(is-a      ?dest object)
(or (has-name ?dest cube) (has-color ?dest wooden))

(is-a      ?src  image)
(has-source ?src message) (has-index ?src 0)
```

the agent society takes on the transient system structure as sketched in Fig. 6.5. The corresponding trace of message passing is shown in Tab. 6.2.

In particular, the *agent communicator* passes the specified task description to the whole agent society and every master agent tries to react to the requested task (see also Appendix E). Although the *recognition fusion agent* as well as the *master FII-* and the *PICA-recognition agents* know how to recognise objects in images, the *recognition fusion agent* only is awarded the contract (1). In contrast to the other recognition agents the *fusion agent* knows about *cubes* as well as about wooden-coloured objects (the *slats*), so that it can be expected in this case that the *fusion agent* should achieve the best recognition results. However, the agent cannot accomplish the requested task on its own. Since this agent only knows how to combine the object hypotheses provided by other agents, it broadcasts the recognition task again. Now, the *master FII-recognition agent* (2a) as well as the *PICA-recognition agent* (2b) are awarded the contract, because they are able to recognise different objects that match the task description. In the following these agents perform the recognition processes simultaneously:

The *master FII-recognition agent* tries to recognise the wooden-coloured *slats* in the image. In order to solve this recognition task the agent needs geometric primitives (especially lines and ellipses) extracted in the image. Since these primitives are not provided by the message data, the agent requests to extract the geometric primitives from the specified image (3a). Next, the *feature extraction agent* is awarded the contract, even though the agent does not understand the source specifications (it only knows about edge images). Again, this agent needs the assistance of the agent society to extract the required edge image from the unknown source (4a). This sub-task is solved by the *master image processing agent*. This agent cuts the image stored in the message into different slices and asks its slaves to apply an appropriate edge operator, where each agent works on a particular image slice (5a, 6a). Note, that the communication between master and slave agents is very simple. Using the resulting edge image (7a) the *feature extraction agent* extracts the requested geometric primitives and resends them to the *master FII-recognition agent* (8a). Now, the *master FII-recognition agent* is able to recognise the specified *slats*. This is done with the assistance of its slave agents, which perform the hypotheses generation as well as the verification of the hypotheses (9a-12a). Then, the recognised *slats* are passed to the *recognition fusion agent* (13a).

Similar, the *PICA-recognition agent* tries to recognise the *cubes* in the image. Since the implemented recognition process employs image regions, the agent requests the society to

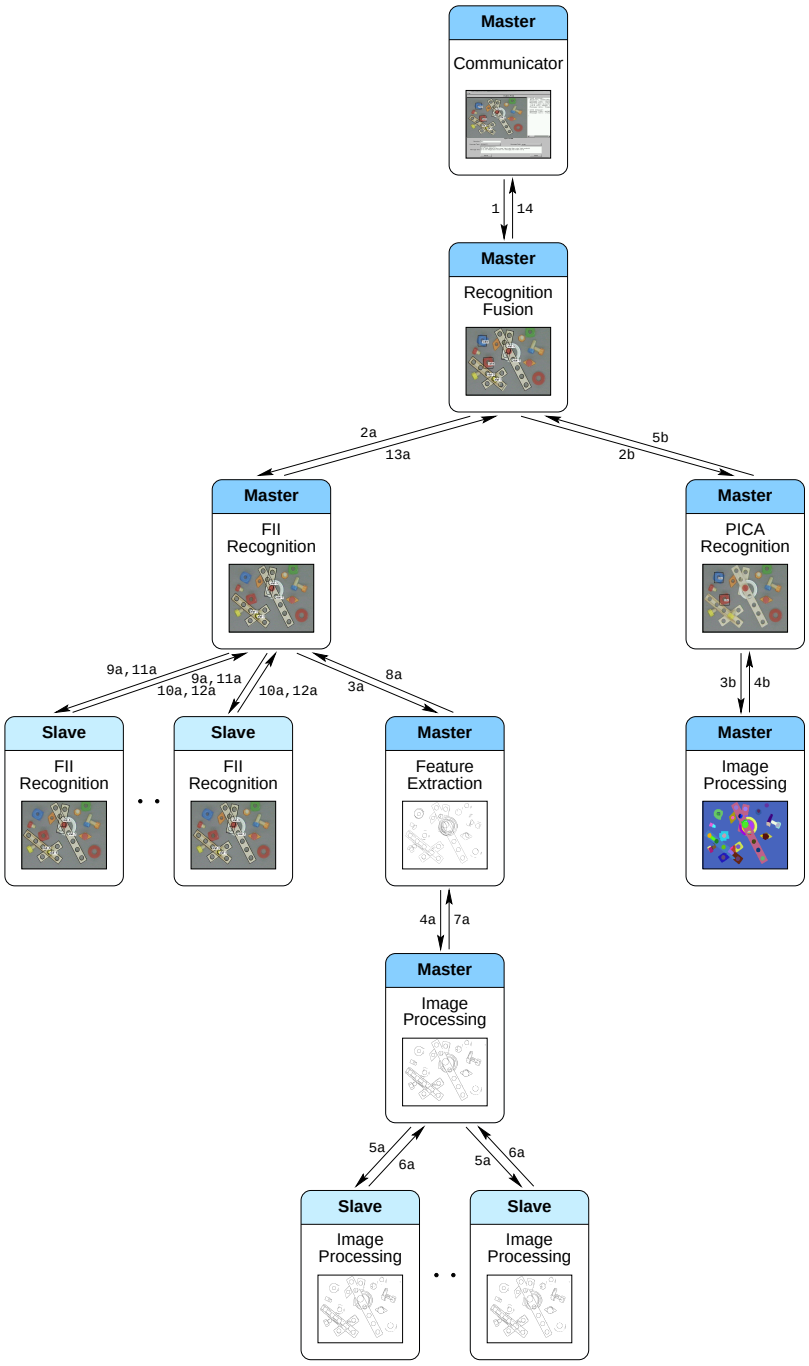


Figure 6.5: Self-organised system structure

Table 6.2: Trace of message passing

```
 1 : REQUEST: (recognise ?dest ?src)
              (is-a ?dest object)(or (has-name ?dest cube)
                                     (has-color ?dest wooden))
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)

 2a: REQUEST: (recognise ?dest ?src)
              (is-a ?dest object)(or (has-name ?dest cube)
                                     (has-color ?dest wooden))
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)
 3a: REQUEST: (extract ?dest ?src)
              (is-a ?dest feature)(or (has-type ?dest line)
                                      (has-type ?dest ellipse))
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)
 4a: REQUEST: (extract ?dest ?src)
              (is-a ?dest image)(has-type ?dest edge)
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)
 5a: REQUEST: (apply-canny)
 6a: ANSWER:  (apply-canny)
 7a: ANSWER:  (extract edge-image image-0)
 8a: ANSWER:  (extract lines UNKNOWN-2)(extract ellipses UNKNOWN-2)
 9a: REQUEST: (generate-hypotheses)
10a: ANSWER:  (generate-hypotheses)
11a: REQUEST: (verify-single-hypotheses)
12a: ANSWER:  (verify-single-hypotheses)
13a: ANSWER:  (recognise ledge-3 image-0)(recognise ledge-5 image-0)
              (recognise ledge-7 image-0)

 2b: REQUEST: (recognise ?dest ?src)
              (is-a ?dest object)(or (has-name ?dest cube)
                                     (has-color ?dest wooden))
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)
 3b: REQUEST: (segmentate ?dest ?src)
              (is-a ?dest region)
              (is-a ?src image)(has-source ?src message)(has-index ?src 0)
 4b: ANSWER:  (segmentate region image-0)
 5b: ANSWER:  (recognise cube image-0)

 14: ANSWER:  (recognise cube image-0) (recognise ledge-3 image-0)
              (recognise ledge-5 image-0) (recognise ledge-7 image-0)
```

segmentate the image. Again, this sub-task is accomplished by the *master image processing agent* (3b,4b). By employing the segmented image regions, the *PICA-recognition agent* generates the requested *cube* hypotheses and sends them to the *recognition fusion agent* (5b).

After the *fusion agent* has received the object hypotheses generated by both recognition

agents, the *fusion agent* performs a last verification process. Finally, the initial recognition task is accomplished and the resulting hypotheses are sent to the *communicator agent* (14).

During this recognition process, the master agents have automatically generated *Clips*-style program scripts, that solve requested sub-tasks. An example for such a program script, which has been generated by the *feature extraction agent* in reply to request (3a), is shown in Tab. 6.3. This program is subdivided into three different sections: In lines 01–04 the *feature extraction agent* requests an edge image from the agent society. In lines 05–10 all feature extraction functions are performed, i.e. the fitting of straight lines and ellipses. These functions have been implemented in C++ and are embedded within the *Clips* environment. Lastly, in lines 11–13 an answer is generated.

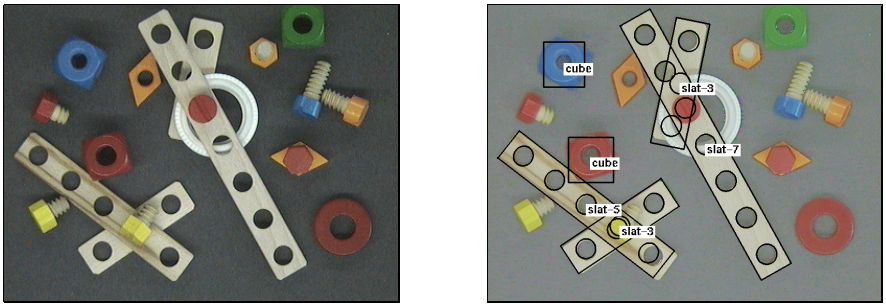
Table 6.3: Script generated by *feature extraction agent* in reply to request (3a)

```
01: (bind ?var-gen7 (make-instance instance-gen9 of GMessage
     (type REQUEST)))
02: (send ?var-gen7 put-text
    "(extract ?dest ?src)(is-a ?dest image)(has-type ?dest edge)
     (is-a ?src image)(has-source ?src message)(has-index ?src 0)")
03: (send ?var-gen7 put-data (send [input-message] get-data))
04: (send-message (instance-name-to-symbol ?var-gen7))

05: (bind ?var-gen10 (send ?var-gen7 get-data 0))
06: (bind ?var-gen3 (extract-edge-points ?var-gen10 10))
07: (delete-data ?var-gen10)
08: (bind ?var-gen5 (extract-conics ?var-gen3 ellipse))
09: (bind ?var-gen2 (extract-lines ?var-gen3))
10: (delete-data ?var-gen3)

11: (send [output-message] put-type ANSWER)
12: (send [output-message] add-text "(extract lines UNKNOWN-2)
     (extract ellipses UNKNOWN-2)")
13: (send [output-message] add-data ?var-gen2 ?var-gen5)
```

The final recognition result of the requested task is depicted in Fig. 6.6, where Fig. 6.6a is the original image and Fig. 6.6b shows the recognised objects. As can be seen, the multi-agent system architecture is able to recognise the objects that match the given object specifications, namely two *3-holed-slats*, a *5-holed-slat*, a *7-holed-slat* as well as two cubes. All other objects present in the image are ignored. Unfortunately, the system fails to detect one of the cubes, where this deficiency is not a problem of the agent architecture but is rather a result of an inaccurate image segmentation.

There are some important points to note here:

- In general the multi-agent vision system organises itself in order to solve a given vision task. Although all types of agents are involved in the aforementioned recognition example, many other vision tasks can be accomplished by employing just a fraction of the



(a) Original image (b) Result

Figure 6.6: Recognition result of requested task

agent society. For example, the task of recognising *slats* requires the FII-recognition, the feature extraction, and the image processing agents, while the task of extracting an edge image is performed by using the image processing agent only.

- The addition or deletion of agents at run-time causes no problems of system stability, assuming that all agents necessary to accomplish a task are available. However, even if all agents of a particular type are deleted which are required to perform a specific vision task, the system may still be able to generate sub-optimal results. For example, if the feature extraction agent in the aforementioned example is deleted, the system has still the capability of recognising the *cubes*. Thus, the modelled vision system is robust with respect to breakdowns of single agents. This is true, especially if more than one agent of each type are instantiated.

- The multi-agent vision system provides scalability, i.e. new agents can be instantiated in order to speed up computation. For example, new slave image processing agents can be added to an agent society, so that edge images can be obtained much faster.

- Furthermore, new functionality can be provided by building new types of agents. Since the agents of a society are independent of each other and they communicate using abstract fact/task descriptions this can be done without having precise knowledge of the internal structure of existing agents.

- Finally, the proposed multi-agent system architecture permits the construction of more complex vision systems that incorporate bottom-up as well as top-down processing methods in a single system. For example, the recognition results might be utilised to trigger the feature extraction agent in order to search for image features that have not been found in a first extraction process.

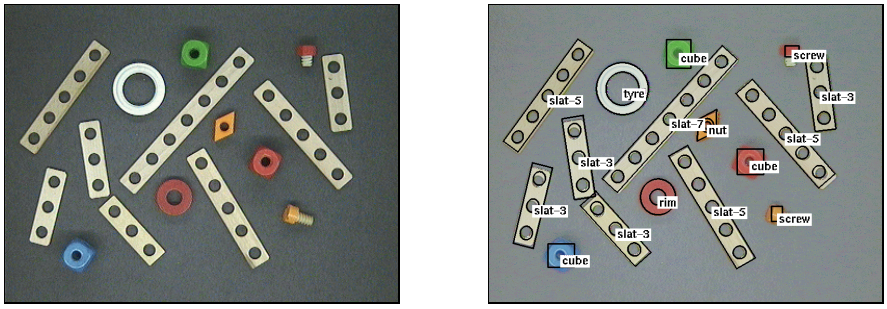
These properties are an important factor for both the autonomy as well as the flexibility of the modelled system.

6.4.2 Recognition Performance

This section discusses the recognition performance of the developed multi-agent recognition system. Since the agent society implements the FII- (Sect. 3.3) as well as the PICA-recognition method (Sect. 4.3), the society is able to recognise all of the Baufix objects described in appendix B. In order to enable a comparison between the recognition results obtained by the different systems, the agent society is applied to same set of test images and is requested to recognise generally every known object present in the images.

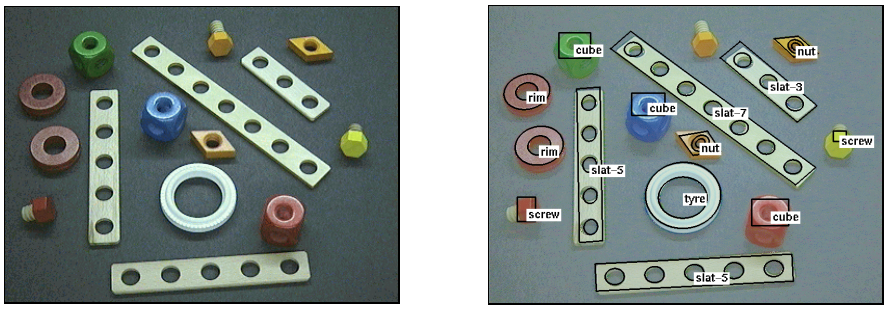
Results for Test Scenes Containing Only Unoccluded Objects

At first, the society is applied to test scenes that are only composed of unoccluded objects. In general, the society has the capability to provide rich descriptions of such scenes. For



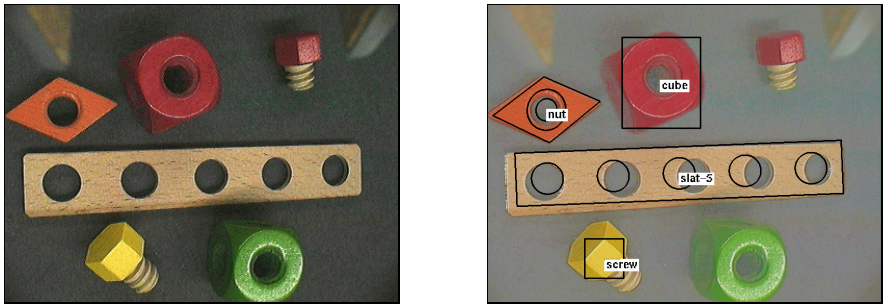
(a) Original image (b) Result

Figure 6.7: DIVA result: unoccluded objects taken with a top-view camera



(a) Original image (b) Result

Figure 6.8: DIVA result: unoccluded objects taken with a front-view camera



(a) Original image (b) Result

Figure 6.9: DIVA result: unoccluded objects taken with a hand-camera

example, in Figs. 6.7 – 6.9 the society has detected most of the objects correctly, even though the images originate from different viewpoints and cameras: Fig. 6.7 has been taken with the top-view, Fig. 6.8 with the front-view, and Fig. 6.9 with a hand-camera. Obviously, the *recognition fusion agent* has successfully merged the object hypotheses provided by the *FII-* and the *PICA-recognition agents*.

The recognition results gained for this type of test scenes are summarised in Tab. 6.4. If these results are compared with the ones obtained for the FII- and the PICA-system, some interesting and important properties of the agent society becomes apparent: First, the agents that implement the complementary recognition methods do not affect each other

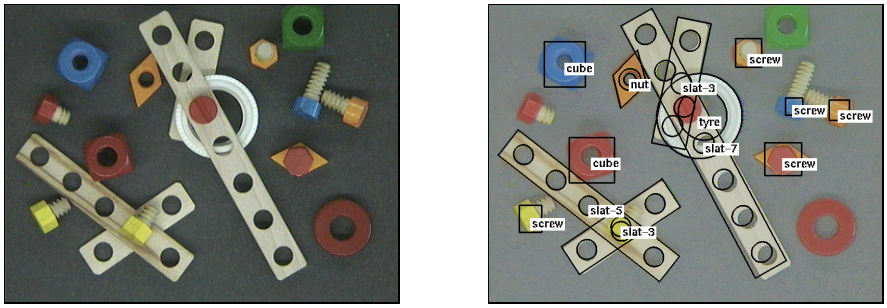
Table 6.4: Recognition results for unoccluded scenes

	present	false negatives		false positives		correct	
cube	396	73	(18%)	43	(11%)	291	(73%)
nut	410	35	(9%)	5	(1%)	370	(90%)
rim	179	28	(16%)	27	(15%)	124	(69%)
screw	691	306	(44%)	28	(4%)	372	(54%)
slat-3	252	7	(3%)	9	(4%)	240	(95%)
slat-5	162	6	(4%)	9	(6%)	150	(93%)
slat-7	105	0	(0%)	9	(9%)	99	(94%)
tyre	136	2	(1%)	1	(1%)	133	(98%)
$\sum$	2331	457	(20%)	131	(6%)	1779	(76%)

in a negative way: The recognition rates for the *rims*, *slats*, and *tyres* are similar to the FII-recognition rates, while the rates for the *cubes* and *screws* are similar to the PICA-recognition rates. As indicated by the increased recognition rates for the *nuts*, which can be recognised by both methods, the multi-agent vision architecture can be rather utilised to integrate different recognition approaches in a beneficial way. Furthermore, the agent society has generated considerable less false positives. This advantage is achieved by the *recognition fusion agent*, which compares the object hypotheses of the different recognition methods in order to select the most likely ones. However, this agent cannot suppress all false positives that are generated due to the problems discussed in Sect. 3.4 and Sect. 4.4.

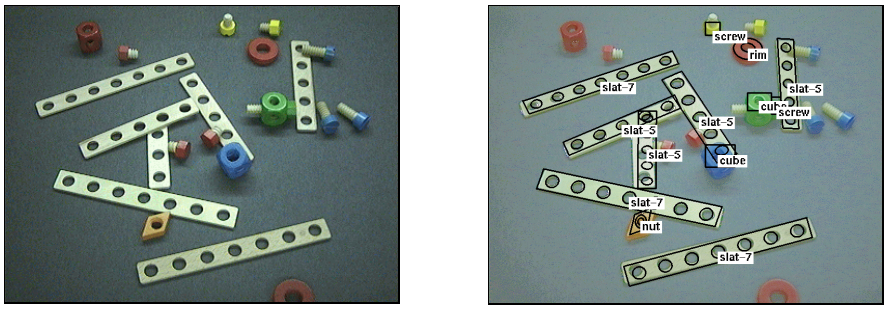
Results for Test Scenes Containing Partially Occluded Objects

Finally, the multi-agent vision system is applied to the set of test images that contain also partially occluded objects. Some recognition results are depicted in Figs. 6.10 – 6.12, which



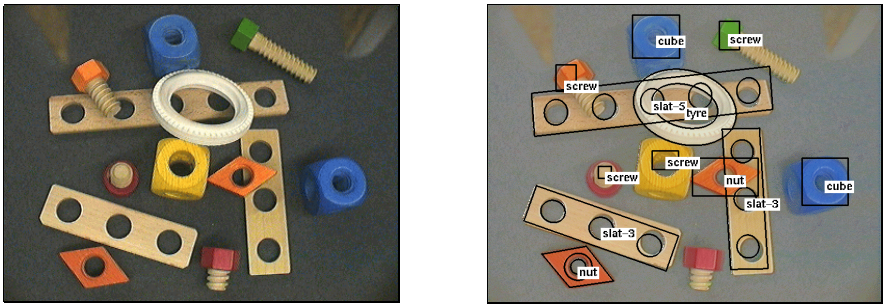
(a) Original image (b) Result

Figure 6.10: DIVA result: occluded objects taken with a top-view camera



(a) Original image (b) Result

Figure 6.11: DIVA result: occluded objects taken with a front-view camera



(a) Original image (b) Result

Figure 6.12: DIVA result: occluded objects taken with a hand-camera

are taken with the top-view, the front-view, and a hand-camera, respectively. As shown, the agent society has still the capability to recognise the objects.

Table 6.5 summarises the recognition results obtained for the occluded test scenes. As expected, the recognition rates are generally lower than the rates that have been achieved for the scenes containing only unoccluded objects. However, the multi-agent vision system performs better than the single FII- or PICA-recognition system. Similar to the case of unoccluded test scenes, the recognition rate for the *nut* has been improved and much less false negatives have been generated.

Table 6.5: Recognition results for unoccluded scenes omitting the hand-camera images

	present	false negatives		false positives		correct	
cube	539	169	(31%)	64	(12%)	313	(58%)
nut	394	78	(20%)	18	(5%)	264	(67%)
rim	202	104	(51%)	18	(9%)	81	(40%)
screw	1032	573	(56%)	30	(3%)	448	(43%)
slat-3	244	147	(19%)	29	(12%)	170	(70%)
slat-5	233	37	(16%)	49	(21%)	155	(67%)
slat-7	207	16	(8%)	61	(30%)	151	(73%)
tyre	142	30	(21%)	0	(0%)	112	(79%)
$\sum$	2993	1154	(39%)	269	(9%)	1694	(57%)

6.4.3 Employing Scripts

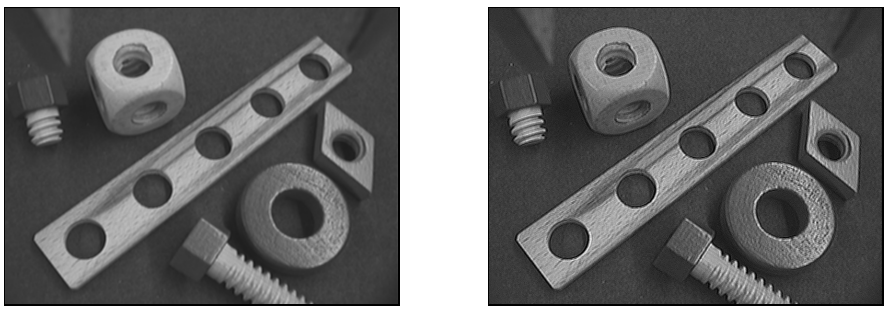
The possibility of passing complex program scripts is another important feature of the proposed DIVA architecture, which is generally not supported by other multi-agent vision approaches. This feature opens up the potentiality to model vision systems that provide some interesting and important properties.

One of the main advantages of permitting the passing of program scripts is that new functionality can be achieved without modifying or rebuilding the agent society. For example, although the implemented *image processing agent* does not know how to enhance an image by sharpening the contrast, the agent provides all elementary functions required to perform such an enhancement. In contrast to other proposed multi-agent vision systems, the new functionality of sharpening images can be achieved by simply sending the script shown in Tab. 6.6 to the *image processing agent* in order to program the agent directly.

Table 6.6: Script for sharpening an image

```
(bind ?image (send [input-message] get-data 0))
(bind ?grey (convert-image ?image grey))
(bind ?laplacian (laplacian-image ?grey))
(bind ?result (sub-image ?grey ?laplacian))
(delete-data ?grey ?laplacian)
(send [output-message] add-data ?result)
```

This program script simply forces the *master image processing agent* to subtract the Laplacian from an input image, which is a well-known method to enhance the contrast of images. The result of employing the script is depicted in Fig. 6.13: Fig. 6.13a shows the input and Fig. 6.13b the sharpen image. As can be seen, especially for the threads of the screws



(a) Original image (b) Result

Figure 6.13: Result of sharpening an image using the script Tab. 6.6

and for the texture of the 5-holed-slat, the structures become much more clear (actually, the script has been applied twice to increase the differences between the images).

Nevertheless, much more complex program scripts containing branches as well as loops are possible. Moreover, the scripts may also contain further requests to other agents, which can be specified by either abstract fact/task descriptions or program scripts.

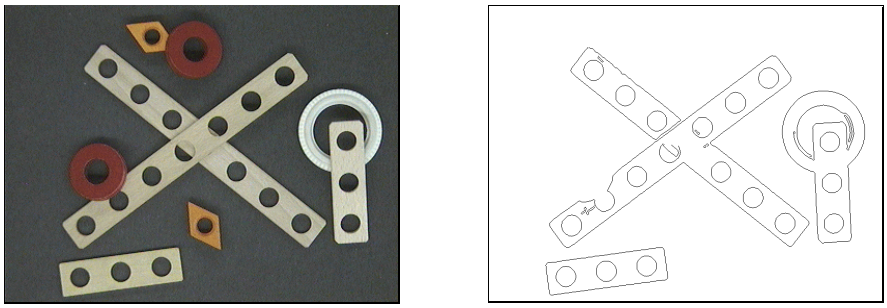
An example for a more complex script, which has minor applicability and is mentioned for demonstration purposes only, is shown in Tab. 6.7. This script can be accomplished by the

Table 6.7: Script for detecting edge points of bright objects

```
; Generate script for image processing agent
;
(make-instance message of GMessage)
(send [message] put-receiver GAgentImageProcessing)
(send [message] put-type SCRIPT)
(send [message] put-text
  "(bind ?src-image (send [input-message] get-data 0))
    (bind ?region-image (get-NULL))
    (if (eq (get-image-type ?src-image) grey)
     then
       (bind ?region-image (threshold-image ?src-image 155))
     else
       (bind ?grey-image (convert-image ?src-image grey))
       (bind ?region-image (threshold-image ?grey-image 155))
       (delete-data ?grey-image)
    )
    (bind ?region2-image (convert-image ?region-image byte))
    (delete-data ?region-image)
    (bind ?edge-image (extract-edge-image ?region2-image
                       5 1.7 G_IMAGE_THRESHOLD_DYNAMIC))
    (delete-data ?region2-image)
    (send [output-message] put-data ?edge-image)"
)
(send [message] put-data (send [input-message] get-data))
(send-message message)

; Perform edge point extraction
;
(bind ?edge-image (send [message] get-data 0))
(bind ?edge-points (extract-edge-points ?edge-image 20))
(delete-data ?edge-image)

; Return result
;
(send [output-message] put-data ?edge-points)
```

feature extraction agent and detects the edge points of the brightest objects in an image. In particular, if this script is passed to the *feature extraction agent* the edge point detection proceeds as follows: First, the *feature extraction agent* generates an additional script which is sent to the *master image processing agent*. This second script forces the *image processing agent* to threshold the input image and to perform an edge operator. However, if it is required, the input image will be first converted into a grey scale image. Next, the *feature extraction agent* uses the generated edge image to extract the edge points, and finally, sends them to the requester. An important point to note here is, that even though it has not been specified explicitly, the *master image processing agent* accomplishes the edge operator with assistance of its slaves to utilise the distributed calculation.

The result of employing this script is shown in Fig. 6.14. As apparent (Fig. 6.14b), only the edge points of the slats and the tyre have been extracted.



(a) Original image (b) Result

Figure 6.14: Result of applying the script of Tab. 6.7

Hence, scripts can be employed by the agents to program each other. Another important use of program scripts would be to teach an agent new functionality. This may be possible by specifying the new script as well as the circumstances under which the program can be applied.

Furthermore, program scripts can also be used to determine or to modify the behaviour of agents at run-time, since it is possible to pass not only single program scripts but also complex knowledge bases, that enable the agents to accomplish diverse tasks which might neither be known nor expected at time of implementation. In order to examine this aspect, the *Clips*-script that defines the knowledge base of the *feature extraction agent* (augmented by a `clear` command) has been sent to a *master image processing agent*. As expected, the modified *master image processing agent* has changed its role and behaves in the following communication processes as a *feature extraction agent*, i.e. the agent has the capability to extract geometric features in images but does not know how to perform image processing tasks any further.

Although the possibility to send complex knowledge bases has not been investigated in more

detail, it obviously provides an enormous amount of flexibility. For example, the agents of a society might be up-dated and expanded at run-time. Furthermore, an agent society might employ a number of 'empty' agents to adapt itself to the requirements and environmental conditions of a particular application. However, for doing so, the agents must be equipped with an extensive set of basic processing functions which are embedded within the *Clips* environment and could be used by the knowledge bases.

7

Conclusions and Future Research

This chapter concludes the thesis. It summarises some of the main results that have been gained and presents directions of possible future research.

7.1 Conclusions

This thesis has addressed the problem of building flexible object recognition systems, that can be integrated in different setups and have the capability to adapt dynamically to different (possibly changing) tasks and environmental conditions. In order to achieve such a flexibility, up-to-date recognition methods have been combined with agent technology in a beneficial way, so that both techniques support each other.

In particular, the thesis has developed two complementary recognition methods based on invariant theory. This theory can be utilised to construct object descriptions that remain constant or at least very stable under the imaging processes of cameras. Therefore, invariant theory can be utilised to recognise objects under various conditions. Especially, once an object has been acquired, it can be recognised in images taken with different cameras from various viewpoints. Furthermore, since invariant descriptions can be restricted to portions of an object, they also enable recognition in presence of partial occlusions.

The first developed recognition method employs geometric-based invariants as object shape descriptions. These descriptions are measured for geometric primitives, such as straight lines and ellipses that have been fitted to the shape of an object, i.e. to its edge points. In contrast to other recognition methods based on geometric invariants the developed one utilises a hypotheses generation method based on fuzzy set theory. As demonstrated, this

approach enhances the commonly used invariant indexing techniques in some respects: First, the developed technique may be used to considerably increase the recognition quality in the difficult case of similar objects; secondly, it expands the applicability to invariant signatures; and lastly, the indexing technique can be extended in closed form, i.e. new (also non-invariant) attributes can be involved in the hypothesis generation technique.

The second recognition method that has been investigated is based on pattern invariants. In contrast to geometric invariants, they are measured for segmented image patches and take the whole image patch information into account. In order to facilitate the generation of object hypotheses, the pattern invariants observed in an image and the invariant representations of acquired objects are compared in a low-dimensional eigenspace by employing principle component analyses (PCA). As opposed to other recognition methods based on PCA, the invariants can be employed to reduce the number of required object representations as well as the number of principle components used to form the eigenspace.

The flexibility of the investigated recognition methods have been further improved by the developed system architecture. In this architecture, a computer vision system is decomposed into a society of autonomous agents, where each agent is responsible for specific tasks, the control strategies are decentralised, and agents communicate using a flexible but easy understandable communication language. This architecture leads to vision systems, that dynamically organise themselves by goal driven-communication processes in order to solve given vision tasks.

As discussed and demonstrated in the experimental results, the developed multi-agent recognition system provides many interesting and important features. For example, (i) the implemented system is robust with respect to breakdowns of single processing modules, (ii) the system is scalable and can be extended by adding agents that implement new functionality, (iii) a vision task is accomplished by employing only those processing modules and resources that are required for the specific task, and (iv) the multi-agent vision architecture permits the passing of complex program scripts which can be utilised to program agents or to modify their behaviour at run-time. Furthermore, it has been shown that the proposed system is suitable to integrate different computer vision algorithms in order to compensate the limits of single methods.

7.2 Future Research

Since this thesis has dealt with the complex problem of building flexible and open computer vision systems, it has had to focus only on a few of the important points. Many interesting problems and questions could not be addressed and should be subject of future work and research. In the following some possible future research directions, that are concerned with computer vision as well as with agent technology, are summarised:

- *Computer vision:*
  The thesis has concentrated on a multi-agent vision system that implements two par-

ticular object recognition methods. This system might be extended and improved in several ways:

First, the recognition capability and applicability of the proposed system might be extended. An interesting extension would be to integrate additional types of invariants to enable the recognition of diverse objects. For example, the possibility to apply the proposed fuzzy invariant indexing technique to complex invariant signatures should be investigated in more detail. Nevertheless, since the recognition methods have been integrated in a single vision system that is based on an open architecture, the recognition performance can further be increased by adding new processing modules, like (i) supplementary recognition methods, (ii) processing modules that fuse the recognition results obtained for different perspective views, or (iii) complex image analyses systems, that validate object hypotheses in great detail. Another very interesting extension would be to involve robotic components in the recognition process. For example, in the setups under consideration (see Fig. 1.1), the robot arms can be utilised to resolve ambiguities or to reduce the complexity of the scene. This could be done, by removing all known objects from the scene, or by grasping an object not yet recognised, so that it can be observed in a pre-defined situation. However, many other types of processing modules might be added to provide new functionality, such as active vision components or 3D scene reconstruction modules.

- *Agent technology:*
  Similar, the proposed multi-agent vision architecture itself provides many starting-points for future work. So, the grammar of the developed communication language should be expanded by allowing additional constraints or information. For example, a specification of the quality of input data such as the illumination conditions of an input image might be exploited to enhance the computed results. Furthermore, the possibility to pass complex program scripts to the agent society can be utilised in many different ways: First, program scripts can be used to teach an agent new functionality. This may be possible by specifying the new script as well as the circumstances under which the program can be applied. In addition, one could also determine the complete behaviour of agents at run-time, because it is generally possible to pass not only single program scripts, but also complex knowledge bases, that enable the agents to accomplish diverse tasks which might neither be known nor expected at time of implementation.

A

Notations

Throughout this thesis a consistent mathematical notation is used. In general vectors are written as small bold type characters, e.g. $\mathbf{x}$, and matrices as capital letters, e.g. $\mathbf{T}$. In order to distinguish between different entities of the same mathematical object, small indices are utilised, e.g. $\mathbf{x_i}$ and $\mathbf{T_i}$. Similar, the components of vectors and matrices are denoted as x_i and T_{ik}, respectively.

The following tables list most of the symbols and their meanings, which are used in this thesis:

Invariant theory (Sect. 2.2)	
G	transformation group
V	vector space
dim G	dimension of transformation group
dim V	dimension of vector space
$\mathbf{g}$	transformation (element) of G
$\mathbf{x}$	vector of V
$\mathbf{x}'$	vector after applying a transformation on $\mathbf{x}$
$\mathbf{T}_{pr}$	projective transformation matrices
$\mathbf{T}_{af}$	affine transformation matrices
$\mathbf{T}_{eu}$	Euclidean transformation matrices
$I(.)$	invariant

Geometric invariants (Sect. 2.3.1)	
$\mathbf{x}$	point in homogeneous coordinates
$\mathbf{l}$	line in homogeneous coordinates
$\mathbf{C}$	conic coefficient matrix in homogeneous coordinates
$\|\mathbf{x}\|$	length of vector
$\|\mathbf{C}\|$	determinant of conic coefficient matrix
$\|[\mathbf{x_1}\ \mathbf{x_2}\ \mathbf{x_3}]\|$	determinant of matrix composed of column vectors $\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$
$\dot{\mathbf{x}}$	first derivative of curve $\mathbf{x}$
$\ddot{\mathbf{x}}$	second derivative of curve $\mathbf{x}$

Pattern invariants (Sect. 2.3.2)	
$p(.)$	image pattern
m_{pq}	regular moment of order p, q
μ_{pq}	central moment of order p, q
$R_{dm}(.)$	Zernike polynomial
Z_{dm}	Zernike moment
$\mathcal{F}(.)$	Fourier transform

Object recognition using geometric invariants (Sect. 3)	
i	invariant measured in an image
$\tilde{I}$	fuzzy invariant object description
$\tilde{O}$	object class modelled as fuzzy singleton
$\mu_{\tilde{I}}(.)$	membership function of invariant description $\tilde{I}$
μ	membership value
$D(.)$	dissimilarity measure

Object recognition using pattern invariants (Sect. 4)	
$p(.)$	segmented 2D image patch normalised in size
$\mathcal{F}_T(.)$	2D Fourier-transform of $p(.)$
$\|\mathcal{F}_T(.)\|$	translation and scale invariant power spectrum
$\mathcal{F}_R(.)$	1D Fourier-transform of $\|\mathcal{F}_T(.)\|$
$\|\mathcal{F}_R(.)\|$	rotation, translation and scale invariant power spectra
$\mathbf{i}$	vector representation of Fourier pattern invariant
$\boldsymbol{\mu}$	mean of pattern invariant set
$\mathbf{C}$	covariance matrix of pattern invariant set
$\mathbf{T}$	transformation matrix that projects pattern invariants into eigenspace

B

Baufix Object Domain

This appendix introduces the object domain which has been used to obtain the experimental results of the geometric-based FII- and of the appearance-based PICA-recognition system. The domain is given by the wooden toy objects of the Baufix construction kit, which can be employed to assemble complex objects, such as airplanes or motor cycles. In the following, the objects of the Baufix domain are explained in more detail, where the objects have been categorised in six different classes:

- *Cubes:*
  The *cubes* of the Baufix domain have rounded edges and corners. Furthermore, as shown in Fig. B.1, each side of a cube has a circular hole, which goes right through the object. The cubes are coloured in red, green, blue, and yellow.

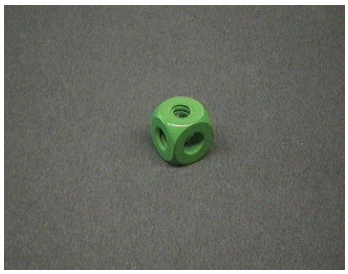


Figure B.1: Cube

Although the six sides of the cubes can be modelled by planar geometric structures of single ellipses and four straight lines, they have not been acquired in the FII-recognition system: Due to the rounded edges, the feature extraction stage of the system often fails to extract the geometric structures reliably. Depending on the viewpoint, the result of the feature extraction process will be either a single ellipse and four lines or a pair of ellipses, so that cubes cannot be recognised by the FII-system robustly. Contrary, since the Fourier invariants employed in the PICA-recognition system can be measured very stable, cubes are recognised by the PICA-recognition system.

- *Nuts:*
  As shown in Fig. B.2, the nuts are orange, diamond-shaped objects which have threaded holes.

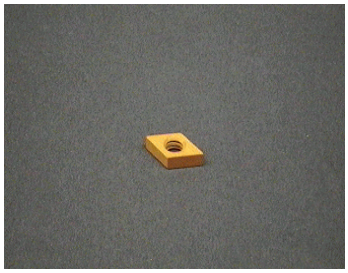


Figure B.2: Nut

Similar to the cubes, nuts can be modelled by a geometric structure of an ellipse and four straight lines. Since nuts have sharp edges, this geometric structure can be extracted reliably. So, nuts are recognised by both recognition systems.

- *Rims:*
  The *rim* is a flat, red coloured cylindrical object, which has a boring.

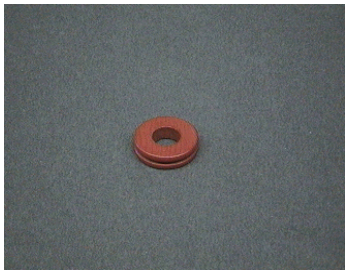


Figure B.3: Rim

This object is recognised by the FII-recognition system by employing the geometric invariants of a pair of coplanar ellipses. It must be noted, that these invariants are

in this case global object descriptors. However, ellipses can be appropriately fitted, even if only about 50% of the edge points are visible. Therefore, rims can also be recognised in presence of partial occlusion.

- *Screws:*
  The Baufix domain contains several types of screws, which differ in their length as well as in the shape of their heads (round and hexagonal). Furthermore, the heads of the screws are painted in different colours, where the colours encode the length of the threads.

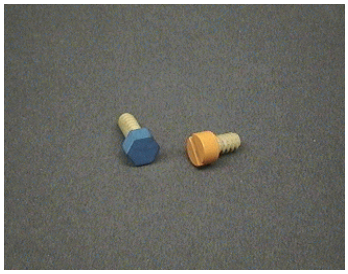


Figure B.4: Screws

Although the screw heads might be modelled by planar geometric structures, they are not recognised by the FII-system, because the feature extraction stage would often fail to detect them correctly (for the round screw heads, however, it is impossible to construct any projective invariant, because the geometric features that can be extracted do not meet the counting argument of invariant theory). Thus, screws are only recognised by the PICA-recognition system.

- *Slats:*
  As shown in Fig. B.5, three different types of slats are provided, which differ in the number of holes as well as in their length. These slats are named: *3-holed-slat*, *5-holed-slat*, and *7-holed-slat*, respectively.

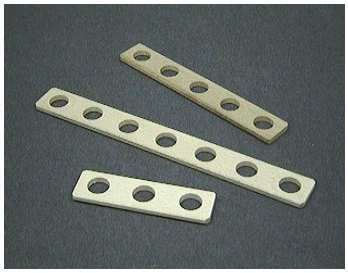


Figure B.5: Slats

The slats are recognised by the FII-recognition system and are modelled as four straight lines and three, five, or seven ellipses.

- *Tyres:*
  The *tyres* of the object domain are white, rubber-made cylindrical objects (Fig. B.6).

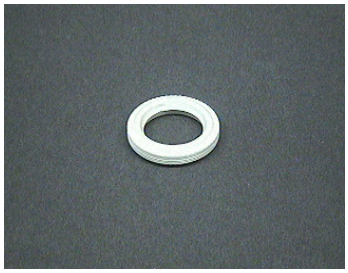


Figure B.6: Tyre

Similar to the rim, the tyres are modelled by the geometric structure of a pair of ellipses and are recognised by the FII-system solely.

C

The Fuzzy Rulebase

This appendix presents the fuzzy if-then classification rules that have been utilised in the FII-recognition system (Sect. 3.3) to perform the recognition process. As described in Sect. 3.2.1, these rules share the general structure:

$$\begin{aligned} &\text{IF } i^k_{m1} = \widetilde{I}^k_{m1} \text{ AND } \cdots \text{ AND } i^k_{mN^k_m} = \widetilde{I}^k_{mN^k_m} \text{ THEN } o^k_m = \widetilde{O}^k \\ &\qquad k = 1, 2, \ldots, K \qquad \text{(object classes)} \\ &\text{with} \quad m = 1, 2, \ldots, M^k \qquad \text{(sub-rules for class } k\text{)} \\ &\qquad n = 1, 2, \ldots, N^k_m \qquad \text{(conditions of a sub-rule)} \end{aligned}$$

where the membership functions of the fuzzified invariant values $\widetilde{I}^k_{mn}$ are determined by the bell-shaped functions $\mu_{\widetilde{I}^k_{mn}}$:

$$\mu_{\widetilde{I}^k_{mn}}(u) = e^{-\frac{(u-\alpha^k_{mn})^2}{2\beta^{k\,2}_{mn}}}, \quad u \in \mathbb{R}$$

In order to provide a deeper insight in the developed FII-technique the following lists show (i) the acquired fuzzy classification rules in human readable form, and (ii) the parameters α^k_{mn} and β^k_{mn} that determine the specific shapes of the corresponding fuzzified invariant values. All these rules have been automatically generated using 30 different training images per object, of which some of the rules have been manually adjusted to improve the recognition performance, e.g. the parameters of the fuzzified invariant values of the *slats* have been equalised. The rules are based on the two functional independent invariants of a pair of coplanar conics (2.5), (2.6) as well as on the three invariants of the geometric structure of a single ellipse and three straight lines (2.8) – (2.10), where the invariants are denoted as I^C_1, I^C_2, I^L_1, I^L_2, and I^L_3, respectively.

Table C.1: Fuzzy classification rules

1.	IF	$(I_1^L \approx 1.3)$	AND	$(I_2^L \approx 1.8)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS nut)
2.	IF	$(I_1^L \approx 1.4)$	AND	$(I_2^L \approx 3.2)$	AND	$(I_3^L \approx 2.5)$	THEN	(object IS nut)
3.	IF	$(I_1^L \approx 2.3)$	AND	$(I_2^L \approx 9.2)$	AND	$(I_3^L \approx 5.5)$	THEN	(object IS nut)
4.	IF	$(I_1^C \approx 3.8)$	AND	$(I_2^C \approx 4.1)$			THEN	(object IS rim)
5.	IF	$(I_1^L \approx 1.5)$	AND	$(I_2^L \approx 1.5)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-3)
6.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 1.1)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-3)
7.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-3)
8.	IF	$(I_1^L \approx 2.1)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-3)
9.	IF	$(I_1^L \approx 1.5)$	AND	$(I_2^L \approx 1.5)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-5)
10.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 1.1)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-5)
11.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-5)
12.	IF	$(I_1^L \approx 2.1)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-5)
13.	IF	$(I_1^L \approx 1.5)$	AND	$(I_2^L \approx 1.5)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-7)
14.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 1.1)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-7)
15.	IF	$(I_1^L \approx 1.6)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 1.6)$	THEN	(object IS slat-7)
16.	IF	$(I_1^L \approx 2.1)$	AND	$(I_2^L \approx 4.3)$	AND	$(I_3^L \approx 2.1)$	THEN	(object IS slat-7)
17.	IF	$(I_1^C \approx 3.3)$	AND	$(I_2^C \approx 3.3)$			THEN	(object IS tyre)

Table C.2: Parameters α_{mn}^k and β_{mn}^k of fuzzified membership functions $\mu_{\tilde{I}_{mn}^k}$

rule	I_1^C		I_2^C		I_1^L		I_2^L		I_3^L		object
	α_{m1}^k	β_{m1}^k	α_{m2}^k	β_{m2}^k	α_{m3}^k	β_{m3}^k	α_{m4}^k	β_{m4}^k	α_{m1}^k	β_{m1}^k	
1	–	–	–	–	1.25	0.2	1.83	0.25	1.59	0.35	nut
2	–	–	–	–	1.4	0.09	3.2	0.15	2.5	0.14	nut
3	–	–	–	–	2.3	0.25	9.15	0.25	5.5	0.45	nut
4	3.77	0.14	4.11	0.16	–	–	–	–	–	–	rim
5	–	–	–	–	1.54	0.1	1.49	0.1	2.05	0.1	slat-3
6	–	–	–	–	1.55	0.1	0.11	0.1	1.57	0.1	slat-3
7	–	–	–	–	1.55	0.1	4.25	0.23	1.55	0.1	slat-3
8	–	–	–	–	2.05	0.1	4.25	0.23	2.05	0.1	slat-3
9	–	–	–	–	1.54	0.1	1.49	0.1	2.05	0.1	slat-5
10	–	–	–	–	1.55	0.1	0.11	0.1	1.57	0.1	slat-5
11	–	–	–	–	1.55	0.1	4.25	0.23	1.55	0.1	slat-5
12	–	–	–	–	2.05	0.1	4.25	0.23	2.05	0.1	slat-5
13	–	–	–	–	1.54	0.1	1.49	0.1	2.05	0.1	slat-7
14	–	–	–	–	1.55	0.1	0.11	0.1	1.57	0.1	slat-7
15	–	–	–	–	1.55	0.1	4.25	0.23	1.55	0.1	slat-7
16	–	–	–	–	2.05	0.1	4.25	0.23	2.05	0.1	slat-7
17	3.26	0.65	3.31	0.08	–	–	–	–	–	–	tyre

D

Agent Knowledge

The knowledge of the autonomous agents, which are part of the proposed multi-agent vision system described in Sect. 6.3, has been modelled using the expert system tool *Clips 6.10* [CLIPS 1998]. This tool is provided as a complex C-library, that implements a *LISP*-style programming language as well as an efficient inference engine, which can be completely controlled by C/C++-programs. Furthermore, *Clips* can be extended by embedding user-defined C-functions within its environment.

Although, this appendix cannot consider the knowledge representation in great detail, some aspects are mentioned to give an impression about the provided high-level knowledge which enables the agents to react to requested tasks.

In general, the knowledge of an agent is stored in facts and in rules. While facts are utilised to store information about different entities such as real world objects, rules mainly determine the behaviour of an agent, i.e. how an agent react in a specific situation.

Examples for facts, which have been stored in the knowledge base of the *recognition fusion agent*, are listed below:

```
(is-a            rim           object)
(has-name        rim           rim)
(has-color       rim           red)

(is-a            blue-screw    object)
(has-name        blue-screw    screw)
(has-name        blue-screw    blue-screw)
(has-color       blue-screw    blue)
```

This short excerpt determines the information about two different objects of the Baufix domain, namely the *rim* and the blue *cube*. As indicated, the facts specify the particular types of entities, their names as well as their properties and attributes. There are two important points to note here: (i) entities can be identified not only by their unambiguous names but also by their features and attributes (like the colour attribute) and (ii) the provided knowledge about an entity can be easily augmented through further facts.

An example for a simple rule, which also has been stored in the *recognition fusion agent*, is given by:

```
; ------------------------------
; makeProgramGetImageFromMessage
; ------------------------------
; If the source of an image is the input-message, this rule adds
; a get-image operation to the program, if the source is the
(defrule makeProgramGetImageFromMessage
   ?f0<-(object (is-a GProgramOperationRequest)
                (program        ?prg)
                (operation      get ?dest ?src)
                (operation-code get ?var-dest ?))
   (is-a ?src image)
   (has-source ?src message)
   (has-index ?src ?index)
   =>

   ; Generate operation get-data-from-message
   (bind ?var-src (sym-cat var- (gensym*)))
   (bind ?operation

   )
   (send ?prg insert-operation 1
         "(bind ?" ?var-dest " (send [input-message] get-data "
         ?index "))"
   )

   ; Retract old operation request
   (send ?f0 delete)
)
```

This rule might be used during the generation process of program scripts, that are suitable to perform requested tasks. In particular, it adds a program line which fetches an image from the input message.

Such rules are employed to determine the behaviour of an agent. Several types of rules are provided, e.g. rules are used to analyse a message text, to generate program scripts (as the one mentioned above), and to perform memory management. Similar to the facts, rules can be easily added in order to provide new functionality.

E

Interpretation of Abstract Task Descriptions

This appendix explains the syntax and the semantics of abstract task descriptions which are specified using the proposed communication language (see Sect. 6.2.2) in more detail. This is done, by describing how master agents analyse such abstract descriptions in order to be able to react to the requested tasks adequately.

In particular, the interpretation process will be described on the basis of a recognition task that can be accomplished by the *recognition fusion agent*. This agent provides the communication language determined by the formal grammar, shown in Tab. E.1, which is stored as facts and rules in the knowledge base of the agent (Appendix D). Since the *recognition fusion agent* has the capability to perform recognition tasks only, the provided communication language is relative simple.

A typical recognition task which has been already mentioned in Sect. 6.4.1 is given by:

```
(recognise ?dest ?src)

(is-a      ?dest object)
(or (has-name ?dest cube) (has-color ?dest wooden))

(is-a      ?src  image)
(has-source ?src message) (has-index ?src 0)
```

This task requests the agent society to recognise all *cubes* as well as all wooden-coloured objects in the image attached to the message.

Table E.1: Communication language provided by the *recognition fusion agent*

```
<abstract-description> ::= (<goal> <variable>*) <goal-condition>*

<goal>                 ::= recognise | recognize
<goal-condition>       ::= <attr-condition> | <not-condition> |
                           <and-condition>  | <or-condition>

<attr-condition>       ::= <is-a-attr>            | <has-index-attr  |
                           <has-name-attr>        | <has-color-attr> |
                           <has-no-of-holes-attr> | <has-source-attr>
<not-condition>        ::= (not <goal-condition>)
<and-condition>        ::= (and <goal-condition>+)
<or-condition>         ::= (or  <goal-condition>+)

<is-a-attr>            ::= (is-a            <variable> <class-spec>)
<has-color-attr>       ::= (has-color       <variable> <color-spec>)
<has-index-attr>       ::= (has-index       <variable> <number>+)
<has-name-attr>        ::= (has-name        <variable> <name-spec>)
<has-no-of-holes-attr> ::= (has-no-of-holes <variable> <no-of-holes-spec>)
<has-source-attr>      ::= (has-source      <variable> <source-spec>)

<class-spec>           ::= image | object
<color-spec>           ::= blue | green  | orange | red | white | wood | wooden |
                           yellow
<name-spec>            ::= cube | blue-cube | green-cube | red-cube |
                           yellow-cube | ledge | ledge-3 | ledge-5 | ledge-7 |
                           nut | rim | screw | blue-screw | green-screw |
                           red-screw | yellow-screw | slat | slat-3 | slat-5 |
                           slat-7 | tyre
<number-of-holes-spec> ::= 3 | 5 | 7
<source-spec>          ::= message

<variable>             ::= ?  <alpha> {<alpha> | <number>}

<alpha>                ::= a | b | c | d | e | f | g | h | i | j | k | l | m |
                           n | o | p | q | r | s | t | u | v | w | x | y | z
<number>               ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

In general, the interpretation process of a message proceeds in the following way: First, the master agents which have received the message analyse the global goal of the requested task. In this case, the goal is to recognise a destination denoted by variable `?dest` from a source denoted by `?src`. If an agent, like the *recognition fusion agent*, knows how to perform a recognition task, it examines the abstract description in further detail: The agent extracts all task specifications that are related to the destination variable `?dest` and determines all entities stored in its knowledge base (see Appendix D) that match the given specifications. The inference engine of the agent performs the required search as a pattern matching process. For example, the above destination specifications can be unified with the following facts stored in the knowledge base of the *recognition fusion agent*:

```
(is-a      cube-red  object)
(has-name  cube-red  cube)
```

That means, the agent is capable of recognising the entity `cube-red`. Similar, the *recognition fusion agent* filters all other types of *cubes* as well as the (wooden-coloured) *slats*.

In the developed multi-agent vision architecture it is sufficient that an agent understands the global goal as well as the destination specifications only. For example, although the *master FII-recognition agent* does not understand the source specifications, which simply specify that the objects should be recognised in the image (`is-a ?src image`) stored in the first data slot (`has-index ?src 0`) of the message (`has-source ?src message`), the *master FII-recognition agent* is able to accomplish the recognition task correctly.

Generally, the agents generate program scripts in order to solve requested tasks. If the agents do not understand the source specifications, i.e. no corresponding entities have been found, these scripts include further message passing to other agents. In such cases the agents request the society to generate a required sub-result from the unknown source. However, if an agent understands the source specifications it will use the provided information to trigger the program script.

It must be noted, that the grammar of the proposed communication language is not static but can be easily expanded in order to adapt the communication language to different requirements. For example, it might be possible and useful to permit the specification of the quality of the input image:

```
(has-quality  ?src  noisy)
```

so that the *master image processing agent* knows that smoothing filters should be applied to the input image in order to enhance the recognition results.

Bibliography

[AITCHISON 1986] J. Aitchison. *The statistical analysis of compositional data*. Chapman and Hall, 1986.

[ARNOLD 1990] V.I. Arnold. *Geometrical methods in the study of ordinary differential equations*. Grundlehren der Mathematischen Wissenschaft. Springer, 1990.

[BEN-ARIE et al. 1996a] J. Ben-Arie, Z. Wang, and K.R. Rao. Iconic representation and recognition using affine-invariant spectral signatures. In *Proc. DARPA Image Understanding Workshop (IUW'96), Palm Springs, CA, USA*, pages 1277–1285, 1996.

[BEN-ARIE et al. 1996b] J. Ben-Arie, Z. Wang, and K.R. Rao. Iconic representation with affine-invariant spectral signatures. In *Proc. International Conference on Pattern Recognition (ICPR'96), Vienna, Austria*, pages 672–676, 1996.

[BIANCHI and RILLO 1997] R.A.C. Bianchi and A.H.R.C. Rillo. A purposive vision system: a multi-agent approach. In *Workshop on Cybernetic Vision 1996, Proc. IEEE Computer Society*, pages 225–230, 1997.

[BINFORD and LEVITT 1993] T.O. Binford and T.S. Levitt. Quasi-invariants: theory and exploitation. In *Proc. DARPA Image Understanding Workshop (IUW'93), Washington, DC, USA*, pages 819–829, 1993.

[BOISSIER and DEMAZEAU 1994a] O. Boissier and Y. Demazeau. ASIC: an architecture for social and individual control and its application to computer vision. In *Proc. European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW'94), Odense, Denmark*, pages 107–118, 1994.

[BOISSIER and DEMAZEAU 1994b] O. Boissier and Y. Demazeau. MAVI: A multi-agent system for visual integration. In *Proc. IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems, Las Vegas, NE, USA*, pages 731–738, 1994.

[BOND and GASSER 1988] A. Bond and L. Gasser, editors. *Readings in distributed artificial intelligence*. Morgan Kaufmann, Los Angeles, CA, 1988.

[BOOKSTEIN 1979] F. Bookstein. Fitting conic sections to scattered data. *Computer Vision Graphics and Image Processing*, 9:56–71, 1979.

[BRILL et al. 1992] M.H. Brill, E.B. Barrett, and P.M. Payton. Projective invariants for curves in two and three dimensions. In [MUNDY and ZISSERMAN 1992], pages 193–214. Springer, 1992.

[BROWN 1992] C.M. Brown. Numerical evaluation of differential and semi-differential invariants. In [MUNDY and ZISSERMAN 1992], pages 215–227. Springer, 1992.

[BURNS et al. 1992] J.B. Burns, R.S. Weiss, and E.M. Riseman. The non-existence of general-case view-invariants. In [MUNDY and ZISSERMAN 1992], pages 120–131. Springer, 1992.

[CANNY 1983] J.F. Canny. A variational approach to edge detection. In *Proc. National Conference on Artificial Intelligence (AIII'83), Washignton, DC, USA*, pages 54–58, 1983.

[CARLSSON 1992] S. Carlsson. Projectively invariant decomposition of planar shapes. In [MUNDY and ZISSERMAN 1992], pages 267–273. Springer, 1992.

[CARLSSON 1994] S. Carlsson. The double algebra: an effective tool for computing invariants in computer vision. In [MUNDY et al. 1994], pages 145–164. Springer, 1994.

[CASASENT and PSALTIS 1976] D. Casasent and D. Psaltis. Position, rotation, and scale-invariant optical correlation. *Applied Optics*, 15:1795–1799, 1976.

[CLIPS 1998] *CLIPS Reference Guide, Volume I, Basic Programming Guide*. Artificial Intelligence Section, Lyndon B. Johnson Space Center, 1998.

[CORKILL 1991] D.D. Corkill. Blackboard systems. *AI Expert*, 9(91):41–47, 1991.

[DAVOLI and TAMBURINI 1993] R. Davoli and F. Tamburini. Data algorithm: a numerical method to extract shape information from gray scale images. Technical Report UBLCS-93-15, Laboratory for Computer Science, University of Bologna, 1993.

[DAVOLI et al. 1999] R. Davoli, F. Tamburini, and R. Gaioni. A translation, rotation and scale invariant transform for grey scale images: a parallel implementation. In *Proceeding of SGI/Cray MPP Workshop, Bologna, Italy*, 1999.

[DENZLER et al. 1995] J. Denzler, B. Heigl, and D. Paulus. Farbsegmentierung für Aktives Sehen. In *Workshop Farbbildverarbeitung, Volume 15 of Fachberichte Informatik, V. Rehrmann (eds.)*, pages 9–12, 1995.

[FAUGERAS 1993] O.D. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, 1993.

[FONGA 1996] H. Fonga. Pattern recognition in gray-level images by Fourier analysis. *Pattern Recognition Letters*, 17:1477–1489, 1996.

[FORSYTH 1990] D.A. Forsyth. A novel algorithm for color constancy. *International Journal of Computer Vision*, 5:5–36, 1990.

[FORSYTH et al. 1990] D.A. Forsyth, J.L. Mundy, A. Zisserman, and C.M. Brown. Projectively invariant representations using implicit algebraic curves. In *Proc. European Conference on Computer Vision (ECCV'90), Antibes, France*, pages 427–436, 1990.

[FRIGO and JOHNSON 1999] M. Frigo and S.G. Johnson, Frigo, M. and Johnson, S.G. *FFTW Users's Manual*. Massachusetts Institue of Technology, 1999.

[GRAF and KNOLL 1999a] T. Graf and A. Knoll. Driving vision systems by communication. In *Proc. IEEE International Conference on Information, Intelligence, and Systems (ICIIS'99), Washington, DC, USA*, pages 640–645, 1999.

[GRAF and KNOLL 1999b] T. Graf and A. Knoll. A multi-agent approach to self-organizing vision systems. In *Proc. Asia-Pacific Conference on Intelligent Agent Technology (IAT'99), Hong Kong, China*, pages 48–52, 1999.

[GRAF and KNOLL 2000] T. Graf and A. Knoll. A multi-agent system architecture for distributed computer vision. *International Journal on Artificial Intelligence Tools (IJAIT)*, 9(2):305–319, 2000.

[GRAF et al. 1998a] T. Graf, A. Knoll, and A. Wolfram. Fuzzy invariant indexing: A general indexing scheme for occluded object recognition. In *Proc. IEEE International Conference on Signal Processing (ICSP'99), Beijing, China*, pages 908–911, 1998.

[GRAF et al. 1998b] T. Graf, A. Knoll, and A. Wolfram. A fuzzy invariant indexing technique for object recognition under partial occlusion. In *Proc. International Workshop on Fuzzy-Neuro Systems (FNS'98), Munich, Germany*, pages 266–273, 1998.

[GRAF et al. 1998c] T. Graf, A. Knoll, and A. Wolfram. Recognition of partially occluded objects through fuzzy invariant indexing. In *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'98), Anchorage, AL, USA*, pages 1566–1571, 1998.

[HEIDEMANN 1998] G. Heidemann. *Ein flexible einsetzbares Objekterkennungssystem auf Basis neuronaler Netze*. PhD thesis, Universität Bielefeld, Technische Fakultät, 1998.

[HEISTERKAMP and BHATTACHARYA 1996] D.R. Heisterkamp and P. Bhattacharya. Invariants of families of coplanar conics and their applications to object recognition. In *Proc. International Conference on Pattern Recognition (ICPR'96), Vienna, Austria*, pages 677–681, 1996.

[HO and CHEN 1995] C. Ho and L. Chen. A fast ellipse and circle detector using geometric symmetry. *Pattern Recognition*, pages 117–124, 1995.

[HOTELLING 1993] H. Hotelling. Analysis of a complex of statistical variables into principle components. *The journal of educational psychology*, 1993.

[HU 1962] M.K. Hu. Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8:179–187, 1962.

[HURWITZ 1897] A. Hurwitz. Über die Erzeugung der Invarianten durch Integration. *Nachr. Akad. Wissenschaft Göttingen*, page 71, 1897.

[HUTTENLOCHER 1988] D.P. Huttenlocher. *Three-dimensional recognition of solid objects from a two dimensional image*. PhD thesis, Department of Electrical Engineering and Computer Science, 1988.

[KERRE and NACHTEGAEL 2000] E.E. Kerre and M. Nachtegael, editors. *Fuzzy Techniques in Image Processing*. Springer, 2000.

[KLIR and YUAN 1995] G. Klir and B. Yuan. *Fuzzy sets and fuzzy logic - theory and applications*. Prentice-Hall, 1995.

[KNOLL et al. 2000] A. Knoll, J. Zhang, T. Graf, and A. Wolfram. Recognition of occluded objects and visual servoing: two case studies of employing fuzzy techniques in robot vision. In [KERRE and NACHTEGAEL 2000]. Springer, 2000.

[KRÖNER and SCHULZ-MIRBACH 1995] S. Kröner and H. Schulz-Mirbach. Fast adaptive calculation of invariant features. In *DAGM-Symposium, Bielefeld, Germany*, pages 23–35. Springer, 1995.

[LAMDAN et al. 1988] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Proc. Computer Vision and Pattern Recognition (CVPR'88)*, pages 335–344, 1988.

[LAMDAN and WOLFSON 1988] Y. Lamdan and H.J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Proc. International Conference on Computer Vision (ICCV'98), Bombay, India*, pages 238–249, 1988.

[LUONG 1993] Q.-T. Luong. Color in computer vision. In C.H. Chen, L.F. Pau, and P.S.P. Wang, editors, *Handbook of pattern recognition and computer science*, pages 311–368. World Scientific, 1993.

[MÜLLER 1993] J. Müller, editor. *Verteilte Künstliche Intelligenz - Methoden und Anwendungen*. Wissenschaftsverlag, 1993.

[MÜLLER et al. 1997] J. Müller, M. Wooldridge, and N. Jennings, editors. *Intelligents Agents III. Agent theories, architectures, and languages*. Springer, 1997.

[MUNDY et al. 1992] J.L. Mundy, D. Kapur, S.J. Maybank, P. Gros, and L. Quan. Geometric interpretation of joint conic invariants. In [MUNDY and ZISSERMAN 1992], pages 87–104. Springer, 1992.

[MUNDY et al. 1996] J.L. Mundy, A. Liu, N. Pillow, A. Zisserman, S. Abdallah, S. Utcke, S.K. Nayar, and C.A. Rothwell. An experimental comparison of appearance and geometric model based recognition. In *Proc. International Workshop on Object Representation in Computer Vision, Cambridge, UK*, pages 247–269, 1996.

[MUNDY and ZISSERMAN 1992] J.L. Mundy and A. Zisserman, editors. *Geometric invariance in computer vision*. MIT Press, 1992.

[MUNDY et al. 1994] J.L. Mundy, A. Zisserman, and D.A. Forsyth, editors. *Applications of invariance in computer vision*. Springer-Verlag, 1994.

[MURASE and NAYAR 1995] H. Murase and S.K. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 15:5–24, 1995.

[OLSON 1994] C.F. Olson. Probabilistic indexing: A new method of indexing 3D data from 2D image data. In *CAD-Based Vision Workshop*, pages 2–8, 1994.

[OLSON 1995] C.F. Olson. Probabilistic indexing for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(5):518–522, 1995.

[PENTLAND et al. 1994] A. Pentland, B. Moghaddam, and T. Starner. Viewbased and modular eigenspaces for face recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR'94), Seatle, WA, USA*, pages 84–91, 1994.

[QUAN et al. 1992] L. Quan, P. Gros, and R. Mohr. Invariants of a pair of conics revisited. *Image and Vision Computing*, 10(5):319–323, 1992.

[RICKHEIT and WACHSMUTH 1996] G. Rickheit and I. Wachsmuth. Collaborative Research Center "Situated Artifical Communicators" at the University of Bielefeld, Germany. *Artificial Intelligence Review*, 10:165–170, 1996.

[RIVLIN and WEISS 1993] E. Rivlin and I. Weiss. Semi-local invariants. In *Proc. Computer Vision and Pattern Recognition (CVPR'93), New York City, NY, USA*, pages 697–698, 1993.

[ROTHWELL 1995a] C.A. Rothwell. The importance of reasoning about occlusions during hypothesis verification in object recognition. Technical Report 2673, INRIA, 1995.

[ROTHWELL 1995b] C.A. Rothwell. *Object recognition through invariant indexing*. Oxford University Press, 1995.

[ROTHWELL 1996] C.A. Rothwell. Reasoning about occlusion during hypothesis verification. In *Proc. European Conference on Computer Vision (ECCV'96), Cambridge, UK*, pages 599–609, 1996.

[ROTHWELL et al. 1993] C.A. Rothwell, D.A. Forsyth, A. Zisserman, and J.L. Mundy. Extracting projective structure from single perspective views of 3D point sets. In *Proc. International Conference on Computer Vision (ICCV'93), Berlin, Germany*, pages 573–582, 1993.

[ROTHWELL et al. 1994] C.A. Rothwell, J.L. Mundy, B. Hoffman, and V.-D. Nguyen. Driving vision by topology. Technical Report 2444, INRIA, 1994.

[ROTHWELL et al. 1995] C.A. Rothwell, J.L. Mundy, W. Hoffman, and V.-D. Nguyen. Driving vision by topology. In *Proc. IEEE International Symposium on Computer Vision, Coral Gables, FL, USA*, pages 395–40, 1995.

[ROTHWELL and STERN 1995] C.A. Rothwell and J. Stern. Unterstanding the shape properties of trihedral polyhedra. Technical Report 2661, INRIA, 1995.

[ROTHWELL and STERN 1996] C.A. Rothwell and J. Stern. Understanding the shape properties of trhedral polyhedra. In *Proc. European Conference on Computer Vision (ECCV'96), Cambridge, UK*, pages 599–609, 1996.

[ROTHWELL et al. 1992] C.A. Rothwell, A. Zisserman, D.A. Forsyth, and J.L. Mundy. Canonical frames for planar object recognition. In *Proc. European Conference on Computer Vision (ECCV'92), Santa Margherita Ligure, Italy*, pages 757–772, 1992.

[SCHEERING 2000] C. Scheering. *Multi-Agenten in einem Situierten Künstlichen Kommunikator*. PhD thesis, Universität Bielefeld, Technische Fakultät, 2000.

[SCHULZ-MIRBACH 1994] H. Schulz-Mirbach. Constructing invariant features by averaging techniques. In *Proc. International Conference on Pattern Recognition (ICPR'94)*, pages 387–390, 1994.

[SCHULZ-MIRBACH 1995] H. Schulz-Mirbach. Invariant features for grey scale images. In *DAGM-Symposium, Bielefeld, Germany*, pages 1–14. Springer, 1995.

[SPRINGER 1964] C.E. Springer. *Geometry and analysis of projective spaces*. W.H. Freeman and Company, 1964.

[VAN GOOL et al. 1992] L.J. Van Gool, T. Moons, E. Pauwels, and A. Oosterlinck. Semi-differential invariants. In [MUNDY and ZISSERMAN 1992], pages 135–156. Springer, 1992.

[VUURPIJL and SCHOMAKER 1998a] L. Vuurpijl and L. Schomaker. A framework for using multiple classifiers in a multiple-agent architecture. In *Proc. European Workshop on Handwriting Analysis and Recognition, In: The Institution of Electrical Engineers, Digest Number 1998/440*, pages 8/1–8/6, 1998.

[VUURPIJL and SCHOMAKER 1998b] L. Vuurpijl and L. Schomaker. Multiple-agent architectures for the classification of handwritten text. In *Proc. International Workshop on Frontier of Handwriting Recognition, Taejon, Korea*, pages 335–346, 1998.

[WEISS 1992] I. Weiss. Noise resistent projective and affine invariants. In *Proc. DARPA Image Understanding Workshop (IUW'92), San Diego, CA, USA*, pages 683–692, 1992.

[WEYL 1946] H. Weyl. *The classical groups and their invariant*. Princeton University Press, 1946.

[WILCZYNSKI 1906] E.J. Wilczynski. *Projective differential geometry of curves and ruled surfaces*. Teubner, Leipzig, 1906.

[WOLFSON 1990] H.J. Wolfson. Model-based object recognition by geometric hashing. In *Proc. European Conference on Computer Vision (ECCV'90), Antibes, France*, pages 526–536, 1990.

[WOOD 1996] J. Wood. Invariant pattern recognition: a review. *Pattern Recognition*, 29(1):1–17, 1996.

[YANAI 1999] K. Yanai. An image understanding system for various images based on multi-agent archtitecture. In *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'99), New Delhi, India*, 1999.

[YANAI and DEGUCHI 1998] K. Yanai and K. Deguchi. An architecture of object recognition system for various images based on multi-agent. In *Proc. International Conference on Pattern Recognition (ICPR'98), Brisbane, Australia*, pages 278–281, 1998.

[ZISSERMAN et al. 1994] A. Zisserman, D.A. Forsyth, J.L. Mundy, C.A. Rothwell, J. Liu, and N. Pillow. 3D object recognition using invariance. Technical Report OUEL 2027/94, Robotic Research Group, University of Oxford, 1994.

Author Index

Subject Index