Włodzimierz Kasprzak

Institute of Control and Computation Engineering
Warsaw University of Technology

ADAPTIVE COMPUTATION METHODS IN DIGITAL IMAGE SEQUENCE ANALYSIS

Manuscript received 12.10.2000

In this work *adaptive computation* techniques are applied for *digital image sequence analysis*, organized by a hierarchy of data abstraction levels (ranging from the signal level over the iconic, segmentation and object recognition levels to the cognitive level). The applied methodology focuses on the biologically-motivated *connectionism* theory (i.e. artificial neural networks and semantic networks) and on *dynamic system* theory (tracking, recursive estimation). The meaning of the term *"adaptive"* in this work is twofold and it corresponds to two modes of image sequence analysis. In the case of a *limited-length* image sequence, called a *N-image*, a stationary scene (environment) is usually expected – a *batch* analysis mode is defined. For unlimited-length image sequences (in general a sequence of *N-images*) a non-stationary scene environment (with moving objects) is usually expected – a *recursive* analysis mode is defined.

Thus, at first a general scheme for batch mode analysis of *N-images* is proposed in the paper. This solution scheme is based on appropriately designed ANN-learning algorithms. Such kinds of biologically-motivated methods may be seen as computational counterparts of the human visual perception mechanism. Three implementations of this scheme are developed and tested for particular analysis tasks at the lower levels of image analysis, i.e. adaptive algorithms for the problem of many image source restoration from their mixtures (on the signal processing level), an adaptive image compression and classification approach (on the iconic level) and a visual motion detection/estimation (on the segmentation level).

In the second part of this work, a general scheme for the recursive analysis of infinite N-image sequences is developed. This solution scheme is based on the dynamic system theory, where the previous N-image results are adapted in the course of current N-image analysis of the sequence. This scheme, called *adaptive object recognition*, solves by general terms the problem of searching for the best, consistent set of object hypotheses, that are generated and tracked in the image sequence. Finally, three implementations of this scheme are developed and tested: a model-independent 2-D object tracking in traffic scenes (on the segmentation level), model-based road following and 3-D vehicle recognition for autonomous navigation, and a traffic scene recognition system (on the scene interpretation level). This kind of application requires highly sophisticated sensors and computation methods, and it constitutes a real challenge for image analysis systems.

Notation

Symbol	*Description*
	In chapters 2 - 5.
$\boldsymbol{x}(t)$	The vector of input signals of an ANN (mixed signals in BSS)
$\boldsymbol{y}(t)$	The vector of output signals from an ANN (separated signals in BSS)
$\boldsymbol{W}, \boldsymbol{H}$	Weight matrices of excitatory and inhibitory connections of an ANN
$\boldsymbol{y}^{(l)}$	Outputs of the l-th layer in a multi-layer network
$\boldsymbol{W}^{(k)}$	A weight matrix of the k–th ANN layer
$\boldsymbol{\Theta}$, Θ_i	A bias input vector and a single bias input in an ANN
$\psi(.)$	A general activation function of an ANN
$\eta(t)$	The learning rate of an ANN's learning rule
$\boldsymbol{s}(t)$	The vector of source signals in BSS or desired outputs of an ANN
$\boldsymbol{z}(t)$	The vector of output signals after redundancy elimination in BSS
$D(\boldsymbol{W})$	The Kullback-Leibler divergence considered for the outputs in BSS
$\boldsymbol{n}(t)$	The vector of additive noise signals in BSS
$\nu_R(t), n_R(t)$	The (unknown) environmental noise and (measured) reference noise
$\boldsymbol{v}(t)$	The vector of pre–whitened signals in BSS
$\boldsymbol{A} = [a_{ij}]_{n \times m}$	An unknown mixing matrix in BSS
$\boldsymbol{f}(.), \boldsymbol{g}(.)$	A vector-pair of activation functions in adaptive BSS
$\boldsymbol{V} = [V_{ij}]_{l \times n}$	A pre–whitening matrix in adaptive BSS
$\boldsymbol{W} = [w_{ij}]_{l \times n}$	A global source separation (de–mixing) matrix in adaptive BSS
$\widehat{\boldsymbol{W}} = [\hat{w}_{ij}]_{l \times l}$	A source separation matrix after pre–whitening in BSS
$\boldsymbol{R}_{xx} = E\{\boldsymbol{x}\boldsymbol{x}^T\}$	The covariance matrix of signal vector $\boldsymbol{x}(t)$
$\boldsymbol{u}_i$	The i-th principal eigenvector of the matrix $\boldsymbol{R}_{xx}$
λ_i	The i-th eigenvalue of the matrix $\boldsymbol{R}_{xx}$
$\widehat{\kappa_4}$	The normalized kurtosis of a source signal
$\boldsymbol{B}(z) = [b_{ijk}]_{n \times p \times N}$	The cube of convolution coefficients in the model of additive noise vector generation $\boldsymbol{n}$ in BSS
$\tilde{\eta}(t)$	The learning rate in adaptive noise cancellation
$\tilde{\boldsymbol{x}}(t)$	Transformed sensor signals after noise cancellation in BSS
	In chapters 6 - 9.
$\boldsymbol{s}(k)$	The state vector (in discrete time) of an object hypothesis
$\boldsymbol{m}(k)$	The *measurement* vector corresponding to an object
$\boldsymbol{F}(k)$	The state transition function
$\boldsymbol{H}(k)$	The state projection function
$\boldsymbol{s}^+(k), \boldsymbol{s}^*(k)$	The predicted and estimated object state vector at time k
$\boldsymbol{I}$	The identity matrix
$\boldsymbol{K}(k)$	The gain matrix of the recursive estimator
$\boldsymbol{P}^+(k), \boldsymbol{P}^*(k)$	The predicted and estimated covariance matrix of object state
$\boldsymbol{Q}(k)$	The covariance matrix of state noise
$\boldsymbol{R}(k)$	The covariance matrix of the measurement error
$\tilde{\boldsymbol{v}}, \boldsymbol{v}$	Contour change vector of a continuous 2-D contour and of its discrete mapped contour in the digital image
v_z	The relative 2-D contour length change rate
$\boldsymbol{s}(k) = (\boldsymbol{s}^d(k), \boldsymbol{\xi}(k))$	A 3-D object state, which consists of trajectory and shape sub-states
$V(k), \omega(k)$	Translatory and rotational velocities in $\boldsymbol{s}^d(k)$

1 INTRODUCTION

The goal of computer-based image analysis can be defined as: *given a sensor data of a single image or many images of some scene, solve one of the following problems (tasks) with increasing complexity:*

1. *the image restoration (reconstruction) problem,*
2. *the image classification problem,*
3. *the image segmentation problem (image structure or visual motion detection),*
4. *the object recognition problem,*
5. *the scene interpretation problem.*

In practice, by restricting the type of scenes in the sensor data to some application area, we achieve a *computer vision system* - a practically-feasible image analysis system (e.g. [HAN78, BRO81, BES85, BRO88, DIG88], [JAI88, NAG88, NIE90, NIB90, SKA95], [THO93, MAU96, TAD97]). For the classification of computer vision systems the type of input data should be considered first. Broadly speaking we can differentiate between single image analysis, many image analysis (volume images, viewpoint dependent image sequence) and the analysis of time sequences of images. In this work we shall concentrate on image sequence analysis, i.e. the analysis of many images (so called N-images) and time sequences of images.

This work reflects the experience of the author in the area of digital image analysis and recognition, especially in the analysis of volume images and time sequences of images [KAS85]– [KAS98], [CIK96a]– [CIK99], [KAK97].

We propose to follow a research methodology which explores the theories of *connectionist* and *dynamic* systems and applies their techniques in image sequence analysis. The biologically-motivated *connectionist* system theory explores the subject of artificial neural networks [LIP89, MAR91] and semantic networks [SHA88, NIS90, SAG90], i.e. net structures, associated learning algorithms and activation mechanisms . We shall show that the ANN–theory is especially suitable to low–level image analysis tasks, whereas the semantic network supports the intermediate and high–level image analysis, by providing a general framework for structural model representation and activation. The *dynamic system* theory provides mechanisms for object tracking and recursive state estimation [HAY96, RUT94, BRA75, WUE88, WUR88].

1.1 AN IMAGE SEQUENCE ANALYSIS SYSTEM

Let us introduce two main classification schemas for image sequence analysis systems, according to:

1. the data abstraction level, and
2. the image sequence processing mode.

The classical general scheme of automatic image analysis, as specified by the pattern recognition theory, assumes processing the data over a hierarchy of abstraction levels [KAN80, NAG79, NAG88, NIE81]:

1. the *signal level*, for example the signal restoration problem (e.g. [GON87, CIC94]),
2. the *iconic level*, for example the problem of iconic-based image classification (e.g. [SKA95, ROW98]),
3. the *segmentation level*, for example the image structure detection (e.g. [MAR80, CAN86]) or visual motion estimation problems (e.g. [AGG88]),
4. the *object recognition level*, for example the model-based object recognition problem (e.g. [BES85, GRI90],
5. the *cognitive analysis level*, for example the many object-scene interpretation problem (e.g. [RIS88]).

Visual object recognition and scene interpretation are key capabilities required for the successful operation of both biological organisms and artificial robots [BES85, GRI90, RIS88]. Obviously, the earlier performed signal restoration / enhancement [GON87] and image classification / segmentation steps [NIE81, SKA95] are often necessary conditions for object recognition / scene interpretation [NIE90].

Image sequence analysis systems may provide vital support to both human and robot work. Their particular application areas could be differentiated between monocular image sequences from a *stationary* camera (A) (with moving objects) or from a *moving* camera (B) (with moving objects and stationary background). The observed scenes can either be laboratory (indoor) scenes (with artificial lighting conditions and a fully predefined environment) (l) or natural (outdoor) scenes (natural light, partly undefined environment) (n). The conditions induced by the case product ($A \times l$) (i.e. stationary camera, artificial lights) are usually suitable for 3–D structure and motion reconstruction of single scene objects [BRO86, SUB88, TSA84, ULL79]. A relatively large image size of the projected object (at least $100 \times 100[pixel^2]$) is necessary for a robust reconstruction. The conditions given by class ($A \times n$) may occur in vision based security systems at entrances and control points or in traffic scene analysis systems [BIE89, KOL93, FER94]. The image sequences given in the case of ($B \times l$) are used in vision based robotics (i.e. an active camera), whereas the class ($B \times n$) applies for example for automatic satellite control [WUE88, GEN92] or for driver assistance systems [MAS92, MAU96].

1.1.1 Hierarchy of image analysis tasks

Main image sequence analysis tasks can be distinguished for each of the five data abstraction levels mentioned above (see *Figure* 1.1).

At the signal level image source restoration and enhancement may be required. For example, in this work we consider here the separation of several source signals from their many mixtures with noise (e.g. [JUT91, AMA96, KAS96a, KAS97a]) (see *Figure* 1.1(a)).

At the iconic level a texture-based classification of well-framed images may already solve the analysis problem in many applications (e.g. classification of "faces" [TUR91]), or at least it may provide the current measurement data required by a recursive object recognition procedure (see *Figure* 1.1(b)).

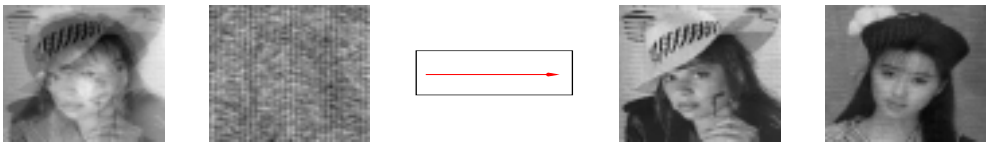
Various processes of image segment and image structure detection [CAN86, CHE87, DER87], attention selection (object window detection) [ROW98], as well as visual motion estimation and segment tracking at the segmentation level are usually obligatory processes in most computer vision systems (see *Figure* 1.1(c)). In the case of image sequences from a stationary camera various robust visual motion detection and estimation methods are available [AGG88]. Pixel-based estimation of *dense* visual motion is called *optical flow* detection [BAR94]. The detected visual motion can be applied for splitting the image into moving and non-moving regions [BOB94], [BOU93]. It also may be applied for the compression of image sequences [DOM98]. The image segment detection in individual images may be extended in the case of image sequence analysis to the segment tracking process. As motion data is immediately available from tracked segments, a so called "sparse" visual is the result of image segment tracking. Various segments can be distinguished: pixel blocks [EKL94], edge segments [DER90, ZHA94, KAS93b], point features [SAL90, KAS94a], contours [KAS93c] and active contours [KAW88, BLA93], symmetric segment groups [BER93] or 2–D objects [REG94], [SCH92]. The sparse visual motion or a dense optical flow may be evaluated for the purpose of 3–D structure reconstruction [DRE82, TSA84] or real object motion [WAX88], [TAN93, TAN94]. In both cases the image data is used to solve the non-regular inverse 3–D projection transformation of an object of specific type [ULL79, SUB88, HUA94].

Most industrial vision systems with CAD-like models of the environment perform a model-based 2-D or 3-D object detection, classification and recognition at the object recognition level (e.g. [BRO81, BES85, HWA86, JAI88]), where single object scenes in single images are usually assumed. Image sequence analysis applications include for example vision systems for automatic satellite docking at space stations [WUE88, GEN92], single object tracking [DRE82, KOL93, KAS95a], traffic scene analysis [FER94, WET94, KAS96] and autonomous road following and driver support [DIC88, MAS92, THO93, KAS97, KAS98]. If many object scenes are analyzed, multiple tracking tasks have to be performed in parallel and various consistency checks should be made at the proper time. The *correspondence* problem between many tracked object hypotheses and new image data (measurements) [REI79] is of exponential complexity. To solve this problem by a matching process a trade–off between statistical optimality and heuristic pruning is sought for in most of the applications [PEA84, COX93]. Both traffic scene analysis and driver support systems are application areas, which deal with many moving objects of complex structure (see *Figure* 1.1(d)).

Up to now only laboratory systems exist that provide complete scene interpretations (descriptions) (at the cognitive level of image analysis), while applying artificial intelligence methodology [HWA86, RIS88, PAU93, KAS95]. The systems have a *knowledge-based architecture* [NIE90, SAG90], i.e. two main modules can be distinguished - the *knowledge representation form* for the storage of the model and data (intermediate analysis results) and the *control* module for the controlling of the analysis flow (see *Figure* 1.1(e)).

General control strategies for scene interpretation were proposed, for example:

1. a hypothesis- generation and verification cycle with deterministic control flow [KAN80, NAG79], and interpretation cycle handling time-dependent processes and image sequence descriptions [NAG88],

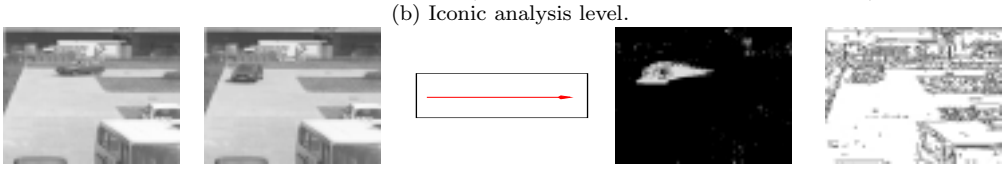


N noisy images with mixed sources → M restored source images

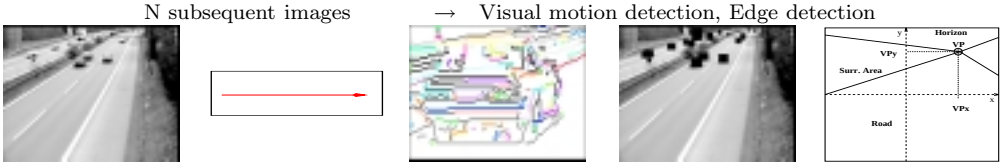
(a) Signal processing level - many-image restoration.

Input image → Classification (e.g. into "face", "fingerprint" or "road")

(b) Iconic analysis level.

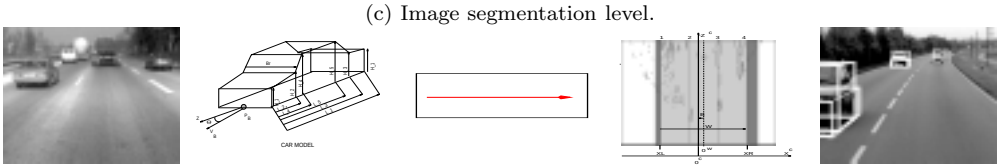


N subsequent images → Visual motion detection, Edge detection



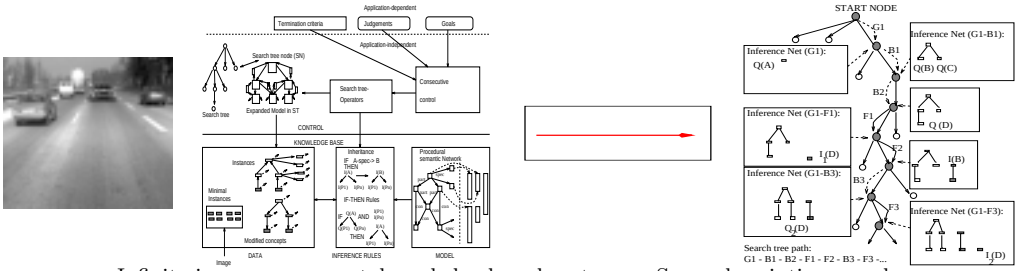
Infinite image sequence → Segment grouping, 2-D object tracking, Vanishing point recognition

(c) Image segmentation level.



Infinite image sequence + 3-D object model → Model-based road and 3-D object recognition

(d) Object recognition level.



Infinite image sequence + knowledge-based system → Scene description search

(e) Cognitive level.

Figure 1.1 Examples of analysis tasks at different processing levels of image sequence analysis: (a) restoration of source images (at the signal level), (b) image classification (iconic level), (c) visual motion detection/estimation, segment detection, segment grouping, 2-D object tracking and recognition (segmentation level), (d) model-based object detection and tracking (object recognition level), (e) many-object scene description (cognitive level).

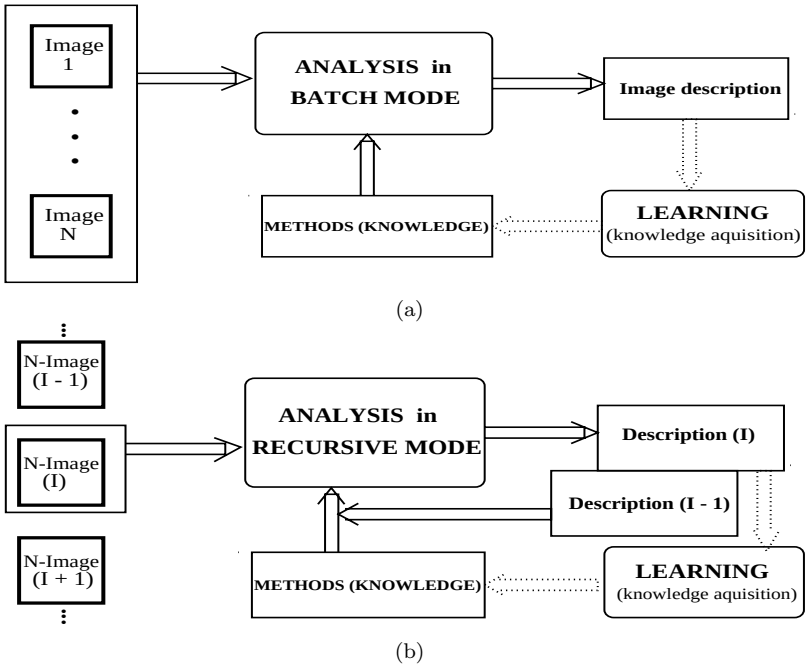


Figure 1.2 Two image sequence analysis modes: (a) the batch mode and (b) recursive analysis mode.

2. transformations in state space - a control flow corresponds to a path in search space [NIE81], or a consecutive search-based strategy [WEI91, KAS94b],
3. "blackboard"-based communication between bi-directional processes with non-deterministic control flow [HAN78a, RIS88].

During the recursive mode of image sequence analysis we need a *consecutive control* mechanism and a properly designed *scene model* for a given application. Consecutive analysis means, that the image analysis (processing) order corresponds to the order of acquired images, and at the same time a predefined limitation for memory and processing time per single image is satisfied.

1.1.2 Finite or infinite image sequence

We distinguish a second classification scheme for image sequence analysis according to the length of processed images (see *Figure* 1.2):

A. batch mode - for image sequence of limited length *N*.

This mode assumes that a single image, a stereo-pair, many images from different viewpoints (taken at the same time) or short time-sequence of *N* images are processed jointly in a single processing step. The (possibly) next *N-image* is processed independently from other *N-images*.

B. recursive mode - for a N-image sequence of unlimited length.

A theoretically infinite sequence of N-images is analyzed in a consecutive (incremental) manner, leading to a steady improvement of the propagated solution.

The term *N-image* means here, that at one processing step up to N images can be available.

The meaning of the term *"adaptive"* in this work is twofold:

1. ANN-learning based analysis of single images or finite image sequences.

   In the batch analysis mode stationary scene (environment) is usually expected. Among connectionist systems the learning-based methods in *artificial neural networks (ANN)* are especially suitable for solutions to analysis problems at levels 1,2 and 3 [PAO89, MAM94].

   In the learning phase the ANN is trained by providing known image data samples. In the analysis phase the ANN is stimulated by the unknown image data and its outputs correspond to the results of the analysis task. If the analyzed data is changing significantly in time (non-stationary systems), a frequent modification (repeated training) of the ANNs will be required.

2. Recursive object estimation in infinite image sequences.

   A recursive analysis mode is especially suitable for the analysis of a non-stationary scene environment (moving objects or moving camera are usually expected). We shall show how model- and knowledge-based systems, especially on the basis of semantic network representation, can be successfully applied for higher level analysis of image sequences [KAS87, KAS89a, NIS90] if combined with the dynamic system theory [HAY96, RUT94, BRA75].

   The proper solution of such analysis tasks requires the application of the dynamic system theory in order to properly adapt (modify) the previous image results to current image analysis results of the time sequence. For continuous systems (i.e. no abrupt changes or wrong hypotheses) the adaptive estimation of the second type can be abstractly modeled as a steady ANN-learning process with an adaptive learning rate.

1.2 OBJECTIVES AND SUBJECT OF THIS WORK

1.2.1 Main objective

The main objective of this work is summarized by the following three targets:

A) to develop **adaptive analysis** schemas for both finite and infinite image sequence analysis,

B) to design exemplary learning-based ANN solutions to iconic and sub-symbolic (i.e. image restoration, classification, motion detection) finite image sequence analysis problems,

C) to design exemplary recursive analysis solutions to symbolic and high-level infinite image sequence analysis problems (i.e. model-based 2-D and 3-D object tracking, general consecutive control).

1.2.2 Adaptive analysis of N images

The advantage of a connectionist approach (on the basis of artificial neural networks) to intermediate-level image analysis over traditional approaches is:

- the possible unification of many aspects of vision by one common theory with relatively simple elementary actions,
- the inherent potential for parallelism, and
- its close connection to biological experiments.

In the past ANNs have been applied successfully for low-level image analysis tasks. The three main types of behavior of ANNs, from the point of view of pattern recognition theory, are: associative memory (e.g. Hopfield net), classification or recognition of patterns (e.g. Hamming net, multi-layer back-propagating network), clustering of patterns (ANN with unsupervised learning).

Let us bear in mind that learning-based analysis processes (i.e. the adaptivity of first type) could potentially appear at all image sequence analysis levels. But up to now it is difficult to apply such an adaptive scheme to the two upper levels, i.e. the object recognition and scene interpretation levels, where it is still under development. In this work we shall develop three particular implementations of the first adaptive analysis scheme at the three lower levels:

Many-image restoration from their mixtures. At the low (iconic) processing level the original signals could be distorted by physical means or be mixed with distortions, like noise etc. By processing a multi-image data, instead of a single image, the restoration and demixing tasks could be solved by unsupervised learning algorithms of ANNs. We propose efficient algorithms for blind source separation and extraction (BSS, BSE).

Image compression and classification. Assuming well-framed images a texture-based compression and classification of entire images could be performed sufficiently fast, as required in different real-time vision applications. But using solely iconic representations even small images may lead to large classification space. Therefore we shall propose a two-step compression-classification pipeline with ANN-learning algorithms, where a PCA-based space reduction (lossy compression step) precedes the DA-based classification (PCA means principal component analysis and DA means discriminant analysis). The classification works well for both single image and many-image data.

Visual motion detection. We apply an efficient adaptive algorithm for visual motion detection (pixel-based motion). The motion image is used as a mask in the segment grouping process, in order to separate the image segments into stationary background and moving objects. A novel approach is suggested, based on blind source deconvolution (BSD), for the separation of image into regions corresponding to differently moving objects.

1.2.3 Adaptive object recognition scheme

In the case of infinite image sequences the analysis system generally deals with non-stationary scenes. This means that previously obtained image analysis results may not fit the current changed environment and should be adapted to the appeared changes.

Thus we need a consecutive analysis scheme which provides the best way for the estimation of dynamic object parameters.

For symbolic- and cognitive-level analysis we propose a general recognition scheme that consists of following features:

Dynamic state model. The unknown state of the object is to be estimated, as only some state projection's (measurements) are available in the image data. Depending on the abstraction levels the state can take one of the following implementations:
- weights of a processing neural network,
- base vectors of a classification space,
- 2-D and 3-D object parameters,
- search tree nodes in a solution search space.

Recursive state estimation. There is a need for next time object detection (measurement) and of prediction and update of each object's state.

Resolving the competitive hypotheses. At all levels always the problem of redundant competitive hypothesis generation may appear and the need for the selection of the best (regarding chosen optimization criterion) consistent subset of hypotheses. Thus a specific problem may be the determination of the number of (possibly overlapping) objects, which appear in the scene.

It will be shown, that by applying the general analysis scheme to some particular problems, a steady improvement both of the dynamic state and the analysis result appears. Three implementations of the adaptive recognition scheme for infinite image sequence analysis will be developed. Two possible application fields of these methods will be presented - road traffic scene analysis and autonomous navigation on roads:

2-D object tracking and recognition. We focus on the tracking of segment groups, represented by their boundary contours in the image, and on the dynamic recognition of the road's vanishing point in the image plane. No explicit object model is provided that can restrict the image analysis procedures.

Model-based object tracking and recognition. By applying the general adaptive recognition scheme in autonomous navigation systems we can provide an efficient model-based road following and 3-D object recognition. The better performance of the recognition task and the estimation of road and object parameters (e.g. road's curvature, object's velocities) in comparison to single image analysis is demonstrated.

Consecutive scene description. The framework of knowledge-based systems is used for the design of an application system for traffic scene description in a consecutive manner. We propose a consecutive graph search for the control module and present procedural semantic network representation of the scene model.

1.3 THE CONTENT AND REFERENCES TO THE AUTHOR'S WORK

The work consists of this introduction and the following nine chapters. The author's motivation for starting research on image analysis was to provide computers with some intelligence, i.e. to make an *adaptable computer* [KAS85], which could adapt itself to user preferences and operate and communicate more conveniently with the user.

The classification of image analysis into a hierarchy of data abstraction levels, proposed in this introduction, reflects the early individual work of the author towards a unified computational model for computer vision systems [KAS86, KAS87, KAS89a].

The main chapters from 2 to 9 deal with two analysis schemas:

1. a general model of adaptive ANN-based analysis of N-images (chapter 2) and three particular implementations of this model (chapters 3-5),
2. an object recognition scheme for the analysis of infinite image sequences (chapter 6) and its three particular implementations (chapters 7-9).

The proposed two image analysis models (chapter 2, 6) are generalizations of the author's experience and previous research results in the considered fields. In the presented complete form they have not been published before. Apart from these models most of the particular methods, presented in chapters 3–5 and 7–9, were developed and tested by the author himself in the past. Hence, appropriate references to previous publications of the author are provided. In most of the referred publications the author of this monograph was the single author or he was the first co-author of a few co-authorized papers. Some of the publications are co-authorized papers, as in the past the author was a member of large international teams, where he collaborated with his team leaders and other team members. Nevertheless, the referred work was done either solely by the author or it presents the author's work done in collaboration with his team leaders - in the latter case, the author has made a dominating contribution. Also a few publications are referred to, where the author is a second or further co-author only - in such cases only the explicitly numbered sections are referred to, for which the author made a unique or dominating contribution.

In chapter 2 ANN-based approaches to low-level image sequence analysis are reviewed and the first adaptive analysis scheme - the ANN-learning based scheme for batch analysis of image sequences - is developed. This chapter contains previously unpublished material and it generalizes the author's experience in low-level image processing and image segmentation, resulting from the development of approaches to adaptive image processing (e.g. [KAS96a, KAS96b, KAS97a], [KAK97] - section 5, [CIK99] - section 4), image sequence segmentation and visual motion detection (e.g. [KAS93a, KAS93b, KAS93c]). Chapters 3, 4 and 5 describe three particular implementations of the above adaptive analysis scheme, developed by the author quite recently.

In chapter 3 the motivation, description and test results of ANN-learning algorithms for the problem of many image restoration from their mixtures (at the signal processing level) are given. At first three types of very efficient separation rules for the basic *blind source separation problem* are proposed. A more detailed description and test results are provided in the author's related work ([KAS96a], [CIK96a] - section 4 , [CIK99] - section 4). Separation rules for the separation problem in large convoluted noise are derived next. This reflects the author's work and tests in this subject, as given in [KAS97a]. At the end of this chapter the newest, unpublished results for solving the blind source extraction problem, i.e. the BSS problem with fewer sensors than sources, are included.

In chapter 4 an adaptive image compression and classification approach is proposed (at the iconic level), which is applicable to well-defined image frames. The approach consists of two main steps: an adaptive method for image compression and an approach method to image class detection. For image compression we use solutions of the

PCA (principal component analysis) or PSA (principal subspace analysis), i.e. unsupervised learning rules for feed-forward neural networks. A very efficient and fast learning algorithm was first proposed by the author and his co-authors in [CIK96] - sections 3-4 and [KAS96b], where detailed explanations and tests of this and related neural algorithms can be found. For the subsequent classification step a newly developed supervised learning rule for ANN is used [KAS00].

Chapter 5 focuses on an adaptive approach to moving object separation from the stationary background (at the segmentation level). At first a 2-D module for image segmentation of image sequences is described [KAS97]. Then a visual motion-based segment grouping process is proposed [KAS93a]. Finally, another motion detection approach is mentioned, which is based on a relaxation ANN [KAS00].

The second part of this work concentrates on the analysis of infinite image sequences, with applications to object tracking and interpretation of many-object scenes. It reflects the author's previous work on general modelling of image sequence analysis (knowledge representation and its control mechanisms) [KAS89a, KAS90, KAS94b, KAS95] and the author's experience of designing and implementing a vision system for autonomous navigation and driver support [KAS94, KAS94a, KAS95a], [KAS97, KAS98]. The model is also influenced by the author's unpublished work on adaptive learning rates in ANN-learning for non-stationary signals (the problem statement itself follows [CIK96b]).

In chapter 6 a general adaptive recognition scheme for the recursive analysis mode is proposed. This approach constitutes the final and unpublished version of a development, which can be followed in the author's previous publications [KAS91, KAS94d, KAS95, KAS96, KAS97, KAS98]. In the subsequent three chapters 7, 8 and 9, different implementations of this general scheme are described, which were developed and implemented by the author for the purpose of traffic scene analysis under ego-motion.

In chapter 7 two model-independent 2-D object tracking methods are described (at the segmentation level of image analysis). The first application deals with segment group tracking (represented by their boundary contours) and it summarizes author's work given in [KAS93a, KAS93b, KAS93c, KAS94a, KAS97]. Secondly, the general approach is applied to the recognition of the vanishing point in the image plane, which allows an on-line correction of the projection conditions. A more detailed description of the method and test results can be found [KAS97].

Chapter 8 deals with model-based object recognition in image sequences under ego-motion. Two specific recognition problems are solved - the road following problem [KAS98] and the 3–D vehicle tracking and recognition problem [KAS94,KAS95a,KAS97].

The third implementation of the adaptive recognition scheme follows in chapter 9 (at the scene interpretation level). A general semantic net-based system for traffic scene analysis is designed in the system shell environment called ERNEST. Objects are equivalent to data inferences (instances, modified concepts) obtained during the analysis. A tree search-based consecutive control is developed and a proper model network for our application is designed. This chapter reflects the author's previous work within the ERNEST environment and his design work on consecutive control and knowledge representation [KAS89a, KAS91, KAS94b, KAS95, KAS97].

The work ends with conclusions in chapter 10, in which the main results and achievements of this work are summarized.

2 ADAPTIVE ANALYSIS OF N IMAGES

A unified adaptive approach to lower- and upper-level image analysis tasks has not been recognized yet, although *connectionist* research exists, which is a synthesis of results in Artificial Intelligence (AI), artificial neural networks (ANN) and parallel distributed processing (PDP). It tries to work out a unified theory for different data levels of signal analysis and different forms of knowledge processing. Up to now the common model forms in connectionism research have been artificial neural networks for low-level analysis and semantic networks for symbolic (high-level) analysis. We consider here three main aspects of an image analysis system design from the point of view of the connectionist theory [MAM94, NIE90]:

1. *Knowledge representation* (the model network). Here we determine the network type and structure, with its inference rules (for semantic networks) or activation mechanism (for ANNs).
2. *Control* (analysis strategy). The analysis flow is the result of activating the model network (i.e. semantic net or ANN) by selecting data samples and applying inference rules or the activation mechanism, respectively.
3. *Model acquisition*. Setting a new model by applying network learning algorithms to sample input (and output) data (unsupervised and supervised learning of ANN, different types of symbolic learning for general networks).

Hence in our view, an automatic model learning ability is an inherent feature of a connectionist-like image analysis system. Conventional statistic methods of model acquisition are based on: regression analysis (e.g. generalized additive model [HAS86] and CART (inductive solution trees and regression) [BRE84]), non-parametric methods (cluster analysis) and conventional classification methods (linear and quadratic discriminant analysis, K-nearest neighbor classifier, kernel-based classifier) or the Bayes classifier [MOL89]. They all influence the development of specific problem-to-model mapping solutions and corresponding learning rules in connectionist systems.

In this chapter a general model for adaptive (i.e. learning-based) analysis of N images at the lower data abstraction levels, i.e. from iconic level to segmentation level, is developed. The considered limited area of image analysis tasks can be modeled by a pipeline of filter-like, information reduction-like and domain-transformation (classification) modules. Some conventional pre- and post-processing, dealing mostly with data reorganization, may also be included.

In the neural network research a typical learning mechanism (in feed-forward ANN) or control mechanism (in feedback ANN) is a relaxation algorithm over nets (for setting weights or neuron activities, respectively) , i.e. a connectionist system converges through local iteration of its elements to a stable state, that corresponds to the required model or solution (response) respectively.

2.1 GENERAL ANN STRUCTURES AND LEARNING TYPES

In this work we consider following basic structures of ANNs:

1. single layer networks (feed-forward and linear recurrent networks),
2. multi-layer feed-forward networks (e.g. backpropagating networks and self-organizing networks),
3. nonlinear feedback (recurrent) networks (e.g. single-layer *Hopfield* net, multi-layer *Boltzmann* machine).

Obviously the main types of ANN-learning approaches are: *supervised* and *unsupervised* learning.

2.1.1 Single layer linear networks (feed-forward, recurrent)

A single-layer feed-forward network consists of neurons of similar type with excitatory connections, represented by their weights $\boldsymbol{W}$. A linear feed-forward network performs the transformation:

$$\boldsymbol{y} = \boldsymbol{W}^T \ \boldsymbol{x} \tag{2.1}$$

of its input vector $\boldsymbol{x}$ to its output vector $\boldsymbol{y}$.

A single layer recurrent network consists of neurons of one type with inhibitory connections between them, represented by their weights $\boldsymbol{H}$. A linear recurrent network performs the following transformation $\boldsymbol{x} \Rightarrow \boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{H} \ \boldsymbol{y}. \tag{2.2}$$

This is equivalent to a feed-forward network which performs the transformation:

$$\boldsymbol{y} = (\boldsymbol{I} + \boldsymbol{H})^{-1} \ \boldsymbol{x}, \tag{2.3}$$

where $\boldsymbol{I}$ is the identity matrix of appropriate size. Combined networks with both excitatory and inhibitory connections can be defined. For example a recurrent network with interneurons (with outputs $\boldsymbol{h}$), having symmetric excitatory and inhibitory connections [PLU93], i.e. a forward path $\boldsymbol{y} \rightarrow \boldsymbol{h}$ and a backward path $\boldsymbol{h} \rightarrow \boldsymbol{y}$. This leads to transformations:

$$\boldsymbol{i} = \boldsymbol{H}^T \ \boldsymbol{y} \tag{2.4}$$

$$\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{H} \ \boldsymbol{h} = \boldsymbol{x} - \boldsymbol{H}\boldsymbol{H}^T\boldsymbol{y}. \tag{2.5}$$

Again, this is equivalent to a feed-forward network which performs the transformation:

$$\boldsymbol{y} = (\boldsymbol{I} + \boldsymbol{H}\boldsymbol{H}^T)^{-1} \ \boldsymbol{x}. \tag{2.6}$$

Hence, linear recurrent networks are usually considered together with feed-forward networks.

2.1.2 Multi-layer feed-forward networks

Backpropagating networks (with hidden layers)

A three-layer perceptron consists of an input layer, an output layer and one hidden layer (more hidden layers are possible) [RUM86]. The transformation of each layer $l, (l = 1, 2, 3)$ is:

$$\boldsymbol{y}^{(l)} = \psi[\boldsymbol{W}^{(l)}\ \boldsymbol{y}^{(l-1)} - \boldsymbol{\Theta}^{(l)}],\ \text{with}\ \boldsymbol{y}^{(0)} = \boldsymbol{x}, \tag{2.7}$$

where ψ is the nonlinear activation function and $\boldsymbol{\Theta}^{(l)}$ is the vector of additional bias inputs. The nonlinearity ψ is usually a sigmoid activation function, defined as:

$$\psi(y) \quad = \quad \frac{1}{1+\exp(-y)},\ \text{with continuous derivative -}\ \frac{\partial\psi}{\partial y} = \psi[1-\psi]. \tag{2.8}$$

The weights are determined during the so-called *back-propagation* algorithm, which is a learning algorithm of supervised type. There is theoretical proof that a three-layer perceptron with n neurons in a single layer is sufficient for the definition of arbitrary measurable sets in the R^n–space [RUM86].

SONN (feed-forward networks with unsupervised learning

Self-organizing neural networks (SONN) are feed-forward networks using unsupervised learning rules. A common network structure consists of elementary neurons with feed-forward excitatory connections and lateral inhibitory connections. The early SONNs operated in the winner-take-all fashion, by using modified Hebbian rules for excitatory connection learning. For example this allows the creation of an adaptive classifier [AMA67]). Examples of representative SONNs are: competitive learning ANN [RUM85], Kohonen map [KOH88], cognitron and neocognitron [FUK88], ART [CAR87].

A *competitive learning* network is applied for unsupervised *clustering* or unsupervised *vector quantization*. In the clustering approach the input data are expected to be grouped in the representation space in "clusters" and the goal of learning is to find these inherent clusters in the input data, while the result of processing new data is to give the cluster label of this input pattern. In vector quantization the learning goal is to find optimal digitalization of a continuous n-dimensional input space and in working mode to represent each input vector by the label of the subspace, to which it belongs. Additionally to clustering in vector quantization we choose the cluster representative element. The basic structure of an ANN performing clustering is a two-layer feed-forward network:

1. The first layer is the competitive learning layer - each of the outputs i is connected to all input units j and the activation value of output i is:

$$y_i = \boldsymbol{w}_i^T\ \boldsymbol{x}. \tag{2.9}$$

2. The second layer performs: (1) a selection of a single output neuron k, with the highest activation value and (2) a change of activations due to a relaxation process, such that $y_k = 1$ and $y_i = 0$ for $i \neq k$. Every output neuron i is connected to all other

outputs m by inhibitory links and to itself by an excitatory link such that:

$$w_{im} = \begin{cases} -\epsilon & \text{, if } i \neq m, \\ +1 & \text{, if } i = m, \end{cases} \tag{2.10}$$

where ϵ is a small positive-valued parameter. It can be shown that this network converges to a situation where only the neuron with highest initial activation survives, whereas the activations of all other neurons converge to zero [LIP89].

The Kohonen's *feature map* approach is applied for the detection of similarities between feature vectors and it can be seen as an extension of a competitive learning network. The structure of output units expresses some order in 2–D space, such that neighbors represent similar classes of input patterns. For example the outputs are organized in an array and the network is used for mapping similar feature vectors to neighbor regions in the 2-D space of outputs. If inputs are distributed in $\boldsymbol{R}^N$ without restrictions and their order must be preserved, then the dimensionality of outputs must be at least N. However, if the inputs form a subspace in $\boldsymbol{R}^N$, then a lower dimensionality of outputs may be used. In a Kohonen network two layers can be distinguished: a feed-forward layer with excitatory connections using the Kohonen's competitive learning and a feed-forward layer with inhibitory connections and excitatory self-connections, performing the "winner-take-all" selection of the output with the highest initial activity.

In the *Cognitron* net, proposed by Fukushima, there exist excitatory neurons E and inhibitory neurons H. The neurons are organized into a multi-layer structure, where each layer l is a 2-D array of excitatory neurons $E^{(l)}(i,j)$ and inhibitory neurons $H^{(l)}(i,j)$. The neuron $E^{(l)}(i,j)$ is connected on the input side via weights $w^l(i+\delta_i, j+\delta_j)$ with neurons $E^{(l-1)}(i+\delta_i, j+\delta_j)$, and via weight $v^l(i,j)$ with neuron $H^{(l-1)}(i,j)$. Here, $\Delta_{i,j} = \{(\delta_i, \delta_j)\}$ is a neighborhood called the *attention area* of the neuron. The output y of an excitatory neuron $E^{(l)}(i,j)$ is computed from:

$$y[E^{(l)}(i,j)] = \psi\left[\frac{1+x_e}{1+x_h} - 1\right] = \psi\left[\frac{x_e - x_h}{1+x_h}\right], \tag{2.11}$$

where x_e is the combined excitatory input from other E-neurons and x_h is the inhibitory input received from the H-neuron, given as follows:

$$x_e = \sum_{\Delta_{i,j}} w^{(l)}(i+\delta_i, j+\delta_j)\ \ y[E^{(l-1)}(i+\delta_i, j+\delta_j)]; \quad x_h = v^{(l)}(i,j)\ y[H^{(l-1)}(i,j)] \tag{2.12}$$

The activation function is as follows:

$$\psi(z) = \begin{cases} z & \text{, if } z \geq 0, \\ 0 & \text{, otherwise.} \end{cases} \tag{2.13}$$

An inhibitory neuron $H^{(l-1)}(i,j)$ gets its inputs via fixed excitatory connections $c^{(l-1)}(\delta_i, \delta_j)$ from the neighbor neurons $E^{(l-1)}(i+\delta_i, j+\delta_j)$. The output activity of an inhibitory neuron is computed as:

$$y[H^{(l-1)}(i,j)] = \sum_{\Delta_{i,j}} c^{(l-1)}(\delta_i, \delta_j)\ \ y[E^{(l-1)}(i+\delta_i, j+\delta_j), \tag{2.14}$$

where $\sum_{\Delta_{i,j}} c^{(l-1)}(\delta_i, \delta_j) = 1$. The Cognitron network can learn to distinguish 2-D patterns in an image, which correspond to the input layer.

2.1.3 Non-linear feedback networks

In feed-forward nets in the working mode (i.e. after the weights were fixed) the information is propagated from input directly to the output. In feedback nets a relaxation process over the neuron activities appear, which terminates in some stable state giving the proper pattern on its output.

Single-layer Hopfield net

The Hopfield net is a single layer network of m output neurons with step nonlinearities at each neuron and a full number of pair-like symmetric connections, which play the role of excitatory connections during Hebbian learning, but which are inhibitory-like connections in the working mode.

Actually a relaxation process appears during the working phase, which always terminates, giving that one of the stored patterns on its output, which is nearest the input. In the working phase the activation value of a single neuron i is given as:

$$z_i(t) = \psi[y_i(t)], \text{ where} \tag{2.15}$$

$$y_i(0) = x_i, \qquad y_i(t) = \sum_j w_{ij} z_j(t-1) + \Theta_i, \ (\forall t > 0). \tag{2.16}$$

The step nonlinearity function $\psi[.]$, when applied to the output value $y_i(t)$ of the neuron, gives the current activation value:

$$z_i(t) = \begin{cases} +1 & \text{, if } y_i(t) > \epsilon, \\ -1 & \text{, if } y_i(t) < \epsilon, \\ z_i(t-1) & \text{, otherwise,} \end{cases} \tag{2.17}$$

with ϵ being some small constant. For $\epsilon = 0$ the above function takes: $z_i(t) = sgn[y_i(t)]$. Hence, the activation of the whole net is given as:

$$\boldsymbol{z}(t) = \psi[\boldsymbol{y}(t)], \text{ where} \tag{2.18}$$

$$\boldsymbol{y}(0) = \boldsymbol{x}, \qquad \boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{z}(t-1) + \boldsymbol{\Theta}, \ (\forall t > 0). \tag{2.19}$$

The Hopfield net reaches a stable state if all the neurons are stable, i.e. $z_i(t) = z_i(t-1)$. If the connections are bidirectional and symmetric, i.e. $w_{ij} = w_{ji}$ then the Hopfield network is guaranteed to reach a stable state. The Hopfield net works as an associative memory - a complete output pattern is retrieved given even an incomplete input pattern.

Multi-layer Boltzmann machine

The Boltzmann machine is an extension of the Hopfield net as it includes hidden units and a stochastic activation rule [ACK85], that is based on the principle of *annealing*. The network consists of visible neurons and a (possibly empty) set of hidden neurons. The units have binary output values and they are activated asynchronously with the probabilities p_i-s depending on the energies $\mathcal{E}_i$ of the neurons and the temperature T of the system:

$$P[z_i = +1] = (1 + e^{-\Delta\mathcal{E}_i/T})^{-1}. \tag{2.20}$$

This network is usually applied to obtain solutions of optimization problems.

2.1.4 Supervised (or associative) learning

In this approach the network is trained by providing both input samples and the desired output patterns to the net. Hence, an a priori set of categories is given by the teacher, in which the patterns are to be classified.

The basic learning rule in supervised learning is called the *Widrof-Hoff* (or *delta*) rule. The input interconnections indexed by j of the i-th neuron are strengthend in proportion to the difference between the current real and desired activations:

$$\Delta w_{ij} = \eta(s_i - y_i)x_j, \tag{2.21}$$

where s_i is the desired activation of i-th output neuron, provided by the teacher, x_j is the activation of the j-th incoming neuron and η is the learning rate.

The generalized delta rule appears in the *backpropagation* algorithm for the multilayer network, where each weight is updated according to the formula:

$$\Delta \boldsymbol{W}^{(l)} = \eta\ \boldsymbol{d}^{(l)}\ [\boldsymbol{y}^{(l-1)}]^T, \quad ((\boldsymbol{y}(0) = \boldsymbol{x}), \tag{2.22}$$

where $\boldsymbol{d}^{(l)}$ is the vector of correction values for neurons from the l-th layer. These vectors are specified in the backward order, starting from highest layer $l = L$ and proceeding backward to the input layer $l = 1$. The backpropagation algorithm minimizes the mean square error between the current output vector $\boldsymbol{y}^{(l)}$ and a desired output vector $\boldsymbol{s}$, provided by the teacher. Generally:

$$\boldsymbol{d}^{(L)} = (\boldsymbol{s} - \boldsymbol{y}^{(L)})\ .*\ [\frac{\partial \psi}{\partial \boldsymbol{y}}], \tag{2.23}$$

$$\boldsymbol{d}^{(l)} = \boldsymbol{d}^{(l+1)}\boldsymbol{W}^{(l+1)}\ .*\ [\frac{\partial \psi}{\partial \boldsymbol{y}}],\ \text{for } l = L-1, ..., 1, \tag{2.24}$$

where $.*$ means an element-by-element multiplication of two vectors. For a sigmoid activation function the above formula can be specialized to:

$$\boldsymbol{d}^{(L)} = (\boldsymbol{s} - \boldsymbol{y}^{(L)})\ .*\ (1 - \boldsymbol{y}^{(L)})\ .*\ \boldsymbol{y}^{(L)}, \tag{2.25}$$

$$\boldsymbol{d}^{(l)} = \boldsymbol{d}^{(l+1)}\boldsymbol{W}^{(l+1)}\ .*\ (1 - \boldsymbol{y}^{(l)})\ .*\ \boldsymbol{y}^{(l)},\ \text{for } l = L-1, ..., 1. \tag{2.26}$$

A further example of the supervised learning rule is the supervised LVQ algorithm, which is proposed in chapter 4 for the discriminant analysis problem during image classification.

For supervised learning in the Boltzmann machine the input and output are set to a training vector pair. With constant input and output the network is annealed, i.e. the temperature of the net is initialized to a high value and then it is continuously lowered until the net approaches thermal equilibrium at $T = 0$. In the equilibrium state the activity of the net is measured during a fixed amount of time, i.e. for each connection the fraction of time is measured during which both connected neurons are active. This annealing and measurement process is repeated for each training vector pair. For

each connection an average (expected probability) for both simultaneously active neurons at thermal equilibrium is computed $< z_i z_j >_c$. The same process is repeated when the output neurons are freely determined by the network. For all input training samples the average values $< z_i z_j >_f$ are measured.

Let the probability that the output neurons are in state A when the system is working freely be given by $P_f(A)$ and the desired probability be denoted as $P_c(A)$. The *Kullback mutual information* is used as the error measure between real and desired behavior of the net:

$$E_K = \sum_i P_c(A_i) log \frac{P_c(A_i)}{P_f(A_i)}. \tag{2.27}$$

In order to minimize the measure E_K the gradient descent can be used:

$$\Delta w_{ij} = -\gamma \frac{\partial E_K}{\partial w_{ij}}. \tag{2.28}$$

On the above assumptions the following can be derived [RUM85]:

$$\frac{\partial E_K}{\partial w_{ij}} = -\frac{1}{T}(< z_i z_j >_c - < z_i z_j >_f). \tag{2.29}$$

Hence the learning rule is:

$$\Delta w_{ij} = \eta(< z_i z_j >_c - < z_i z_j >_f). \tag{2.30}$$

2.1.5 Unsupervised learning (self-organization)

In this learning type an output unit is trained to respond to clusters of patterns within the input. The network must autonomously develop its own representation of the input training set.

In the basic *Hebbian* learning the excitatory weight between two neurons is increased if their activities are positively correlated. If two neurons i, j are simultaneously active then their excitatory interconnection must be strengthend:

$$\Delta w_{ij} = \eta y_i x_j, \tag{2.31}$$

where η is the learning rate. This type of learning can be used for principal component analysis in a single-layer network (see the solution to PCA in chapter 4).

The *Anti-Hebbian* learning appears when the j-th inhibitory weight of an i-th neuron is increased:

$$\Delta h_{ij} = \eta y_i y_j, \tag{2.32}$$

An anti-Hebbian learning algorithm can be derived from the minimization of mutual information between every pair of output units. It can be used to de-correlate the outputs from a single-layer neural network, so that they respond to different patterns (see the solution to blind source separation in chapter 3).

Competitive learning (as used for unsupervised clustering and LVQ) requires the computation of initial activities $\boldsymbol{y}(0)$ of all output neurons and the subsequent selection of

the best neuron, i.e. with the highest activation. Let us first assume that both input vectors $\boldsymbol{x}(t)$ and weight vectors $\boldsymbol{w}_i$ are normalized to unit length. Once the winner k has been selected, its weight vector in the competitive learning layer is updated as follows:

$$\boldsymbol{w}_k(t+1) = \frac{\boldsymbol{w}_k(t) + \eta[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]}{|\boldsymbol{w}_k(t) + \eta[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]|}, \tag{2.33}$$

where the division ensures the normalization of vector $\boldsymbol{w}$. Hence, every time an input vector is presented, the weight vector closest to it is selected by the second layer and by applying the above rule this vector is effectively rotated towards the input vector.

If the input data is not normalized, one should select the winning neuron k with its weight vector $\boldsymbol{w}_k$ by using the Euclidean distance measure:

$$|\boldsymbol{w}_k - \boldsymbol{x}| \leq |\boldsymbol{w}_j - \boldsymbol{x}|, \forall j. \tag{2.34}$$

Then the learning rule performs a shift of selected weight vector towards current input:

$$\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) + \eta[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]. \tag{2.35}$$

In the *Kohonen* network the competitive learning rule takes the form:

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + \eta f(i,k)[\boldsymbol{x}(t) - \boldsymbol{w}_i(t)], \forall i \in Outputs. \tag{2.36}$$

Now the weights of the winning neuron k and of all its neighbors i are adapted by applying a decreasing function $f(i,k)$. For $i = k$ the maximum value is $f(k,k) = 1$ and h decreases with the distance between outputs i and k.

In the *Cognitron* network a modified competitive learning rule is applied, i.e. the absolute value of incoming weight $|w_{ij}^l|$ of neuron o_i^l is increased only if it has a positive input y_j^{l-1} activation and a maximum output activation value y_i^l among the neighbors of neuron n_i^l. Both excitatory and inhibitory connections of the excitatory neurons are trained during learning, whereas the inhibitory neurons have constant-valued incoming connections only. The learning scheme ensures, that if an excitatory neuron has a relatively large output value, the excitatory connection weights grow faster than its inhibitory connection weights, and vice versa.

In case of feedback networks the unsupervised learning rules are similar to the rules applied in feed-forward nets. In the learning mode of the *Hopfield* net some number K of binary prototype patterns $\boldsymbol{X}^k, (k = 1, ..., K)$ is provided, each pattern is a vector of length m. The applied *Hebbian* learning rule takes the following form:

$$w_{ii} = 0, \text{ for } i = 1, ..., m; \tag{2.37}$$

$$w_{ij}(0) = 0, \quad \Delta w_{ij}(t) = \boldsymbol{X}_k[i]\ \boldsymbol{X}_k[j], \text{ for } i \neq j. \tag{2.38}$$

A relaxation network of similar class to the Hopfield net is proposed in chapter 5 for the adaptive computation of optical flow in a N-image sequence.

2.2 THE SCHEME OF ADAPTIVE N-IMAGE ANALYSIS

2.2.1 ANNs for image analysis

The main types of behavior of ANNs, from the point of view of pattern recognition theory, were distinguished in the past:

1. associative memory - recall of patterns (e.g. Hopfield net [AMA77, HOP82]),
2. classification or recognition of patterns (e.g. multi-layer network with supervised back-propagating learning [RUM86]),
3. clustering of patterns (e.g. Kohonen maps [KOH88]),
4. feature extraction (e.g. Neocognitron [FUK88]).

Due to a modified Hebbian rule for inhibitory connections it is possible to represent by SONN simultaneously multiple patterns (image features or texture objects). By further developing lateral excitatory connections - the SONNs should be able to represent image sequences [MAR91]. In the next chapters we propose our individual developments – applications of ANNs for: the restoration of m images from their N mixtures (chapter 3), the compression of images (chapter 4), single image or N-image classification (chapter 4) and visual motion estimation in N images (chapter 5).

2.2.2 The scheme

The application field of low-level image sequence analysis is reflected by the design of our scheme. The proposed adaptive analysis scheme is given as a processing pipeline of many adaptive modules (AM)(*Figure* 2.1(a)), coordinated by a general control module. A single adaptive module can process either a single image, a set of image windows obtained by scanning the image or all N images in parallel.

The general structure of each adaptive module consists of four blocks (*Figure* 2.1(b)):

- a control module for the coordination of learning and active work modes of the neural network, for parameter setting and for the synchronization of input data and the (eventual) desired patterns,
- a pre-processing module, for data re-ordering (image scanning, image to window conversion, lighting normalization, image re-scaling, masking the window with other data, etc.),
- the neural network structure for input to output transformation of the image data,
- the learning algorithm for weight training,
- a post-processing module, for output data-reordering (individual results for many image windows are converted to single image results, many image outputs may be converted to one image or data vector, etc.).

2.2.3 Module types

According to our general view on ANNs, we can distinguish four types of neural networks, that may appear in the adaptive modules:

- feed-forward nets with unsupervised learning for filter-like mapping between vector spaces – image restoration/enhancement, image compression - and clustering (e.g. PCA nets, ICA nets, SONNs),

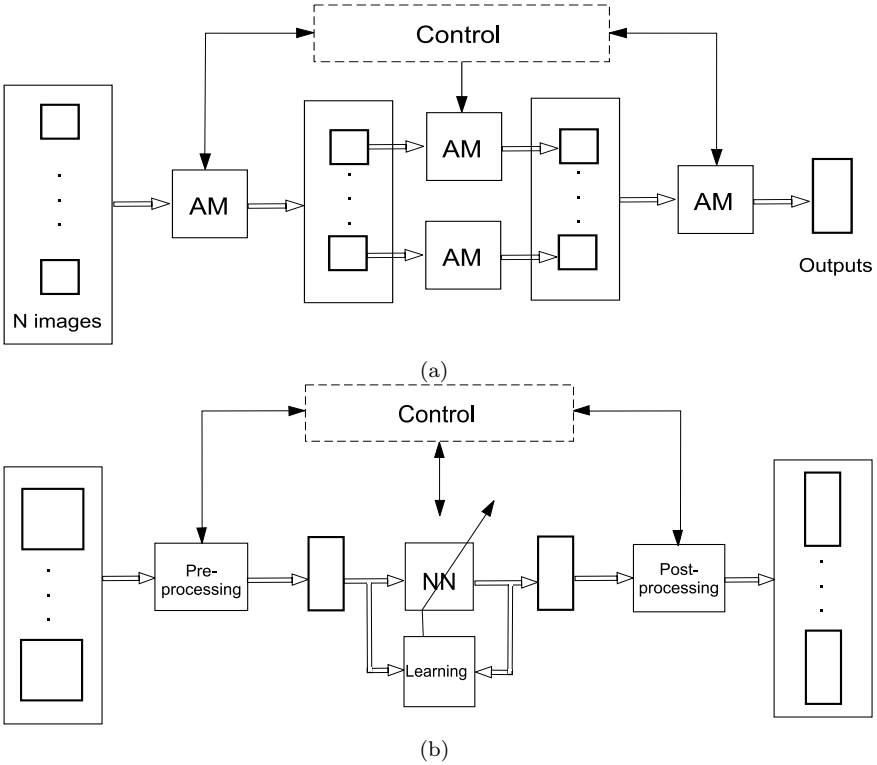


Figure 2.1 The adaptive analysis scheme for low-level analysis of N images: (a) adaptive analysis modules, (b) the structure of individual AAM.

- feed-forward nets with supervised learning for image classification and feature detection (e.g. a backpropagation net, discriminant analysis by supervised LVQ),
- feedback nets with unsupervised learning for associative memory and optic flow estimation (e.g. the Hopfield net, relaxation networks),
- feedback nets with supervised learning for correspondence problems – visual motion detection, arbiter of multiple detections or inconsistencies (e.g. the Boltzmann machine, mean field annealing).

Obviously, the module can also differ in the type of input and output structure, which require adequate pre- and post-processing stages. They perform mostly combinatorial functions but they may also include register memory (for example - if an input image is decomposed into independently analyzed windows). The following conversions are possible: (1) N-image to M-image mapping, where $M \leq N$, (2) image to image mapping, (2) image to vector mapping, (3) vector to scalar mapping.

2.2.4 Control modes

Each adaptive analysis module can work in one of two modes: *learning* mode or *active work* mode, i.e. restoration, compression, classification, motion detection etc. The learning is usually required to be done before the analysis process, if the neural network

is applied to the analysis of stationary processes, i.e. the stochastic characteristics of class detection or other analysis schemes does not change for given sample data. The analysis process can sometimes be modeled as the learning process of ANN itself, i.e. a pixel clustering process views each image as a new environment to be learned. Both learning and active mode require well–prepared, pre-processed image data be given on the input of ANN.

Preprocessing. Only some general preprocessing methods can be proposed, as this step heavily depends on the type of application field, where the images are coming from.

Learning. For example in classification, given the class labeling scheme designed by the user, the system automatically finds the basis vectors of subspaces, in order to provide the best compression and classification capabilities, indicated by the class labels associated with the training images.

Active work. After the input image is preprocessed and scanned to an input vector, and the vector is transformed by the ANN-defined transformation a comparison must be made between the result of this transformation and every class (i.e. the distance to its mean), represented in the memory of the system. The best match (i.e. the class with shortest distance to current result) is selected.

2.3 GENERAL NETWORKS FOR INTERMEDIATE- AND HIGH-LEVEL ANALYSIS

In the neural net approach, the properties of objects are modeled "holistically" in the weight parameters of an ANN. The trained networks represent implicit knowledge about the attributes required for object recognition. However, being a "holistic" system it is really not possible to build a single ANN that can cope with all possible configurations of many simultaneous objects in a complex scene. Rather multiple ANNs have to be applied to certain regions of interest, but their coordination is not yet well understood. Consequently most of the research done so far in ANNs for vision, deals with low-level visual perception. A key objective of further research in ANNs is to perform non-local operations required for the image segmentation process and symbolic attribute detection.

Up to now we have applied ANNs successfully for iconic- and segmentation-level image analysis. In connectionist systems for intermediate- and high-level vision, two other classes of non-relaxation algorithms for general networks or semantic networks, respectively, are also of interest: the *marker propagation-spreading* activation control over general networks and the *parallel derivation* rule for the representation form of semantic networks [SHA88].

Common to all semantic network approaches is the explicit structuring of the domain knowledge using a decomposition and specialization hierarchy of concepts. Knowledge about attributes and relations between parts of the concepts is modeled in an explicit way. Unfortunately, the inherent ambiguity of segment detection may lead to many

competitive interpretations for complex objects. Additionally, the acquisition and adaptation of the knowledge base requires a considerable work amount and expert skills from the system designer.

Algorithms for learning symbolic knowledge (common to the A.I. community) can be grouped into inductive and analytic methods [CAR89]. *Inductive learning* means the generalization of a model from a sequence of positive instances and known counter-examples of a concept. Examples of such an approach are: model-based learning [WIN75], induction of decision trees [QUI86] and conceptual clustering [MIC83].

Analytic methods learn from previous solutions of problems, while applying a broad knowledge about the application area. Instances of concepts are not simply sets of features, but conclusions or control rules. Examples of this approach class are: explanation–based learning [MIT86], multi–level chunking [LAI86], iterative macro-operators and learning by analogy [CAR86], and case-based learning [HAM89].

3 ICA ALGORITHMS FOR IMAGE RESTORATION

The goal of *blind source separation* (BSS) is to extract (statistically independent) unknown source signals from their linear mixtures without knowing the mixing coefficients [AMA96, BEL95, COM91, COM94, CAR96, CIC94, KAR97, OJA97, YAN97]. This blind signal processing technique has applications in data communication, speech processing and biomedical signal processing problems (MEG-EEG data). In this section, this technique will be applied for restoration of many-image sets.

The study of BSS and the related single channel *blind deconvolution* problem began some 10-15 years ago mainly in the area of statistical signal processing. Quite recently, BSS has become a highly popular research topic in unsupervised neural learning. Neural network researchers have approached the BSS problem from different starting points, such as information theory [AMA96, BEL95] and nonlinear generalizations of Hebbian / anti–Hebbian learning rules [CIC94a, JUT91, KAR97, MOR96]. Although many neural learning algorithms have been proposed for the BSS problem, in their corresponding models and network architectures it is usually assumed that the number of source signals is known a priori. Typically it should be equal to the number of sensors and outputs. However, in practice these assumptions do not often hold.

In recent papers, the author and his co-authors have proposed various neural network architectures and associated adaptive learning algorithms [KAS96a, CIK96a, KAK97, CIK99, KAS97a]. This chapter summarizes the main achievements of author's work on the subject of adaptive solutions to the BSS problem for multi-image sources.

3.1 THE BLIND SOURCE SEPARATION PROBLEM

At first, the BSS problem will be defined, some main solutions of related work will be referred to and the suitability of such methods will be discussed for handling cases where the number of sources is unknown.

3.1.1 The basic BSS problem

Assume that there exist m zero mean source signals $s_1(t), \ldots, s_m(t)$ that are scalar-valued and mutually (spatially) statistically independent (or as independent as possible) at each time instant or index value t. The original sources $s_i(t)$ are unknown to the observer, who has to deal with n possibly noisy but different linear mixtures $x_1(t), \ldots, x_n(t)$ of the sources (usually for $n \geq m$). The mixing coefficients are some unknown constants. The task of blind source separation is to find the waveforms $\{s_i(t)\}$ of the sources, knowing only the mixtures $x_j(t)$ and usually the number m of sources.

Denote by $\boldsymbol{x}(t) = [x_1(t), \ldots, x_n(t)]^T$ the n-dimensional t-th data vector made up of the mixtures at discrete index value (usually time) t. The BSS mixing model can then

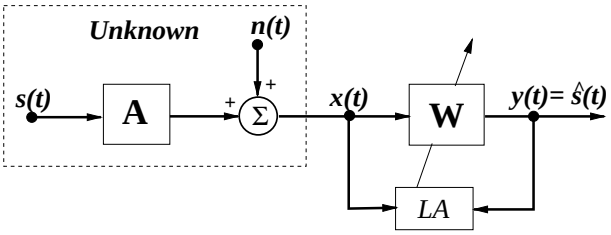


Figure 3.1 The mixing model and neural network for blind separation. LA means learning algorithm.

be written in the vector form

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{n}(t) = \sum_{i=1}^{m} s_i(t)\boldsymbol{a}_i + \boldsymbol{n}(t). \tag{3.1}$$

Here $\boldsymbol{s}(t) = [s_1(t), \ldots, s_m(t)]^T$ is the source vector consisting of the m source signals at the index value t. Furthermore, each source signal $s_i(t)$ is assumed to be a stationary zero mean stochastic process. $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m]$ is a constant full-rank $n \times m$ mixing matrix whose elements are the unknown coefficients of the mixtures (for $n \geq m$). The vectors $\boldsymbol{a}_i$ are basis vectors of *Independent Component Analysis* (ICA) [COM91, COM94, KAR97].

Bearing in mind the above general case, let us start with the noise–free simplified mixing model, where the additive noise $\boldsymbol{n}(t)$ is negligible so that it can be omitted from consideration. Let us assume further that in the general case the noise signal has a Gaussian distribution but none of the sources is Gaussian. In the simplified case at most one of the source signals $s_i(t)$ is allowed to have a Gaussian distribution. These assumptions follow from the fact that it is impossible to separate several Gaussian sources from each other [CAO96].

In standard neural and adaptive source separation approaches, an $m \times n$ separating matrix $\boldsymbol{W}(t)$ is updated so that the m-vector

$$\boldsymbol{y}(t) = \boldsymbol{W}(t)\boldsymbol{x}(t) \tag{3.2}$$

becomes an estimate $\boldsymbol{y}(t) = \hat{\boldsymbol{s}}(t)$ of the original independent source signals. *Figure* 3.1 shows a schematic diagram of the mixing and source separation system. In neural realizations, $\boldsymbol{y}(t)$ is the output vector of the network and the matrix $\boldsymbol{W}(t)$ is the total weight matrix between the input and output layers.

There are following limitations (indeterminancy) of the solution: (1) it is impossible to determine the amplitudes of sources without additional assumptions and (2) the estimated source signals $\boldsymbol{y}(t) = \hat{\boldsymbol{s}}(t)$ could be ordered differently than the primary source signals $\boldsymbol{s}(t)$.

Hence, the estimate $\hat{s}_i(t)$ of the i-th source signal may appear in any component $y_j(t)$ of $\boldsymbol{y}(t)$. The amplitude of the source signals $s_i(t)$ and their estimates $y_j(t)$ are typically scaled so that they have unit variance. This ambiguity can be expressed mathematically as:

$$\boldsymbol{y}(t) = \hat{\boldsymbol{s}}(t) = \boldsymbol{W}\boldsymbol{A}\boldsymbol{s}(t) \simeq \boldsymbol{P}\boldsymbol{D}\boldsymbol{s}(t), \tag{3.3}$$

where $\boldsymbol{P}$ is a permutation matrix and $\boldsymbol{D}$ is a nonsingular scaling matrix. If required, after convergence of $\boldsymbol{W}$ the original mixing matrix can be estimated from the pseudo-inverse operation:

$$\hat{\boldsymbol{A}} = (\boldsymbol{W}^T\boldsymbol{W})^{-1}\boldsymbol{W}^T. \tag{3.4}$$

3.1.2 Approaches to BSS

With a neural realization in mind, it is desirable to choose the learning algorithms so that they are as simple as possible but yet provide sufficient performance. Many different neural separating algorithms have been proposed recently [JUT91, CIC94, AMA96, CAR96, KAR94], [MOR96, OJA97, YAN97].

Herault & Jutten first proposed a neural rule for the BSS problem [COM91, JUT91, COM94] – a generalized Hebbian rule:

$$\frac{dw_{ij}}{dt} = -\eta(t)f[y_i(t)]g[y_j(t)], \quad \text{and } w_{ii}(t) = 1 \quad (\forall i), \tag{3.5}$$

where $f(y), g(y)$ are odd nonlinear functions. This learning algorithm can be generalized to matrix form:

$$\frac{d\boldsymbol{W}}{dt} := -\eta(t)\left[\boldsymbol{f}[\boldsymbol{y}(t)]\boldsymbol{g}^T[\boldsymbol{y}(t)]\right], \quad \text{with } w_{ii}(t) = 1, \quad (\forall i). \tag{3.6}$$

Cardoso & Laheld [CAR96] considered a two-stage approach, with *whitening* first and a subsequent *rotation* by a unitary matrix. These two steps were then combined into a single adapting rule:

$$\Delta\boldsymbol{W} = -\eta(t)\left[\boldsymbol{y}(t)\ \boldsymbol{y}^T(t) - \boldsymbol{I} + \boldsymbol{g}[\boldsymbol{y}(t)]\boldsymbol{y}^T(t) - \boldsymbol{y}(t)\boldsymbol{g}[\boldsymbol{y}^T(t)]\right]\boldsymbol{W}(t). \tag{3.7}$$

For stability a self–normalized version was proposed:

$$\Delta\boldsymbol{W} = -\boldsymbol{\mu}(t)\left[\frac{\boldsymbol{y}(t)\ \boldsymbol{y}^T(t) - \boldsymbol{I}}{1 + \boldsymbol{\mu}(t)\boldsymbol{y}^T(t)\boldsymbol{y}(t)} + \frac{\boldsymbol{g}[\boldsymbol{y}(t)]\boldsymbol{y}^T(t) - \boldsymbol{y}(t)\boldsymbol{g}[\boldsymbol{y}^T(t)]}{1 + \boldsymbol{\mu}(t)\boldsymbol{y}^T(t)\boldsymbol{g}[\boldsymbol{y}(t)]}\right]\boldsymbol{W}(t). \tag{3.8}$$

Karhunen, Oja et al. [KAR97] proposed a two–layer network for PCA type separation, with the first whitening layer and the second separation layer. The total separation matrix is given as:

$$\boldsymbol{B} = \boldsymbol{W}^T\boldsymbol{V}, \tag{3.9}$$

where $\boldsymbol{W}^T$ denotes the orthogonal ($\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$) separating matrix applied to the whitened vectors $\boldsymbol{v}$. The effects of second–order statistics on the nonlinearities are removed by *whitening* the data first:

$$\boldsymbol{v}(t) = \boldsymbol{V}\boldsymbol{x}(t). \tag{3.10}$$

A simple neural algorithm for computing $\boldsymbol{V}$ is [KAR97]:

$$\boldsymbol{V}(t+1) = \boldsymbol{V}(t) - \eta(t)[\boldsymbol{v}(t)\boldsymbol{v}^T(t) - \boldsymbol{I}]\boldsymbol{V}(t) \tag{3.11}$$

For the separation step Karhunen et al. [KAR97] proposed a bi–gradient algorithm:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \eta(t)\boldsymbol{v}(t)\boldsymbol{g}[\boldsymbol{y}^T(t)] + \eta_1(t)\boldsymbol{W}(t)[\boldsymbol{I} - \boldsymbol{W}^T(t)\boldsymbol{W}(t)], \tag{3.12}$$

with $0.5 \leq \tilde{\eta} \leq 1.0$. This stochastic gradient algorithm maximizes the sum $\sum_{j=1}^{m} E\{\boldsymbol{f}[y(j)]\}$ assuming that the weight matrix $\boldsymbol{W}$ is orthonormal.

Bell & Sejnowski [BEL95] defined the mutual information, i.e. the information about input $\boldsymbol{x}$ that is contained in output $\boldsymbol{u}$:

$$I(\boldsymbol{u}, \boldsymbol{x}) = H(\boldsymbol{u}) - H(\boldsymbol{u}/\boldsymbol{x}), \tag{3.13}$$

where $H(\boldsymbol{u})$ is the entropy in the output, while $H(\boldsymbol{u}/\boldsymbol{x})$ is information which did not come from the input. In the noiseless case the mutual information can be maximized by maximizing the output entropy:

$$H(u) = -E[ln(F_u(u))], \text{ where } F_u(u) = \frac{F_x(x)}{|det[J]|} \tag{3.14}$$

is the probability density function, $u = f(y)$ is the nonlinear activation function, $[J]$ is the Jacobian matrix of nonlinear transformation $\boldsymbol{f}(\boldsymbol{y})$.

The gradient descent algorithm

$$\frac{\delta I(\boldsymbol{u}, \boldsymbol{x})}{\delta \boldsymbol{W}} = \frac{\delta H(\boldsymbol{u})}{\delta \boldsymbol{W}} \tag{3.15}$$

leads for sigmoidal activation functions

$$f(y) = \frac{1}{1 + e^{-y}}; \quad \boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{w}_0 \tag{3.16}$$

to the following learning rule:

$$\Delta \boldsymbol{W}(t+1) = \eta \left\{ [\boldsymbol{W}^T]^{-1} + \boldsymbol{x}(t) \left[\boldsymbol{I} - 2\boldsymbol{f}[\boldsymbol{y}(t)]\right]^T \right\} \tag{3.17}$$

$$\Delta \boldsymbol{w}_0 = \eta \left[\boldsymbol{I} - 2\boldsymbol{f}[\boldsymbol{y}(t)]\right]. \tag{3.18}$$

For a hyperbolic tangent

$$\boldsymbol{f}(\boldsymbol{y}) = \frac{1 - e^{-2y}}{1 + e^{2y}}, \tag{3.19}$$

this leads to the following learning rule:

$$\Delta \boldsymbol{W}(t+1) = \eta \left\{ [\boldsymbol{W}^T]^{-1} - 2\boldsymbol{f}[\boldsymbol{y}(t)]\boldsymbol{x}^T(t) \right\} = \eta \left[\boldsymbol{I} - 2\boldsymbol{f}[\boldsymbol{y}(t)]\boldsymbol{y}^T(t)\right] \boldsymbol{W}^{-T}. \tag{3.20}$$

Unfortunately both rules are non–neural, as they require a matrix inversion.

The performance of all learning-based approaches to BSS usually strongly depends on the stochastic properties of the source signals. These properties can be examined from the higher-order moments (cumulants) of the source signals. A very useful cumulant is a fourth–order moment, called the *kurtosis*. The *normalized kurtosis* of a source signal $s_i(t)$ is defined as follows:

$$\widehat{\kappa_4}[s_i(t)] = \frac{\mathrm{E}\{s_i(t)^4\}}{\mathrm{E}^2\{s_i(t)^2\}} - 3. \tag{3.21}$$

If the source $s_i(t)$ is a *Gaussian* signal then its kurtosis $\widehat{\kappa_4}[s_i(t)]$ is equal to zero. Source signals that have a negative kurtosis are often called *sub–Gaussian* ones. Typically, their probability distribution is "flatter" than the Gaussian distribution. Respectively, *super–Gaussian* sources (with a positive kurtosis) usually have a distribution which has a longer tail and sharper peak when compared with the Gaussian distribution. The choice of nonlinear functions in neural separating algorithms depends on the sign of the normalized kurtoses of the sources [CAR98].

3.1.3 Separation with estimation of the number of sources

A standard assumption in BSS is that the number m of the sources should be *known* in advance. Like in most neural BSS approaches, we have assumed up to now that the number m of the sources and outputs l are *equal* in the separating network. Generally, both these assumptions may not hold in practice. In the next section different approaches for neural blind separation with simultaneous determination of the source number m will be proposed. The approaches are described more deeply in the author's co-work in [CIK96a, CIK99]. The only additional requirement in these approaches is that the number of available mixtures is greater than or equal to the true number of the sources, that is, $n \geq m$.

For the completeness of our considerations, let us first briefly discuss the difficult case where there are fewer mixtures than sources: $n < m$. Then the $n \times m$ mixing matrix $\boldsymbol{A}$ in (3.1) has more columns than rows. In this case, complete separation is usually out of the question. This is easy to understand by considering the much simpler situation where the mixing matrix $\boldsymbol{A}$ is *known* (recall that in BSS this does not hold), and there is no noise. Even then the set of linear equations (3.1) has an infinite number of solutions because there are more unknowns than equations, and the source vector $\boldsymbol{s}(t)$ cannot be determined for arbitrary distributed sources. However, some kind of separation may still be achievable in special instances at least. A novel approach is proposed by the author in the last section of this chapter.

It is not always necessary or even desirable in BSS problems to separate all the sources contained in the mixtures. In situations where the number of sensors is large only a few of the most powerful sources may be of interest.

3.2 SEPARATION WITH SOURCE NUMBER DETECTION

In this section two alternative source separation approaches for solving the basic BSS problem are proposed, as derived and implemented by the author and his co-author's quite recently [KAS96a, CIK96a, KAK97]. The first approach uses pre–whitening, while the second approach tries to separate and to determine the source number directly from the input data. The theoretical basics of the proposed learning rules are given in [AMA96, CIK99]. Here let us introduce special additional techniques, such as: the estimation of the number of sources and redundancy removal among the outputs of the networks.

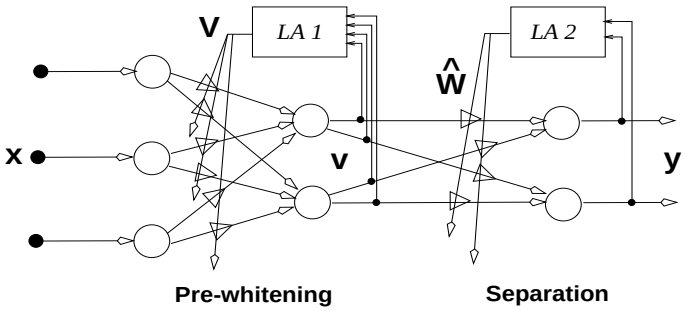


Figure 3.2 The two–layer feed–forward network for pre–whitening and blind separation – a neural network with signal reduction during pre–whitening.

3.2.1 Separation after a pre–whitening layer

Figure 3.2 shows a two-layer neural network for blind source separation, where the first layer performs *pre–whitening (sphering)* and the second one - separation of sources. The respective weight matrices are denoted by $\boldsymbol{V}$ and $\widehat{\boldsymbol{W}}$. The operation of the network is described by

$$\boldsymbol{y}(t) = \widehat{\boldsymbol{W}}(t)\boldsymbol{v}(t) = \widehat{\boldsymbol{W}}\boldsymbol{V}\boldsymbol{x}(t) = \boldsymbol{W}(t)\boldsymbol{x}(t), \tag{3.22}$$

where $\boldsymbol{W} \equiv \widehat{\boldsymbol{W}}\boldsymbol{V}$ is the total separating matrix.

The network of *Figure* 3.2 is useful in context with such BSS algorithms that require whitening of the input data for good performance. In *whitening (sphering)*, the data vectors $\mathbf{x}(t)$ are pre-processed using a whitening transformation

$$\boldsymbol{v}(t) = \boldsymbol{V}(t)\boldsymbol{x}(t). \tag{3.23}$$

Here $\boldsymbol{v}(t)$ denotes the whitened vector, and $\boldsymbol{V}(t)$ is an $m \times n$ whitening matrix.

If $n > m$, where m is known in advance, $\boldsymbol{V}(t)$ simultaneously reduces the dimension of the data vectors from n to m. In whitening, the matrix $\boldsymbol{V}(t)$ is chosen so that the covariance matrix $\mathrm{E}\{\boldsymbol{v}(t)\boldsymbol{v}(t)^T\}$ becomes the unit matrix $\boldsymbol{I}_m$. Thus the components of the whitened vectors $\boldsymbol{v}(t)$ are mutually uncorrelated and they have unit variance. Uncorrelatedness is a necessary condition for the stronger independence condition. After pre-whitening the separation task usually becomes easier, because the subsequent separating matrix $\widehat{\boldsymbol{W}}$ can be constrained to be orthogonal [KAR97, OJA97]:

$$\widehat{\boldsymbol{W}}\widehat{\boldsymbol{W}}^T = \boldsymbol{I}_m, \tag{3.24}$$

where $\boldsymbol{I}_m$ is the $m \times m$ unit matrix. Whitening seems to be especially helpful in large-scale problems, where separation of sources can sometimes be impossible in practice without the reduction of the separation matrix size.

Neural learning rules for pre–whitening

There exist many solutions for whitening the input data [KAR97, CAR96, SIL91]. Two simple adaptive, on–line learning rules for pre–whitening have the following matrix

forms:

$$\boldsymbol{V}(t+1) = \boldsymbol{V}(t) + \eta(t)\left[\boldsymbol{I} - \boldsymbol{v}(t)\boldsymbol{v}^T(t)\right], \tag{3.25}$$

$$\boldsymbol{V}(t+1) = \boldsymbol{V}(t) + \eta(t)\left[\boldsymbol{I} - \boldsymbol{v}(t)\boldsymbol{v}^T(t)\right]\boldsymbol{V}(t). \tag{3.26}$$

The first algorithm is a local one, in the sense that the update of every weight v_{ij} is made on the basis of two neurons i and j only. The second algorithm is a robust one with an *equivariant property* [CAR96] as the global system (in the sense that the update of every synaptic weight v_{ij} depends on outputs of all neurons), described by a combined mixing and de–correlation process:

$$\boldsymbol{P}(t+1) \stackrel{df}{=} \boldsymbol{V}(t+1)\boldsymbol{A} = \boldsymbol{P}(t) + \eta(t)\left[\boldsymbol{I} - \boldsymbol{P}(t)\boldsymbol{s}(t)\boldsymbol{s}^T(t)\boldsymbol{P}^T(t)\right]\boldsymbol{P}(t), \tag{3.27}$$

is completely independent from the mixing matrix $\boldsymbol{A}$. Both these pre–whitening rules can be used in context with neural separating algorithms.

Nonlinear Principal Subspace learning rule for the separation layer

The *Nonlinear PCA subspace rule* developed and studied by Oja, Karhunen, Xu and their collaborators (see [KAR94, OJA97]) employs the following update rule for the orthogonal separating matrix $\widehat{\boldsymbol{W}}$:

$$\widehat{\boldsymbol{W}}(t+1) = \widehat{\boldsymbol{W}}(t) + \eta(t)\boldsymbol{g}[\boldsymbol{y}(t)]\left[\boldsymbol{v}(t) - \widehat{\boldsymbol{W}}^T(t)\boldsymbol{g}[\boldsymbol{y}(t)]\right]^T, \tag{3.28}$$

where $\boldsymbol{v}(t) = \boldsymbol{V}(t)\boldsymbol{x}(t)$, $\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t)$, and $\boldsymbol{y}(t) = \widehat{\boldsymbol{W}}(t)\boldsymbol{v}(t)$. Vector $\boldsymbol{g}[\boldsymbol{y}(t)]$ denotes a column vector whose i–th component is $g_i[y_i(t)]$, where $g_i(t)$ is usually an odd and monotonically increasing nonlinear activation function. The learning rate $\eta(t)$ must be positive for stability reasons. A major advantage of the learning rule (3.28) is that it can be realized using a simple modification of one–layer standard symmetric PCA network, allowing a relatively simple neural implementation [KAR94, KAR97]. The separation properties of (3.28) have been analyzed mathematically in simple cases in [OJA97].

Signal number reduction by pre–whitening

The first class of approaches for source number determination in the BSS problem is based on the natural compression ability of the pre–whitening layer. If standard Principal Component Analysis (PCA) is used for pre–whitening, one can then simultaneously compress information optimally in the mean-square error sense and filter the possible noise [KAR97]. In fact the PCA whitening matrix $\boldsymbol{V}$ can be computed as

$$\boldsymbol{V} = \sqrt{\boldsymbol{R}_{xx}^{-1}} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{U}^T, \tag{3.29}$$

where $\boldsymbol{\Lambda} = diag(\lambda_1, ..., \lambda_n)$ is a diagonal matrix of the eigenvalues and $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_n]$ is the orthogonal matrix of the associated eigenvectors of the covariance matrix $\boldsymbol{R}_{xx} = E\left[\boldsymbol{x}(t)\boldsymbol{x}^T(t)\right] = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$.

If there are more mixtures than sources ($n > m$), it is possible to use the PCA approach for estimating the number m of the sources. If m is estimated correctly and the

input vectors $\boldsymbol{x}(t)$ are compressed to m–dimensional vectors $\boldsymbol{v}(t)$ in the whitening stage using the network structure in *Figure* 3.2, then there are usually no specific problems in the subsequent separation stage.

In practice, the source number is determined by first estimating the eigenvalues λ_i of the data covariance matrix $\mathrm{E}\{\mathbf{x}(t)\mathbf{x}(t)^T\}$. Let us denote these ordered eigenvalues by

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0. \tag{3.30}$$

In the ideal case where the noise term $\mathbf{n}(t)$ in (3.1) is zero, only the m largest "signal" eigenvalues $\lambda_1, \ldots, \lambda_m$ are nonzero, and the rest $n - m$ "noise" eigenvalues of the data covariance matrix are zero. If the powers of the sources are much larger than the power of noise, the m largest signal eigenvalues are still clearly larger than the noise eigenvalues, and it is straightforward to determine m from the breakpoint. However, if some of the sources are weak or the power of the noise is not small, it is generally difficult to judge what is the correct number m of sources just by inspecting the eigenvalues.

Let us assume a modified network structure, where the possible data compression takes place in the separation layer instead of the pre–whitening layer. Generally such a modified network structure is not recommendable if the power of the noise is not small or the number of mixtures n is larger than the number m of the sources. This is easy to understand, because in this case whitening without data compression tends to amplify the noise by making the variances of n components of the whitened vectors $\mathbf{v}(t)$ all equal to unity.

3.2.2 Source separation without pre–whitening

The disadvantage of whitening is that for ill-conditioned mixing matrices and weak sources the separation results may be poor. Therefore, some other neural algorithms have been developed that learn the separating matrix $\boldsymbol{W}$ directly. A single layer performs the linear transformation

$$\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t), \tag{3.31}$$

where $\boldsymbol{W}$ is an $n \times n$ square nonsingular matrix of synaptic weights updated according to some on–line learning rule. In this section we discuss simple neural network models and associated adaptive learning algorithms, which do not require any pre–processing.

General (robust) global rule

The whitening algorithms discussed so far can be easily generalized for the blind source separation problem. For example, a general form of the learning rule (3.26) was proposed as [CIC94a]:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \eta(t)\left\{\boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)]\boldsymbol{g}[\boldsymbol{y}^T(t)]\right\}\boldsymbol{W}(t), \tag{3.32}$$

where $\eta(t) > 0$ is the adaptive learning rate and $\boldsymbol{I}$ is the $n \times n$ identity matrix. $\boldsymbol{f}(\boldsymbol{y})$ $= [f(y_1), ..., f(y_n)]^T$ and $\boldsymbol{g}(\boldsymbol{y}^T) = [g(y_1), ..., g(y_n)]$ are vectors of nonlinear activation functions, where $f(y), g(y)$ is a pair of suitably chosen nonlinear functions.

These nonlinear functions are used in the above rule mainly for introducing higher–order statistics or cross–cumulants into computations. The rule tries to cancel these

higher-order statistics, leading to an at least approximate separation of sources (or independent components). The choice of the activation functions $f(y), g(y)$ depends on the statistical distribution of the source signals.

The above rule can be derived more rigorously by using the concept of *Kullback–Leibler* divergence (or mutual information) and the *natural gradient* concept, as proposed in [AMA96]. The dependency among output signals is measured by the Kullback–Leibler divergence between the joint and the product of the marginal distribution of the outputs:

$$D(\boldsymbol{W}) = \int p(\boldsymbol{y}) \log \frac{p(\boldsymbol{y})}{\Pi_{a=1}^{n} p_a(y^a)} d\boldsymbol{y}, \tag{3.33}$$

where $p_a(y^a)$ is the marginal probability density function (pdf). This Kullback–Leibler divergence D($\boldsymbol{W}$) is related to $H(\boldsymbol{y})$ - the average mutual information of the outputs $\boldsymbol{y}$ - by:

$$D(\boldsymbol{W}) = -H(\boldsymbol{y}) + \sum_{a=1}^{n} H(y^a) \tag{3.34}$$

where $H(\boldsymbol{y}) = -\int p(\boldsymbol{y}) \log[p(\boldsymbol{y})] d\boldsymbol{y}$, while $H(y^a) = -\int p_a(y^a) \log[p_a(y^a)] dy^a$ - is the marginal entropy.

To calculate each $H(y^a)$ the *Gram-Charlier* expansion can be applied in order to approximate the probability density function $p_a(y^a)$. Next, by applying the theory of higher order cumulants to the gradient descent algorithm:

$$\frac{d\boldsymbol{W}}{dt} = -\eta(t) \frac{\delta D}{\delta \boldsymbol{W}}, \tag{3.35}$$

the following learning algorithm is derived:

$$\Delta \boldsymbol{W}(t+1) = \eta(t) \left[\boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)] \; \boldsymbol{y}^T(t) \right] \boldsymbol{W}^{-T}(t). \tag{3.36}$$

The nonlinear activation function was also directly derived during these transformations, to be approximated by a non-monotonic function [AMA96]:

$$f(y) = \frac{3}{4} y^{11} + \frac{25}{4} y^9 - \frac{14}{3} y^7 - \frac{47}{4} y^5 + \frac{29}{4} y^3. \tag{3.37}$$

Unfortunately, the above rule requires a matrix inversion, which makes it non–neural. To improve its behavior, by observing that the mixing matrix is nonsingular, a better gradient descent algorithm form may be used:

$$\frac{d\boldsymbol{W}}{dt} = -\eta(t) \frac{\delta D}{\delta \boldsymbol{W}} \boldsymbol{W}^T \boldsymbol{W}, \tag{3.38}$$

which finally leads to the above neural rule (3.32).

Simplified (local) rule

The learning rule (3.32) can be simplified by applying another generalized gradient form:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) \mp \eta(t) \frac{\partial D}{\partial \boldsymbol{W}} \boldsymbol{W}^T(t). \tag{3.39}$$

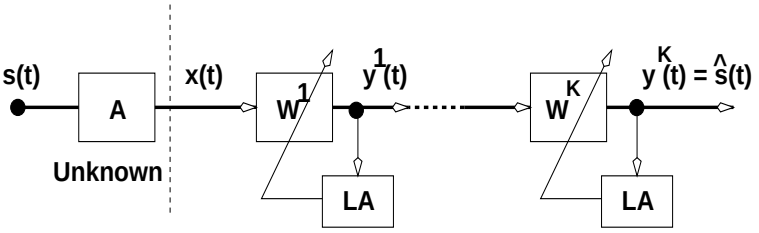


Figure 3.3 A multi–layer neural network architecture for blind source separation without pre–whitening.

In this case we obtain a relatively simple self–normalized local learning rule [CIC94]:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) \pm \eta(t) \left\{ \boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)] \boldsymbol{y}^T(t) \right\}. \tag{3.40}$$

This learning rule is stable for both signs + and − under zero initial conditions. The local learning rule (3.40) can be regarded as a generalization of the local whitening rule (3.25). Furthermore, this is the simplest learning rule for the BSS problem that to our knowledge has been proposed thus far.

Equivariant property

It may be observed that the learning rule (3.32) has a so–called *equivariant* property [AMA96, CAR96]. This means that its performance is independent of the scaling factors and mixing matrix $\boldsymbol{A}$. To show this, multiply both sides of the equation (3.32) by the nonsingular fixed matrix $\boldsymbol{A}$:

$$\boldsymbol{W}(t+1)\boldsymbol{A} = \boldsymbol{W}(t)\boldsymbol{A} + \eta(t) \left\{ \boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)] \boldsymbol{g}[\boldsymbol{y}^T(t)] \right\} \boldsymbol{W}(t)\boldsymbol{A}. \tag{3.41}$$

Hence, the difference equation appears:

$$\boldsymbol{P}(t+1) = \boldsymbol{P}(t) + \eta(t) \left\{ \boldsymbol{I} - \boldsymbol{f}[\boldsymbol{P}(t)\boldsymbol{s}(t)] \boldsymbol{g}^T[\boldsymbol{P}(t)\boldsymbol{s}(t)] \right\} \boldsymbol{P}(t), \tag{3.42}$$

which describes the combined mixing and de–mixing system. It should be noted that the matrix

$$\boldsymbol{P}(t) \equiv \boldsymbol{W}(t)\boldsymbol{A} \tag{3.43}$$

describing the global system is explicitly independent of the mixing matrix $\boldsymbol{A}$.

Therefore, the algorithm is able to extract extremely weak signals mixed with strong ones provided that there is no noise. Moreover, the condition number of mixing matrix can then be even 10^{15}, and it depends only on the precision of the calculations [CIC94a].

The simplified local learning rule (3.40) does not have the equivariant property. Hence, a single layer neural network with this learning rule may sometimes fail to separate signals, especially if the problem is ill–conditioned. However, by applying a multi–layer structure this algorithm is also able to solve very ill–conditioned separation problems. In such a case the same simple local learning rule (3.40) is applied for each layer, as illustrated by *Figure* 3.3. However, for each layer different nonlinear functions can be used for introducing different higher-order statistics, which usually improves the quality of separation.

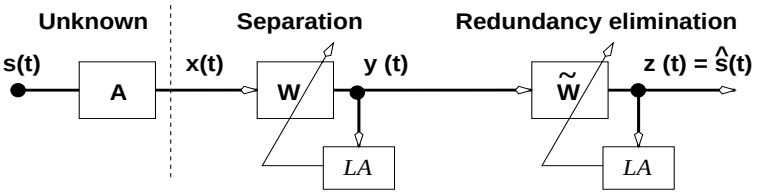


Figure 3.4 The scheme of a two–layer neural network for blind separation and redundancy elimination.

Simultaneous separation and redundancy reduction in small noise

The separation algorithms (3.32) and (3.40) presented so far for the complete (determined) source case ($m = n$) can be applied in the more general (over-determined) case, when the number of sources is unknown, but not larger than the number of sensors, that is if $n \geq m$. In this case the dimension of matrix $\boldsymbol{W}(t)$ is still $n \times n$. If $n > m$ there appears a redundancy among the separated signal set, meaning that one or more signals are extracted in more than one channel. If additive noises exist in each sensor channel, then they appear on the redundant outputs.

Signal redundancy reduction in a noise–free case

Let us consider the (somehow artificial) noise–free case (basic BSS) in over-determined mixing. Then some separated signals appear in different channels with different scaling factors. In [CIK96a] it was proposed to add a post–processing layer to the separation network for the elimination of redundant signals. Thus the applied neural network consists of two or more layers (*Figure* 3.4), where the first sub–network (a single layer or a multi–layer) simultaneously separates the sources and the last (post–processing) layer eliminates redundant signals. The post–processing layer determines the number of active sources in the case where the number of sensors (mixtures) n is greater than the number of the primary sources m. Such a layer is described by the linear transformation $\boldsymbol{z}(t) = \widetilde{\boldsymbol{W}}(t)\boldsymbol{y}(t)$, where the synaptic weights (elements of the matrix $\widetilde{\boldsymbol{W}}(t)$) are updated using the following adaptive local learning algorithm:

$$\begin{array}{rcl} \widetilde{w}_{ii}(t) & = & 1, \quad \forall t \; \forall i \\ \Delta\widetilde{w}_{ij}(t) & = & -\eta(t) f[z_i(t)] g[z_j(t)], \quad i \neq j, \end{array} \tag{3.44}$$

where $g(z)$ is a nonlinear odd activation function (e.g. $g(z) = \tanh(\alpha z)$) and $f(z)$ is either a linear or slightly nonlinear odd function.

The learning rules for redundancy elimination given above can be derived using the same optimization criterion which was used for source separation (eqs. (3.32), (3.40)), but with some constraints for the elements of the matrix $\tilde{\boldsymbol{W}}(t)$, e.g. $\tilde{w}_{ii}(t) = 1, \forall i$. It should be noted that the rule (3.44) is similar to the well–known Herault–Jutten rule [JUT91], but it is now applied to a feed–forward network with different activation functions.

Pos.	Separation rule	Description
1	$\Delta \boldsymbol{W} = \eta \left[\boldsymbol{I} - \boldsymbol{f}(\boldsymbol{y}) \boldsymbol{g}(\boldsymbol{y})^T \right] \boldsymbol{W}$	Global algorithm with equivariant property $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} = \boldsymbol{W}\boldsymbol{A}\boldsymbol{s}$
2	$\Delta \boldsymbol{W} = \pm\eta \left[\boldsymbol{I} - \boldsymbol{f}(\boldsymbol{y}) \boldsymbol{g}(\boldsymbol{y})^T \right]$	Simple local algorithm $\boldsymbol{y}^l = \boldsymbol{W}^l \boldsymbol{y}^{l-1}, \quad \boldsymbol{y}^0 = \boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$
3	$\Delta \boldsymbol{W} = \eta \boldsymbol{f}(\boldsymbol{y}) \left[\boldsymbol{v}^T - \boldsymbol{f}(\boldsymbol{y})^T \boldsymbol{W} \right] \simeq \eta \left[\boldsymbol{f}(\boldsymbol{y}) \boldsymbol{y}^T - \boldsymbol{y} \boldsymbol{f}(\boldsymbol{y})^T \right] \boldsymbol{W}$	Nonlinear PCA with pre-whitening $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{v}, \qquad \boldsymbol{v} = \boldsymbol{V}\boldsymbol{x} = \boldsymbol{V}\boldsymbol{A}\boldsymbol{s}$

Table 3.1 Three tested separation algorithms.

3.2.3 Selection of activation functions

The choice of the activation function (nonlinearity) depends on the sign of *normalized kurtosis* (3.21) of the source signals. It has recently been shown [CAR98] that this is sufficient for successful separation, though a knowledge of the probability densities of the sources would help to achieve a better accuracy. In blind separation, these densities are usually unknown. If the source signals are expected to have negative kurtosis values, that is, they are *sub–Gaussian* signals, we choose in the global algorithm (3.32)

$$f(y_j) = y_j^3 \qquad \text{and} \qquad g(y_j) = y_j, \tag{3.45}$$

$$\text{or} \quad f(y_j) = y_j^3 \qquad \text{and} \qquad g(y_j) = \tanh(\alpha y_j). \tag{3.46}$$

On the other hand, for *super–Gaussian* sources with positive kurtosis, we choose

$$f(y_j) = \tanh(\alpha y_j) \qquad \text{and} \qquad g(y_j) = y_j, \tag{3.47}$$

$$\text{or} \quad f(y_j) = \tanh(\alpha y_j) \qquad \text{and} \qquad g(y_j) = y_j^3, \tag{3.48}$$

for obtaining successful separation.

When the source signals have both positive and negative kurtosis values, a combination of the above functions can be applied. If the signs of the kurtosises are unknown, one can estimate them adaptively fairly easily during the separation process. In the case of the local rule, the first choices of the nonlinearities given above are applied for both types of sources. For the nonlinear PCA rule (3.28) the following holds:
- $g_i[y_i] = \tanh(\alpha y_i)$ - for sub-Gaussian sources and
- $g_i[y_i] = y_i + \tanh(\alpha y_i)$ (or $g_i(y_i) = y_i^3$) - for super-Gaussian sources.

3.2.4 Testing the two solutions

In this subsection some experiments are presented using the three considered separation algorithms: the global algorithm with equivariant property (3.32), the local algorithm (3.40) and the nonlinear PCA subspace rule (3.28) with prewhitening, as summarized in *Table* 3.1. The separation results are evaluated by comparing images showing the true sources and the separated sources. This gives a good qualitative assessment of the achieved performance. Different types of image sources are applied in a single experiment – sources with both positive or negative kurtosis and a Gaussian noise image are mixed together.

In our experiments, natural or synthetic grey-scale images (with 256 grey levels) are used; their size is equal to 256×384 and 256×256. Before the start of the learning

procedure the image signals should be transformed to zero–mean signals, and for compatibility with the learning rate and initial weights they are also scaled to the interval $[-1.0, 1.0]$. For presentation, the resulting signals are mapped back to the grey–level interval of $[0, 255]$. Zero signals having small amplitudes around 0.0 correspond to uniformly grey images.

The obtained results can be assessed quantitatively by using suitable mathematical measures, like PSNR (peak signal-to-noise ratio) between each reconstructed source and the corresponding original source, and an error index EI for the whole set of separated sources:

$$EI = \frac{1}{m}\left[\left(\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{|p_{ij}|^2}{\max_i |p_{ij}|^2}\right) - n\right] + \frac{1}{n}\left[\left(\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{|p_{ij}|^2}{\max_j |p_{ij}|^2}\right) - m\right]. \qquad (3.49)$$

The numbers p_{ij} are entries of a normalized matrix $\boldsymbol{P}(t)$, derived from $\hat{\boldsymbol{P}}(t) \in R^{n\times m}$:

$$\hat{\boldsymbol{P}} = \widetilde{\boldsymbol{W}}\boldsymbol{W}^{(k)}...\boldsymbol{W}^{(1)}\boldsymbol{V}\boldsymbol{A},$$

by normalizing every non–zero row $i = 1, ..., n$ of the matrix $\hat{\boldsymbol{P}}$ in such a way that $\max_i |p_{ij}| = 1$. In the ideal case of perfect separation, the matrix $\boldsymbol{P}$ becomes a permutation matrix. Then only one of the elements on each row and column equals to unity, and all the other elements are zero. In this ideal case EI achieves its minimum possible value 0.

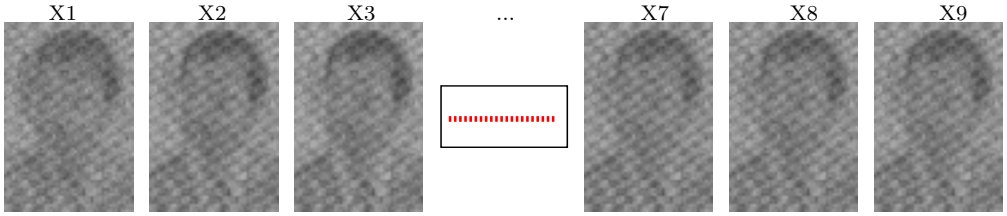
In order to achieve fast convergence of the weights during the learning process, we apply a descending learning rate. The initial step-size $\eta(t)$ depends on the expected signal amplitude and the initial values of $\boldsymbol{W}$. We use a descending $\eta(t)$ which is the largest possible, providing a fast learning and convergence of the algorithm. Usually for signal amplitudes in the interval $[-1, 1]$ and initial weight values < 1, η is below 0.1. The initial matrix $\boldsymbol{W}(0)$ is a non-zero random matrix with elements scaled to the interval $[-1, 1]$. In every experiment the final weights W_{lim} are taken for computation of the quantitative results.

Reduction during pre-whitening

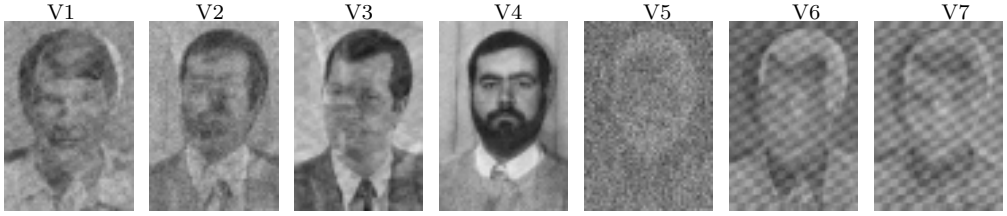
In the first experiment the two–layer ANN of *Figure* 3.2 is applied, performing first the source number estimation and then a consecutive separation. The mixing takes place without additive noise (although one of the source signals is itself a noise signal) and there are fewer sources than sensors (but more than two sources exist). As shown in *Figure* 3.5, seven images were mixed by using a randomly generated, ill–conditional mixing matrix $\boldsymbol{A}_{9\times 7}$. The source set consists of four natural face images with negative normalized kurtosis κ_4, one noise image with negative κ_4, one synthetic image with positive κ_4 and one natural image with slightly positive κ_4.

The source images and mixtures used in these experiments are rather demanding for separation algorithms, because for them the assumptions made on the data model (1) are actually not valid. This is because the applied face images are clearly correlated (see *Table* 3.2), with locally changing stochastic properties. Thus the source signals are not

(a) 7 sources S1–S7



(b) 9 mixtures X1–X9



(c) 7 pre–whitened signals V1–V7

(d) 7 separated signals Y1–Y7 (error index EI=0.31; PSNR [dB]:
26.9 23.2 22.6 19.9 33.0 20.9 34.2)

Figure 3.5 Example of correct source number determination in PCA based pre–whitening and with consecutive separation.

independent, but they are non-stationary also – some signal samples are more and some less correlated to each other. During the learning process we can select less correlated output samples than the overall correlation factor is, thus concentrating the learning process on approximately half the amount of signal samples.

It is remarkable that one is able to achieve sufficiently good quality separation for practical purposes, especially by the algorithms (3.32) and (3.40) which do not use pre–whitening. Generally, algorithms applying pre–whitening are not able to separate the clearly correlated face images. This follows from the fact that the pre-whitened data vectors $\boldsymbol{v}(t)$ are uncorrelated and in the separation phase this property is preserved.

Signals	Signal–pair correlations (×100)								
	Sources								
	S1- 2	S1- 3	S1- 4	S1- 5	S2- 3	S2- 4	S2- 5	S3- 4	S3- 5	S4- 5
$\boldsymbol{s}$	21.3	44.9	32.7	1.27	35.5	21.5	0.75	41.5	0.39	0.84
	Sensor (mixed) signals								
	X1-2	X1-3	X1-4	X1-5	X2-3	X2-4	X2-5	X3-4	X3-5	X4-5
$\boldsymbol{x}$	98.6	98.8	98.4	99.6	97.3	97.5	99.2	99.6	99.4	99.2
	Separated (output) signals								
	y1-2	y1-3	y1-4	y1-5	y2-3	y2-4	y2-5	y3-4	y3-5	y4-5
	After pre–whitening and nonlinear PCA separation								
$\boldsymbol{y}$	-0.26	10.0	-0.28	1.04	-1.60	-1.68	-1.78	-2.84	-9.593	-9.36
	After local rule separation								
$\boldsymbol{y}^{(5)}$	-7.84	5.37	-2.52	6.47	2.99	3.99	-0.93	-2.70	-3.17	-3.88
	After global rule separation								
$\boldsymbol{y}$	10.2	14.3	13.0	-0.61	12.1	10.2	0.25	19.7	-0.98	-0.24

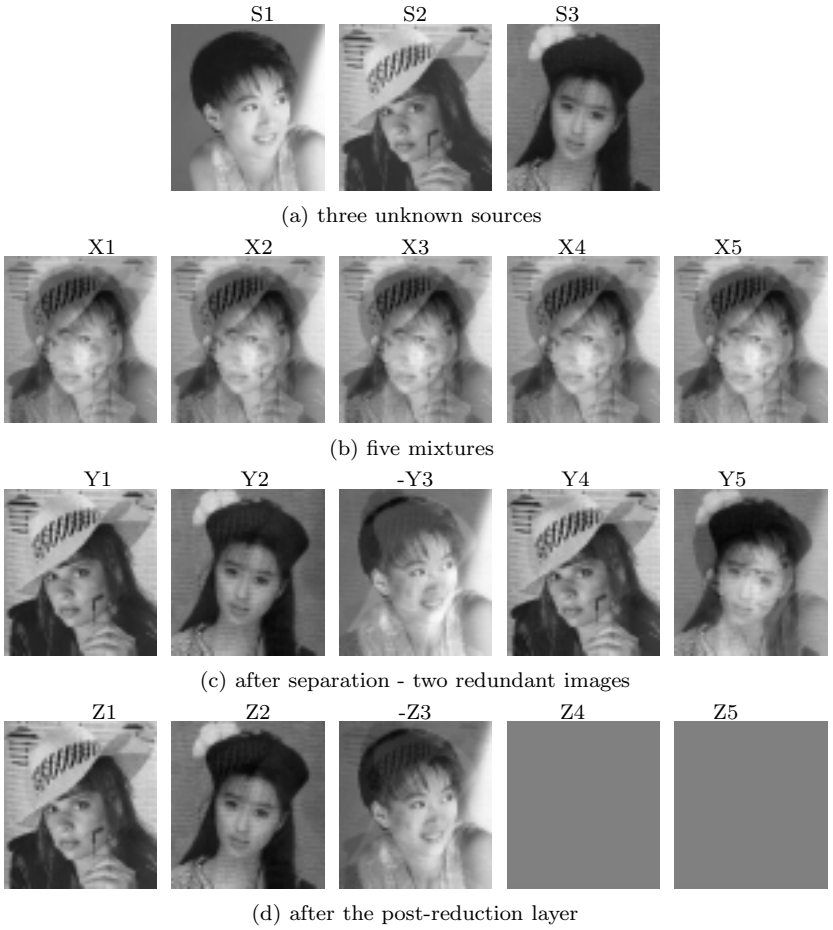
Table 3.2 Correlation values (×100) between pairs of sources, i.e. $E\{s_i s_j\}$, and between pairs of signals after mixing, pre–whitening and separation.

The nonlinear PCA subspace rule (3.28) is an exception among learning algorithms employing pre–whitening, because it can provide a separating matrix $\widehat{\boldsymbol{W}}(t)$ that is not orthogonal. But also the output images found by the algorithms (3.32) and (3.40) are in fact more independent than the original correlated sources, in any case they are less correlated than the original sources (see *Table* 3.2). For the global algorithm (3.32), an obvious explanation is that it can be derived by minimizing the Kullback-Leibler divergence - the minimum is achieved for independent outputs.

The results are shown in *Figure* 3.5. There are 7 source images shown in the first row: S1–S4 are face images and S5–S7 are textures. All the sources were sub-Gaussian except S7 which had a small positive kurtosis value. The images labeled X1–X9 show 9 mixtures formed of these sources, and V1–V7 are the 7 pre–whitened images. The separated sources Y1–Y7 are shown in the bottom row – they are very close to the original. In this experiment, the nonlinear PCA subspace rule was used for the orthogonal separating matrix $\widehat{\boldsymbol{W}}$. In this noiseless case, the correct number of sources is obtained directly as a by–product of the PCA–based whitening, and it is equal to the number of nonzero eigenvalues of the data covariance matrix. Hence, the basic structure (*Figure* 3.2) where the data compression takes place already in the pre–whitening layer, yields better results if the number of mixtures $n > m$, and $l = m$ sources are separated. However, in practical situations it may happen that we estimate m wrongly and the second approach should be used instead.

Source separation with redundancy reduction

Now two experiments dealing with over-determined separation are presented, where the second class of architectures is applied which do not use pre–whitening. In these networks, either the global single–layer rule (3.32) or the local multi–layer rule (3.40) are used. Moreover, in the noise–free case an additional post–processing layer, using the learning rule (3.44), is applied for redundancy reduction.



(a) three unknown sources

(b) five mixtures

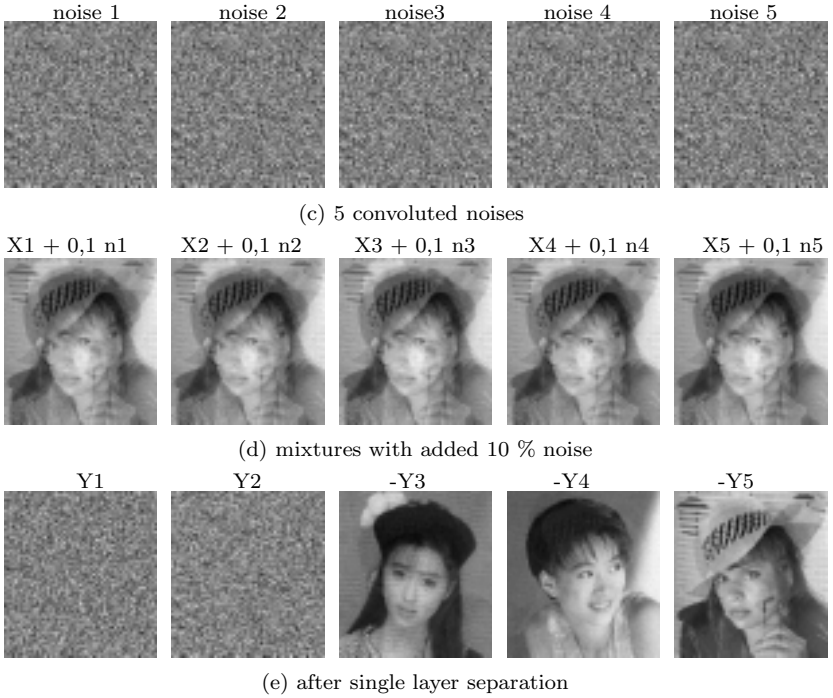
(c) after separation - two redundant images

(d) after the post-reduction layer

Figure 3.6 Example of blind separation with subsequent redundancy reduction (no noise) (published by the author in [CIK99] - section 4). Three unknown sources (a) are mixed to five mixture signals (b). After a single layer with global rule (c) two redundant images always appear. After applying the post–processing layer both these signals are suppressed (d).

Three original images were mixed by a well conditional matrix, which is assumed to be completely unknown to the separation network. This matrix $\boldsymbol{A}_{5\times 3}$ has a condition number of 43.1. In the first experiment no noise was assumed, whereas in the second case additional additive noise was added to every sensor image (mixture). The results of processing the first set of mixtures (noise–free) are given in *Figure* 3.6, whereas the results for the noisy case are provided in *Figure* 3.7.

Usually the local learning rule used in the multi–layer network structure estimates the sources in a sequential order - the first source after one processing cycle, the second source after the second processing cycle etc. The global learning rule determines all the sources simultaneously using a single–layer network. If there are more outputs and mixtures than sources ($n > m$), the separation quality is usually slightly worse than in the



(c) 5 convoluted noises

(d) mixtures with added 10 % noise

(e) after single layer separation

Figure 3.7 Example of blind separation with more sensors than sources under an additive large noise (published by the author in [CIK99] - section 4). Three sources are mixed to 5 sensor images (see *Figure* 3.6(a),(b)) and 5 convoluted noise images (c) are added to the mixed images - 10% of one noise signal to one sensor image (d). After a single layer with global rule (e) two noise images appear, but the three others correspond clearly to the 3 source images. In this case the redundancy reduction layer need not to be applied.

case where the number of mixtures is correct ($n = m$). The final redundancy elimination layer suppresses redundant signals and does not switch between channel signals.

In the noisy case even no redundancy among the sources occurs. On the "free" output channels the noise signals appear instead (compare the bottom row in *Figure* 3.7). The separated images are already of high quality. Hence, the redundancy elimination layer may even lead to a slight decrease in the quality of separation. Then it is better to choose as outputs of the network those signals $y_i(t)$ from the separation layer which correspond to non–suppressed channels in the reduction layer.

3.3 BSS IN CONVOLUTED NOISE

The basic approaches to blind source separation (BSS) assume that sensor signals are noiseless or noise is considered as one primary source [AMA96, BEL95, CAR96, JUT91]. Obviously, more than one source could be noise, but mostly one of them might be Gaussian noise if it is necessary to extract all source signals including noise.

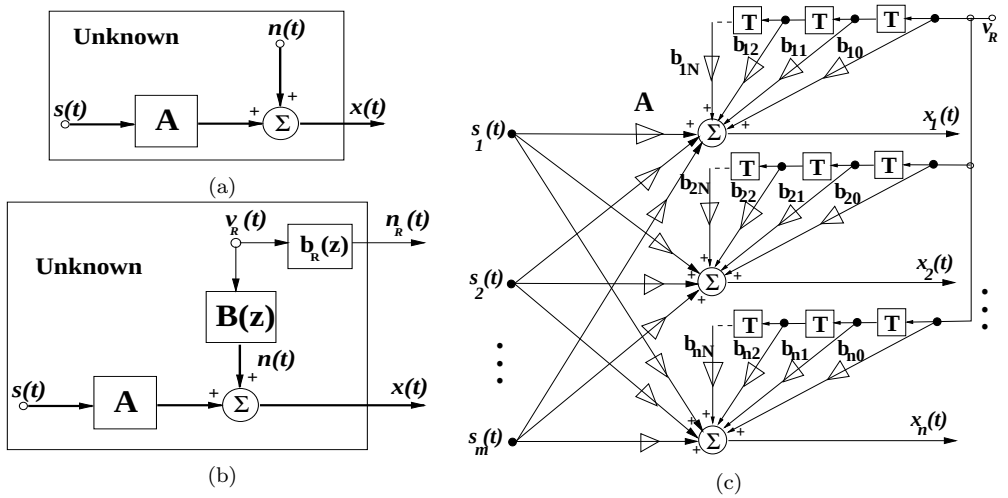


Figure 3.8 The model of source mixing with additive noise: (a) general model, (b) a mixing model with convoluted noise, (c) a more detailed model of the additive convoluted noise.

A more realistic and practical model of BSS considers that different unknown noise signals, possibly representing colored noise, are added to each sensor signal [GER96]. Such a situation appears in almost all real–life (real–world) problems. Now the following problem arises: how to efficiently separate signals if additive noise can no longer be neglected? Alternatively the problem can be stated as, how to cancel or suppress the additive noise.

In this section we describe an adaptive approach to simultaneous source separation and cancellation of additive, convoluted noise from many–source mixtures if some corresponding reference noise can be measured independently of the sensor signals. This approach was proposed by the author in [KAS97a].

3.3.1 The extended BSS problem

The mixing model

Let us consider the complete BSS problem, in which the mixing model includes additive noise (*Figure* 3.8 (a)):

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{n}(t), \tag{3.50}$$

where $\boldsymbol{s}(t) = [s_1(t), ..., s_m(t)]^T$ is a vector of *unknown*, independent primary sources, $\boldsymbol{A}$ is an $m \times n$ unknown mixing matrix, $\boldsymbol{x}(t) = [x_1(t), ..., x_n(t)]^T$ is the *observed (measured)* vector of sensor signals and $\boldsymbol{n}(t) = [n_1(t), n_2(t)..., n_n(t)]^T$ is an additive noise vector. We assume that the noise signals $n_i(t)$ are de-correlated to source signals $s_i(t)$, $\forall i$.

In general, the problem is rather difficult because we have $n + m$ unknown signals (where m is the number of sources and n is the number of sensors). Hence the problem is highly under–determined - without any a priori information about the mixture model and noise it is very difficult or even impossible to solve [GER96].

The noise model

However, in many practical situations one can measure at least some (transformed) image of the (unknown) *environmental* noise $\nu_R(t)$, assuming that no source signals are present during the measurement process. The measured noise will be denoted further as *reference* noise $n_R(t)$.

The unknown environmental noise $\nu_R(t)$ influences each sensor, but it could be added with different strengths. Moreover, noise could reach each sensor with some delay due to finite time propagation of signals. For this reason the n-dimensional additive noise vector $\boldsymbol{n}(t)$ is of convoluted signal type, which can be modeled by the *Finite Impulse Response* (*FIR*) model [GER96, WID85] (*Figure* 3.8 (b)), i.e.

$$n_i(t) = \sum_{j=0}^{N} b_{ij}(z)\nu_{Rj}(t) = \sum_{j=1}^{p}\sum_{k=0}^{N} b_{ijk}\nu_{Rj}(t-kT), \tag{3.51}$$

where z^{-1} is the unit delay and $\{\nu_{Rj}\}$-s are individual environmental noises. In this model the p environmental noises $\boldsymbol{\nu}_R = [\nu_{R1}, \nu_{R2}, ..., \nu_{Rp}]^T$ are added to each sensor (mixture of sources) with different unit delays T and various (but unknown) coefficients b_{ijk}. Up to N delays are considered. In other words, we assume that additive noise constitutes a convolution and superposition of some environmental noises $\{\boldsymbol{\nu}_{Rj}\}$ (*Figure* 3.8 (c)):

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{B}(z)\boldsymbol{\nu}_R(t), \tag{3.52}$$

where $\boldsymbol{B}(z) = [b_{ij}(z)]_{n\times P}$, with $b_{ij}(z) = b_{ij0} + b_{ij1}z^{-1} + ... + b_{ijN}z^{-N}$.

For simplicity of further consideration let us assume that only one single environmental noise ν_R ($p = 1$) is available. However, one could easily extend our approach for the case of an arbitrary number of noises.

Measured reference noise

As already mentioned, in order to propose a realistic solution for source separation in a noisy environment, one should be able to measure the environmental noise ν_R directly or at least its convoluted form n_R. Let us assume that a convoluted reference noise $n_R(t)$ can be measured independently from the sensors measuring the noisy source mixtures (*Figure* 3.8 (b)). Noise $n_R(t)$ will also be modeled as the convolution of the unknown environment noise $\nu_R(t)$:

$$n_R(t) = \sum_{j=0}^{N_R} b_{Rj}\nu_R(t-jT) = \boldsymbol{b}_R(z)\nu_R(t). \tag{3.53}$$

Thus a general mixing model contains the following unknown elements: the matrices $\boldsymbol{A}$ and $\boldsymbol{B}(z)$, and the vector $\boldsymbol{b}_R(z)$. It is assumed that the number of sources m and the number of time delay units N (i.e. maximum order of FIR filters) are unknown (however, the number of sensors must be larger or equal to the number of sources).

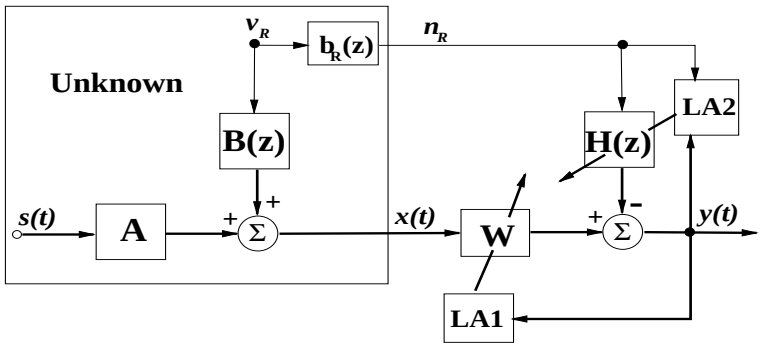


Figure 3.9 The basic de–mixing model - applying two learning algorithms $LA1, LA2$ simultaneously.

3.3.2 Two adaptive solutions

Two alternative solutions are described: in the first one a simultaneous separation of signals and subtraction of additive noise is performed (using an adaptive FIR filter in each channel), whereas in the second model a reduction or cancellation of noise is performed first and after it the blind separation of sources takes place.

Basic model

In the basic approach two learning steps are simultaneously performed: the signals are separated from their linear mixture and the additive noise is estimated and subtracted (*Figure* 3.9). Thus the output signals are derived as:

$$\begin{aligned} \boldsymbol{y}(t) &= \boldsymbol{W}\boldsymbol{x}(t) - \boldsymbol{H}(z)n_R = \\ &= \boldsymbol{W}\boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{W}\boldsymbol{B}(z)\nu_R - \boldsymbol{H}(z)\boldsymbol{b}_R(z)\nu_R, \end{aligned} \tag{3.54}$$

where $\boldsymbol{H}(z) = [h_1(z), ..., h_n(z)]^T$, with $h_i(z) = h_{i0} + h_{i1}z^{-1} + h_{i2}z^{-2} + ... + h_{iM}z^{-M}$.

In such case the de–mixing model can be described by a set of equations:

$$y_i(t) = \tilde{y}_i(t) - n_i(t) = \tilde{y}_i(t) - \sum_{j=0}^{M} h_{ij} n_R(t - jT), \tag{3.55}$$

where $\tilde{y}_i(t) = \sum_{j=1}^{n} w_{ij}(t)x_j(t), (i = 1, 2, ..., m)$.

For simplicity of consideration let us assume that signals from $\boldsymbol{y}(t)$ are properly scaled and ordered in accordance with $\boldsymbol{s}(t)$. Then, $\boldsymbol{y}(t) \simeq \boldsymbol{s}(t)$ if

$$\boldsymbol{H}(z)\boldsymbol{b}_R(z) = \boldsymbol{W}\boldsymbol{B}(z), \quad \text{i.e.} \quad \sum_{k=1}^{N} h_i(z)b_R(z) = \sum_{j=1}^{m} w_{ij}b_j(z), \forall i, \tag{3.56}$$

$$\text{and} \quad \boldsymbol{W}\boldsymbol{A} = \boldsymbol{I}. \tag{3.57}$$

The number of time delay units M in the de–noising model should be at least equal to the sum of corresponding numbers N, N_R in the mixing model (i.e. $M \geq (N + N_R)$), but in practice, especially if $N_R > 1$ it should be much larger.

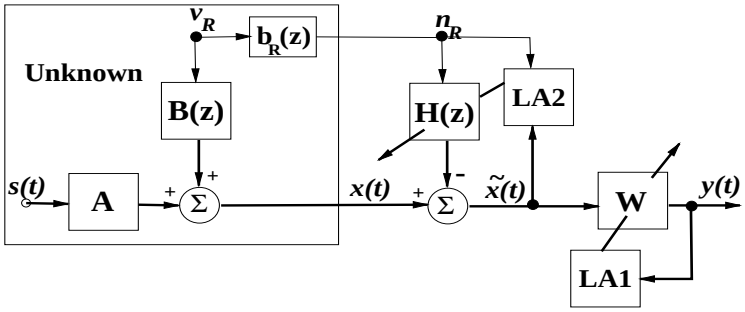


Figure 3.10 Alternative (simplified) model for blind separation with noise cancellation as pre–processing.

Alternative simplified model

In the simplified model the two learning steps are performed in sequence. A pre–processing takes place in order to cancel the noise, contained in the mixture, and after that the real separation of the noise-free signals is performed (*Figure* 3.10). This is the most simple way to deal with noise cancellation. Its drawback is that in real life problems one hardly expects to make a perfect noise cancellation. The output signals are now derived from:

$$\boldsymbol{y}(t) \quad = \quad \boldsymbol{W}[\boldsymbol{x}(t) - \boldsymbol{H}(z)n_R] = \boldsymbol{W}\boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{W}[\boldsymbol{B}(z) - \boldsymbol{H}(z)\boldsymbol{b}_R(z)]\nu_R, \quad (3.58)$$

and it is obvious, that $\boldsymbol{y}(t) \simeq \boldsymbol{s}(t)$ if (again, problems of signal scaling and permutation are ignored):

$$\boldsymbol{W}\boldsymbol{A} = \boldsymbol{I} \quad \text{and} \quad \boldsymbol{H}(z)\boldsymbol{b}_R(z) = \boldsymbol{B}(z). \quad (3.59)$$

Learning rules for source separation

In the separation layer one can apply the global learning rule (3.32), which was derived in the natural gradient approach [AMA96]:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \eta(t)\frac{\partial D}{\partial \boldsymbol{W}}\boldsymbol{W}^T\boldsymbol{W} \quad (3.60)$$

and which has the robust form of:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \eta(t)\left[\boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)]\ \boldsymbol{y}^T(t)\right]\boldsymbol{W}(t), \quad (3.61)$$

where $\eta(t)$ is the learning rate and $\boldsymbol{f}[\boldsymbol{y}(t)]$ is a vector of nonlinear activation functions.

Alternatively, using a modified form of stochastic gradient descent technique

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) - \eta(t)\frac{\partial D}{\partial \boldsymbol{W}}\boldsymbol{W}^T \quad (3.62)$$

one can obtain the simple local learning rule:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \eta(t)\left\{\boldsymbol{I} - \boldsymbol{f}[\boldsymbol{y}(t)]\boldsymbol{y}^T(t)\right\}. \quad (3.63)$$

Adaptive delta rule for noise cancellation

An adaptive learning algorithm for the on–line update of noise-cancellation coefficients $\boldsymbol{H}(t) = [h_{ij}(t)]$ can be derived by applying the concept of minimization of generalized output energy $\tilde{\boldsymbol{x}}(t) = [\tilde{x}_1(t), \tilde{x}_2(t), \ldots, \tilde{x}_n(t)]^T$. The following cost function (generalized energy) can be formulated:

$$J(\boldsymbol{H}) = \sum_{i=1}^{n} \rho_i(\tilde{x}_i), \tag{3.64}$$

where $\rho_i(\tilde{x}_i)$ is a suitably chosen loss function, typically [CIC94]:

$$\rho_i(\tilde{x}_i) = \frac{1}{\beta} \ln cosh(\beta \tilde{x}_i) \quad \text{or} \quad \rho_i(\tilde{x}_i) = \frac{1}{p}|\tilde{x}_i)|^p \tag{3.65}$$

$$\text{and} \qquad \tilde{x}_i(t) = x_i(t) - \sum_{j=1}^{n} h_{ij} n_R(t - jT), \quad \forall i. \tag{3.66}$$

Minimization of this cost function according to stochastic gradient descent leads to the following learning algorithm:

$$h_{ij}(t+1) \quad = \quad h_{ij}(t) - \tilde{\eta}(t) \frac{\partial J(\boldsymbol{H})}{\partial h_{ij}} = h_{ij}(t) + \tilde{\eta}(t) f_R[\tilde{x}_i(t)] n_R(t - jT). \tag{3.67}$$

$f_R(\tilde{x}_i(t))$ is a suitably chosen nonlinear activation (error) function defined as :

$$f_R(\tilde{x}_i(t)) = \frac{\partial \rho_i(\tilde{x}_i)}{\partial \tilde{x}_i}. \tag{3.68}$$

Typically: $f_R(\tilde{x}_i(t)) = \tanh(\beta \tilde{x}_i)$ or $f_R(\tilde{x}_i(t)) = \tilde{x}_i^3$. The optimal choice of activation function depends on the distribution of noise and source signals. For Gaussian noise it is optimal to set $f_R(\tilde{x}_i) = \tilde{x}_i(t)$, for sub–Gaussian signals one can use $f_R(\tilde{x}_i) = tanh(\beta \tilde{x}_i(t))$ and for super–Gaussian: $f_R(\tilde{x}_i) = |\tilde{x}_i|^P sign(\beta \tilde{x}_i(t)), p = 2, 3, 4$.

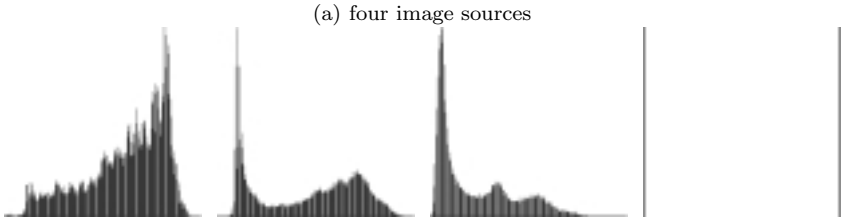
3.3.3 Testing the approach

The author has tested the proposed approach for various de-mixing models and for both image and sound sources. By computer simulation experiments it was verified, that the approach is applicable for a large amount of noise [KAS97a]. Convoluted noises were modeled using 10–th order FIR filters.

Direct measurement of reference noise

In the first experiment four (unknown) natural images were mixed by an ill–conditioned mixing matrix $\boldsymbol{A}_1$ of size $m = 4, n = 4$ (*Figure* 3.11) with condition number $cond(\boldsymbol{A}_1) = 106.68$. At first we assumed the environment noise to be given to us directly, as the reference noise, i.e. $n_R = \nu_R$ and $\boldsymbol{b}_R = \boldsymbol{1}$ (*Figure* 3.12). Convoluted noises

(a) four image sources



(b) image histograms

Figure 3.11 Four source images and their histograms.

were modeled using four 10–th order FIR filters (N=10), where all of them introduce large additive noise. Their coefficients in vector form were randomly chosen as:

$$\begin{aligned}
\boldsymbol{b}_1 &= [1.2, 1.4, 1.1, 1.0, 0.9, 0.8, 0.7, 0.55, 0.40, 0.3] \\
\boldsymbol{b}_2 &= [1.3, 1.2, 1.5, 1.1, 0.95, 0.84, 0.77, 0.65, 0.54, 0.4] \\
\boldsymbol{b}_3 &= [1.1, 1.2, 1.3, 1.15, 0.99, 0.74, 0.87, 0.85, 0.94, 1.0] \\
\boldsymbol{b}_4 &= [1.2, 1.0, 0.8, 0.55, 0.28, 0.14, 0.37, 0.48, 0.64, 0.9]
\end{aligned}$$

In the de-mixing model the number M of delay units was chosen to be equal to 25. Both basic and simplified de-mixing models (*Figure* 3.9, 3.10) gave very good results of noise cancellation and source separation. Already after one epoch of signal data the weights in $\boldsymbol{h}$ achieved equilibrium point, e.g. final estimation of the deconvolution matrix $\boldsymbol{H}(z)$ for the basic model was :

$$\begin{aligned}
\boldsymbol{h}_1 = [&1.2022, 1.4011, 1.0967, 0.9982, 0.8962, 0.7949, 0.6999, 0.5470, 0.3998, 0.2989, \\
&-0.0118, -0.0052, -0.0037, -0.0028, 0.0017, -0.0014, -0.0009, 0.0000, \\
&-0.0125, -0.0070, -0.0084, -0.0067, -0.0019, -0.0016, -0.0022] \\
\boldsymbol{h}_2 = [&1.3016, 1.2035, 1.4990, 1.1013, 0.9489, 0.8345, 0.7707, 0.6486, 0.5409, 0.4009, \\
&-0.0112, -0.0036, -0.0020, -0.0033, 0.0021, -0.0006, -0.0010, 0.0016, \\
&-0.0122, -0.0051, -0.0068, -0.0069, -0.0016, -0.0019, -0.0034] \\
\boldsymbol{h}_3 = [&1.1017, 1.2028, 1.2985, 1.1510, 0.9889, 0.7351, 0.8707, 0.8487, 0.9413, 1.0011, \\
&-0.0104, -0.0035, -0.0022, -0.0028, 0.0024, -0.0000, -0.0005, 0.0014, \\
&-0.0112, -0.0048, -0.0059, -0.0061, -0.0013, -0.0018, -0.0032] \\
\boldsymbol{h}_4 = [&1.2018, 1.0025, 0.7986, 0.5514, 0.2799, 0.1356, 0.3705, 0.4789, 0.6419, 0.9018, \\
&-0.0096, -0.0032, -0.0020, -0.0024, 0.0027, 0.0003, -0.0004, 0.0013, \\
&-0.0103, -0.0043, -0.0049, -0.0058, -0.0014, -0.0021, -0.0035]
\end{aligned}$$

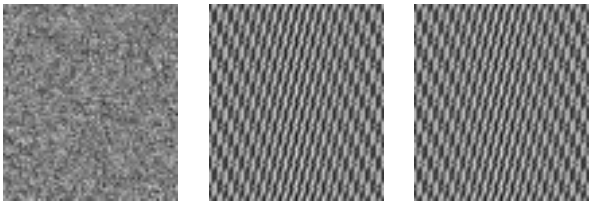


Figure 3.12 Three noise images used in our computer experiments for generation of additive convoluted noises.

Image sources	Image sources			Noise images $v_R(t)$		
	Face 2	Face 3	Stripes	Noise 1	Noise 2	Noise 3
Face 1	22.60	0.275	0.709	0.316	0.165	0.195
Face 2	-	14.05	0.226	0.304	0.040	0.914
Face 3	-	-	0.857	0.323	0.111	0.158
Stripes 3	-	-	-	0.149	0.0002	0.025

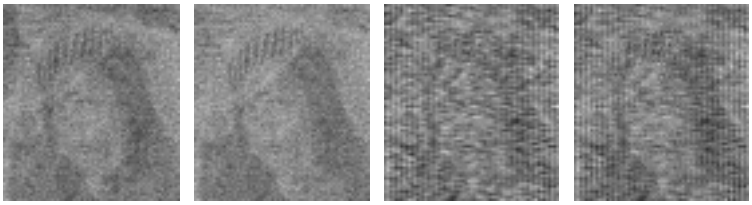
Table 3.3 Source image and noise image correlations (in %), i.e. $E\{s_i s_j\} \times 100$.

From the above data it is evident that the estimation error for each matrix element is below 1% (a similar quality was achieved for the simplified model). The exemplary results for this simple reference noise case are illustrated in *Figure* 3.13.

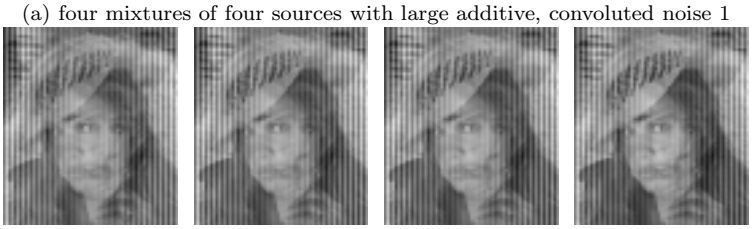
Convoluted reference noise

In the next experiments the general reference noise measurement model is assumed in which $\boldsymbol{b}_R$ is some non-zero vector, and $n_R(t) = \boldsymbol{b}_R(z)\nu_R(t)$. Let a 10–order FIR filter be given: $\boldsymbol{b}_R = [1.5, 1.3, 1.2, 0.92, 0.75, 0.70, 0.75, 0.86, 0.94, 1.0]$. It is assumed, that these coefficients are completely unknown. In the de-mixing model the number M of delay units was selected to be equal to 100. If compared to the previous case of simplified reference noise measurement, the learning process is slower and the noise cannot be fully canceled. Although in the pre-processing model of noise cancellation we were able to suppress the additive noise in the mixture of source signals, in the subsequent source separation stage this noise is again amplified to some visible form in one of the outputs. Other $m-1$ outputs correspond to $(m-1)$ estimated sources. However, the quality of such separated $(m-1)$ sources strongly depends on the condition number of the mixing matrix. In the case of the matrix $\boldsymbol{A}_1$ (of size 4×4) only 3 sources were extracted with rather poor quality. For another mixing matrix $\boldsymbol{A}_2$ (of size 4×4), due to its low condition number ($cond(\boldsymbol{A}_2) = 38.78$) a successful separation of 3 sources (from a total of 4) with high quality was possible. Illustrative results are given in *Figure* 3.14 (a),(b).

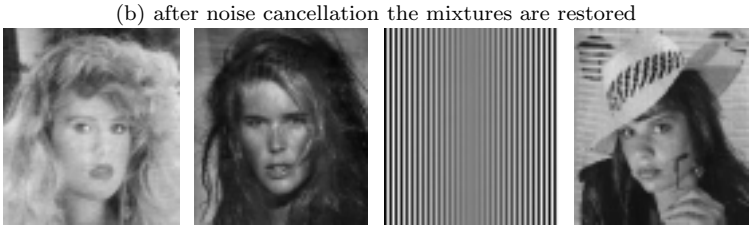
In order to extract successfully all sources under additive noise, at least one auxiliary sensor may be useful. The de-mixing process provided good results, even if the environment noise was not directly available, while applied to mixtures obtained from another mixing matrices $\boldsymbol{A}_3$ (of size 5×4) and $\boldsymbol{A}_4$ (of size 5×3), with additive noise in each sensor: $cond(\boldsymbol{A}_3) = 13.475$, $cond(\boldsymbol{A}_4) = 7.575$. For a visual illustration of the obtained results see *Figure* 3.14 (c),(d).



(a) four mixtures of four sources with large additive, convoluted noise 1



(b) after noise cancellation the mixtures are restored



(c) after blind separation and noise cancellation

Figure 3.13 Example of blind source separation and noise cancellation, if the original environmental noise is directly available.

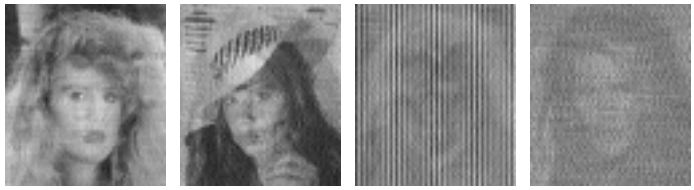
It should be emphasized that performance depends on learning rates $\eta(t), \tilde{\eta}(t)$ – on a proper setting of their initial values and decay factors – and on chosen nonlinear activation functions f, f_R . The provided result samples show that for a given set of parameter values both solution models performed well and gave nearly similar results.

3.4 BSS WITH FEWER SENSORS THAN SOURCES

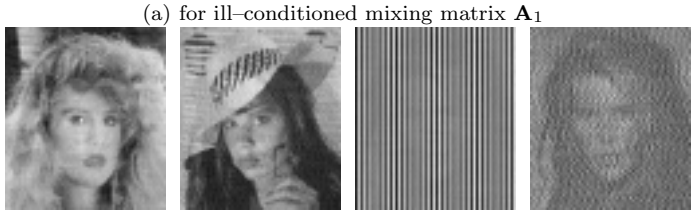
A related problem to BSS is the blind source extraction (BSE) problem, where the number of sensor signals m is lower than the number of mixed sources n, i.e. $m < n$. This topic is still under investigation. At this stage, a partly non-neural solution to this problem for some restricted source classes is proposed (for sparse binary and constrained grey-scale natural images), at least.

3.4.1 The BSE problem

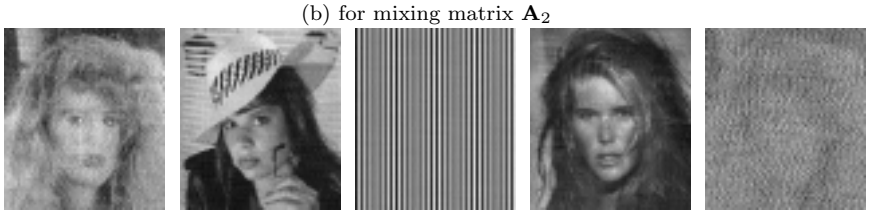
The BSE problem has only recently gained larger attention. Pajunen [PAJ96] has proposed an algorithm for binary source separation, that separates m binary sources



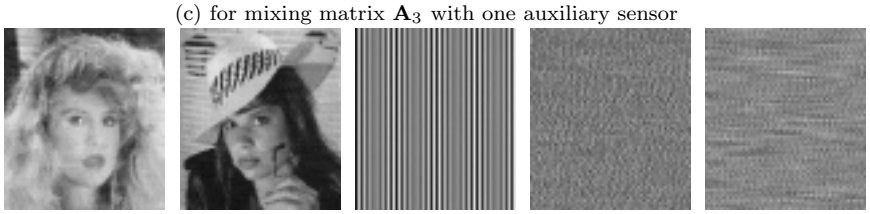
(a) for ill–conditioned mixing matrix $\mathbf{A}_1$



(b) for mixing matrix $\mathbf{A}_2$



(c) for mixing matrix $\mathbf{A}_3$ with one auxiliary sensor



(d) for mixing matrix $\mathbf{A}_4$ with two auxiliary sensors.

Figure 3.14 Results of image separation with noise cancellation if convoluted reference noise $\mathbf{n}_R$ is available only.

from two or more mixtures. The restrictive assumptions about sources are that the mixture vectors must not overlap, and the mixing matrix must have non–parallel columns. Simulation results have been presented for the noise–free case.

Chen & Donoho [CHD98] have applied a so called *Basis Pursuit* approach for spectrum estimation. This work is related to our research in the sense that *Basis Pursuit* decomposes a signal into an optimal superposition of dictionary elements, where optimal means having the smallest l^1 norm of coefficients among all such decompositions. Their dictionary includes over-complete cosine and sine bases, and the Dirac basis. Hence this optimization principle leads to decompositions that can be very sparse.

This topic has recently been studied theoretically in [CAO96]. The authors show that it is possible to separate the m sources into n disjoint groups if, and only if, $\boldsymbol{A}$ has n linearly independent column vectors, and the remaining $m-n$ column vectors satisfy the special condition that each of them is parallel to one of these n column vectors.

3.4.2 Proposed solution

The author has studied the BSE problem only recently, hence a partly non-neural solution can be given at this stage. Assuming that the sensor signals are available only, the BSE-approach consists of two main steps and one optional final step:

1. estimation $\widehat{\boldsymbol{A}}$ of the (unknown) mixing matrix $\boldsymbol{A}$ from mixed (sensor) signal vector $\boldsymbol{x}(t)$,
2. extraction of source signals $\widehat{s}(t)$, for given $\widehat{\boldsymbol{A}}$ and $\boldsymbol{x}(t)$,
3. a final post-processing step for specific signals (option).

Mixing matrix estimation

For this key problem, still no robust solution can be given. At least for some signals, the author has tested a two-layer neural network to give satisfactory results. The first layer performs a pre-whitening of the sensor signals $\boldsymbol{x}(t)$: $\boldsymbol{v}(t) = \boldsymbol{V}\boldsymbol{x}(t)$, where $\boldsymbol{V} = \{V_{ji}\}$ is the matrix of weights of the pre–whitening feed-forward layer. It provides on its output an orthogonal set of mixed signals - the vector signal $\boldsymbol{v}(t) = [v_1(t), ..., v_m(t)]^T$ satisfying the condition:

$$E\{\boldsymbol{v}\boldsymbol{v}_1^T\} = \boldsymbol{I}_m. \tag{3.69}$$

In the learning process of this whitening layer a standard neural whitening rule is applied:

$$\begin{array}{ll} \delta V_{ji}(k) & = -\eta(k)\boldsymbol{v}_j(k)\boldsymbol{v}_i^T(k), \qquad \forall i \neq j, \\ \delta V_{ji}(k) & = \eta(k)(1 - \boldsymbol{v}_j(k)\boldsymbol{v}_i^T(k)), \quad \forall i = j, \end{array} \tag{3.70}$$

where the initial matrix $\boldsymbol{V}(0)$ is a random nonzero matrix.

Let $\boldsymbol{A}$ be the unknown mixing matrix. Including the whitening step into the mixing process the combined mixing matrix is given as: $\boldsymbol{A}_1 = \boldsymbol{V}\boldsymbol{A}$.

The task of the second feed-forward layer: $\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{v}(t)$ is limited to the learning process, i.e. a natural gradient learning algorithm for a blind estimation of the mixing matrix $\boldsymbol{W} = \widehat{\boldsymbol{A}}_1$ is applied. $\widehat{\boldsymbol{A}}_1$ is an estimate of the combined mixing matrix $\boldsymbol{A}_1$. The learning algorithm takes the following form:

$$\begin{array}{ll} \Delta\widehat{\boldsymbol{A}}_1(k) & = -\eta(k)\widehat{\boldsymbol{A}}_1(k)\left[\psi[\boldsymbol{y}(k)]\boldsymbol{y}^T(k)\widehat{\boldsymbol{A}}_1(k)\widehat{\boldsymbol{A}}_1^T(k) - \boldsymbol{y}(k)\psi[\boldsymbol{y}(^T(k)]\right] \\ & = -\eta(k)\left[\widehat{\boldsymbol{A}}_1(k)\psi[\boldsymbol{y}(k)]\boldsymbol{y}^T(k) - \boldsymbol{v}(k)\psi[\boldsymbol{y}(^T(k)]\right], \end{array} \tag{3.71}$$

where the initial matrix $\widehat{\boldsymbol{A}}_1(0)$ satisfies the condition $\widehat{\boldsymbol{A}}_1(0)\widehat{\boldsymbol{A}}_1^T(0) = \boldsymbol{I}_m$. From the orthogonality requirement for matrix $\boldsymbol{A}_1$ it follows also that: $\boldsymbol{y} = \widehat{\boldsymbol{A}}_1^T\boldsymbol{v}$.

Extraction of sources

After the estimate $\widehat{\boldsymbol{A}}$ (or $\widehat{\boldsymbol{A}}_1$) of the mixing matrix $\boldsymbol{A}$ (or combined mixing matrix $\boldsymbol{A}_1$, respectively) is known, there exist potentially many solutions to the under-determined source extraction problem. We can solve it in special cases, at least. When source signals are spiky and sparse signals, in the sense that they fluctuate mostly around zero

and only occasionally have nonzero values, the problem of estimation of unknown signals can be converted to the extended linear programming problem, i.e. finding the optimal sequence of estimated source signals $\widehat{s}_i(k)(i = 1, ..., n,)$, which minimize the l_1 norm:

$$\sum_k \sum_{i=0}^{n} |\widehat{s}_i(t)|, \tag{3.72}$$

subject to the constraints:

$$\widehat{\boldsymbol{A}}\widehat{s}(t) = \boldsymbol{x}(t), \quad \text{or} \quad \widehat{\boldsymbol{A}}_1\widehat{s}(t) = \boldsymbol{v}(t), \ \forall k. \tag{3.73}$$

A very efficient linear programming algorithm that allows one to minimize the l^1 norm is known, called the FOCUSS algorithm [GOR97]. The author has applied this FOCUSS algorithm with the following iteration rule:

$$\widehat{s}(k+1) = \boldsymbol{D}(k+1)inv[\widehat{\boldsymbol{A}}\boldsymbol{D}_(k+1)]\boldsymbol{x}(k+1), \tag{3.74}$$

where the diagonal matrix $\boldsymbol{D}$ is obtained by:

$$\boldsymbol{D}(k+1) = diag[|\widehat{s}(k)|^{1-p/2}], \tag{3.75}$$

and $inv[.]$ means the pseudo-inverse operation:

$$inv[\boldsymbol{W}] = \boldsymbol{W}^T(\boldsymbol{W}\boldsymbol{W}^T)^{-1}. \tag{3.76}$$

The initial diagonal elements are to $diag[\boldsymbol{D}(0)] = [1, ..., 1]^T$ and the parameter $p = 0.5$. During the above iteration process a competition between the columns of $\widehat{\boldsymbol{A}}$ appears, which of them should represent the vector $\boldsymbol{x}$. At the end some of the columns survive only to represent $\boldsymbol{v}$.

Postprocessing for (n-1) sensors to (n) sources

After all the previous steps a proper estimation of the sources is usually done for all such vector samples, where at least one source sample is equal to zero. If all sources have non-zero values, then the estimated sample vector is estimated wrongly. At least in some specific signal cases these could be corrected. In this subsection a postprocessing step is proposed, that is generally valid if (n-1) sensors are available for (n) sources. On this assumption the crossing section of (n-1) hyperplanes in the n-hyperspace determines a line in the n-space. A wrong signal vector corresponds to a point on this line, whereas the solution point is located somewhere else on this line. Thus the proper solution can be obtained by a linear shift of the estimated point. The direction cosine of the solution line is dependent on the known (estimated) mixing matrix and it is independent of the sources. Thus, if we could only find the proper correction value (at a given time sample) for one estimated source, we could properly correct all the remaining outputs.

Usually there is no need for any post-correction if the sources are spiky signals, i.e. with high probability in each time sample at least one of the sources is equal to zero.

The author has tested the proposed correction mechanism for several types of source signals. The sources may be binary signals, three-valued positive signals or they may be of general waveform but subject to a so called *ST-constraint*, i.e. they fluctuate rather slowly and smoothly in comparison with the sampling frequency - only one source is allowed to have a rapid amplitude skip between two consecutive samples.

Postprocessing if more than one sensor is lost

In this case the estimated results may be wrong, if at least (n-k) sources are not equal to zero at each sample time. Otherwise, this could be a realistic assumption if all the sources are spiky signals.

3.4.3 Test results

During our tests we observed that all, except one (the blind matrix estimation rules) performed very well, and the sources were properly extracted. Further work is needed on a proper modification of the adaptive method for mixing matrix estimation.

In the first experiment the reconstruction of binary edge images is shown (*Figure* 3.15). The second experiment on natural image reconstruction requires the running of a post-processing step. The sources are face images satisfying the ST-restriction (*Figure* 3.16).

3.5 CONCLUSIONS

In this chapter the restoration of images from their multi-image mixtures was experimentally studied. At first two promising neural network approaches were presented for the problem of blind separation of an unknown number of sources, where only the maximum possible number of active sources is known in advance. In such a case the number of sensors is usually larger than the number of source signals. We have presented both qualitative and quantitative results for the two dominant classes of such networks which differ with respect to the need of pre–whitening. The main advantages of the proposed methods are their simplicity, adaptivity (the algorithms can be used on–line), and in some cases locality and/or robustness for badly scaled and ill–conditioned mixing matrices.

Some new issues arose from the results of our experiments. In the basic ICA/BSS model it is assumed that the source signals are mutually independent. For example for the face images this does not generally hold even as an approximation, because they are usually clearly correlated. Due to this fact clear differences in the behavior of separation methods can be observed. In the methods applying pre–whitening, the orthogonality constraint set on the separating matrix forces the output signals to be mutually uncorrelated (except for the nonlinear PCA rule). The second class of learning algorithms tries to perform whitening and separation simultaneously in one or more layers. Thus they respond to higher–order statistics of the source signals at the same time as the

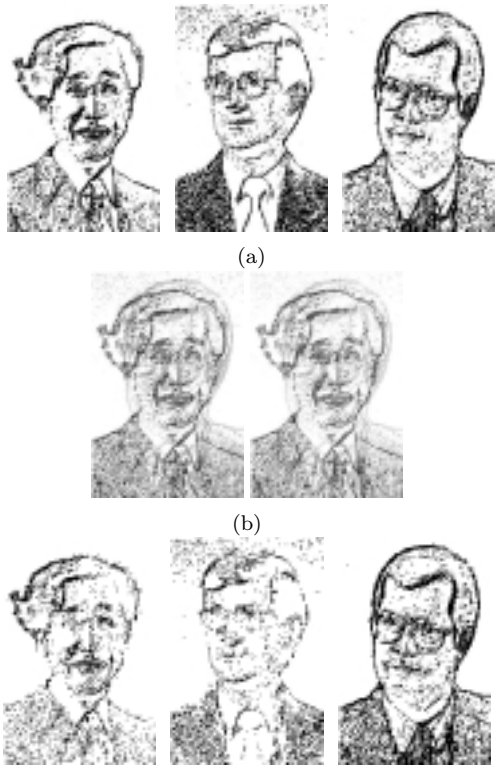


Figure 3.15 Example of edge image reconstruction: (a) the three binary edge images, (b) their two mixtures, (c) the three extracted edge images.

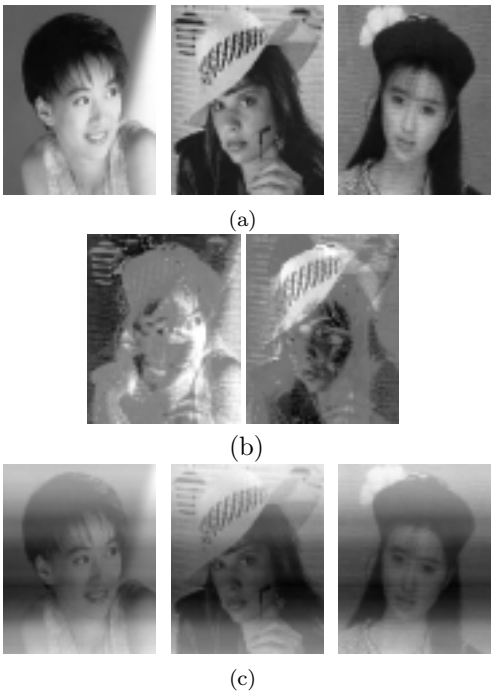


Figure 3.16 Example of face image reconstruction, subject to ST-restriction: (a) the three source images, (b) their two mixtures, (c) the three extracted output signals.

sources are de-correlated. As a result, the outputs of these networks are not necessarily uncorrelated for correlated source signals.

It is interesting to note that the separation algorithms can roughly retrieve the original source signals even though they are not completely independent or not even uncorrelated (which is a considerably milder condition than independence). The output images given by these algorithms are in fact more independent than the original correlated source images. This is especially true for the natural/relative gradient rule which tries to minimize a measure of independence, namely the Kullback–Leibler divergence.

The next topic of this chapter was the behavior of the separating networks when a significant amount of additive and/or multiplicative noise in the sensor data appears. Summarizing the separation and noise cancellation ability of our approach, we conclude that:

1. if the environment noise is directly measurable (available), the mixing matrices can have moderate condition numbers of the order 200 or less,

2. if only a convoluted image of environment noise is available, the mixing matrices should have rather low condition numbers, less than 50, in order to achieve good restoration results.

To re-cap, with small additive noise in the sensor signals, our separation approach is able to separate signals mixed by very ill–conditioned mixing matrices, with condition numbers even larger than 10^6.

For the restoration of unknown source signals from their signal mixtures distorted by large additive noise we proposed an adaptive approach, which is restricted to situations where the unknown colored (or Gaussian) noise can be modeled as a convoluted noise mixture of known reference noises. The approach was tested on image sources and sound sources, but it is generally applicable to various classes of non–Gaussian signals, also to speech signals and biomedical signals.

The pre–processing based noise cancellation scheme is able to eliminate or almost completely to cancel the additive noise, i.e. if the original environment noise is measurable the noise is fully canceled, but if a convoluted image of such noise is measurable only, the additive noise is nearly eliminated at this stage. In the subsequent blind separation stage in the first case the sources are very well separated, however, in the second case the noise is amplified and appears in one of the outputs, so one source signal is usually lost. If the mixing matrix is very ill–conditioned the noise can appear also on other outputs, corrupting the $(m-1)$ separated sources.

The same behavior has been observed in the case of simultaneous separation and noise cancellation. In practice, for this basic approach the appropriate decay of the learning rates may be more difficult to establish, i.e. without careful study of this problem the separation results will usually be slightly worse than the results of the simplified approach. If the convoluted image of the environment noise is available only, an overdetermined sensor case is required. Usually one or two additional sensors are sufficient for successful extraction of all sources.

Finally a solution to blind source extraction was proposed, i.e. the BSS problem if fewer sensors than sources are available. This method is restricted to specific sources, like sparse signals or digital signals sufficiently continuous in time. Computer experiments are very promising.

4 AN ADAPTIVE APPROACH TO IMAGE COMPRESSION AND CLASSIFICATION

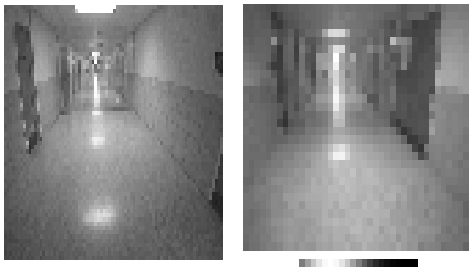
In this chapter an adaptive (ANN-learning based) approach to the classification of well-framed images, preceded by a compression step, is developed. The approach is based entirely on image space transformation. In the first section the problem is introduced and some related approaches are referenced. Then the three-step analysis system (preprocessing, compression, classification) is generally introduced. In the second section a very efficient ANN learning-based algorithm for PCA (principal component analysis) is proposed, verified and tested on real data. This algorithm was developed by the author and his co-workers quite recently and its capability for fast and accurate determination of the PCA space was demonstrated [CIK96] - section 4, [KAS96b]. Alternative adaptive solutions for the PSA (principal subspace analysis) are also proposed and tested in this section [KAS96b]. In the third section a new adaptive method for the classification stage is developed. Finally the applicability of this approach is discussed and illustrated by sample applications.

4.1 THE IMAGE CLASSIFICATION APPROACH

Different image classification systems can be distinguished regarding the type of representation *(image features)*: classification in image space, in the frequency domain and by the use of geometrical features.

In the first approach (classification in image space) a $N \times M$ iconic image is scanned into a signal vector, constituting a point in a space of size $N \times M$. Hence there is a direct classification of images in such a space. Such a type of image classification was applied in two well-known vision-based robot navigation systems: ALVINN (developed at CMU) and ROBIN (developed at Univ. of Maryland) [CHE98]. Both systems map visual inputs directly into output control signals, i.e. they associate with each image an appropriate control class. ALVINN uses a two-layer feedforward network, whereas ROBIN uses a RBF (radial basis function) network. The output pattern of each training sample is a Gaussian distribution peaked at the desired control class, i.e. the corrected heading direction of the robot. After the neural network is trained, the stimulated output heading is taken as the peak of a Gaussian fit to the outputs of neurons at output layer (*Figure* 4.1). The proposed image size reduction is a geometric transformation only. Another system, that uses a real image compression step with subsequent classification, is described and conventionally implemented (by numerical methods) in [SWE96].

Image classification in the frequency domain requires a transformation of the image into a frequency domain. Alternatively the image can be considered as a global or local region, and the frequency features can be computed over one global or many local



(a) (b), (c)

Figure 4.1 Example of a training image pair: (a) the input image is (b) reduced in size to 30×40 and (c) the output vector, which achieves its maximum at the correct heading direction [CHE98].

regions. The classification takes place in the space spanned over the frequency features (for example FFT or wavelet features) [SKA95].

The classification using geometric features requires a pre-detection of discrete image features, like pixel blocks, edges or regions. For example a connectionist, multi-layer ANN system, called *Cresceptron* [WEN94], handles internal image representation in terms of geometrical features. In fact only edge images are used in order to better handle lighting variations. The input to the classification network, called the *attention image*, corresponds to an image window of pre-determined size. The system is organized around a hierarchical grouping process, i.e. smaller object features (edge segments) are grouped along with their neighbors to intermediate-level features and later the higher-level features are combined to the object. The neural network consists of multiple levels, with 2 or 3 feed–forward layers at each level. At each layer there are many neural square planes, where each plane represents a concept and the response at a certain location on the plane indicates the presence of the concept. The nodes at layer 0 correspond to pixels in the input (attention image). Three types of layer are distinguished: (1) the pattern-detection layer, (2) the node-reduction layer and (3) the blurring layer (it blurs the input so that shape generalization is made possible).

4.1.1 The proposed scheme

The proposed image classification scheme consists of three consecutive steps:

- image framing - detection of a well-framed image window,
- image compression - reducing the window vector x to a smaller vector y,
- classification of vector y in the (learned) classification space.

The step of image framing performs image enhancement, detection of an image window of proper size with an eventual rescaling and finally a scanning of this window to a signal vector (image window of size $n \times m = M$ is scanned to an input vector x of size $M \times 1$).

In the compression step to the input vector the Karhunen–Loeve transformation (obtained by PCA or PSA methods) is applied. The principal components of the autocorrelation matrix of samples of input vectors are computed (adaptively learned) on sample data and then they are applied for input vector compression (only the N most important principal components are used) - this compression is based on the principal

component analysis (PCA). Alternatively the PSA (principal subspace analysis) can be applied for vector signal compression.

The classification step is performed via a discriminant analysis (DA) based detection of classes in the reduced classification space of size N. The basis vectors of this transformation are called the "Most Discriminating" features (MDFs)).

The image framing step can be performed in an adaptive way - a representative approach is described in [ROW98]. In the learning phase - the selection of the attention window is made "by hand". Then automatically a new concept is generated, the network is updated and incremented, recursively from lower level to higher level. During the active work several sizes of the attention window are allowed. With each fixed window size, the image is covered by the window at a set of grid points. Each image covered by the window is scaled down to the size of the standard attention window (for example of size 64×64 pixels).

In this work adaptive realizations of the two space transformation steps (methods of PCA/PSA and DA) are proposed. Let us define the discriminant Karhunen-Loeve transformation (DKL) as:

$$\boldsymbol{Z} = \boldsymbol{D}\boldsymbol{W}\boldsymbol{X}, \tag{4.1}$$

where the KL transformation (as the result of PCA or PSA analysis) is:

$$\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}, \tag{4.2}$$

and the subsequent DA transformation is:

$$\boldsymbol{Z} = \boldsymbol{D}\boldsymbol{Y}. \tag{4.3}$$

4.1.2 PCA-based image compression

Principal Component Analysis (PCA) is a powerful data analysis tool in multivariate statistics [JOL86, AMA77]. Up to now many neural network learning algorithms have been proposed for the PCA and its generalizations (e.g. [OJA92, CIC94, KAR94]). The well known Oja's Learning Algorithm ([OJA82]) has been further extended in [SAN89] (the GHA method), [KUN91] (the APEX method), [ABB93] (the SAMHL) and [BAN95] (the RLS method). In recent papers several new theoretic developments in neural network based PCA have been described (e.g. [BAL89, TAY93, XU92]). PCA has also been widely studied and used in pattern recognition and signal processing [NIW93, TUR91].

Since $\boldsymbol{W}$ is a semi–positive definite matrix it can be reconstructed as:

$$\boldsymbol{W} = \sum_{r=1}^{n} \lambda_r \boldsymbol{u}_r \boldsymbol{u}_r^T = \boldsymbol{U}\Lambda\boldsymbol{U}^T \tag{4.4}$$

where $\boldsymbol{U}$ represents the matrix of eigenvectors $\boldsymbol{u}_r$ of $\boldsymbol{W}$ and Λ the diagonal matrix of positive eigenvalues λ_r with n being the rank of the matrix $\boldsymbol{W}$. Each input vector $\boldsymbol{x}_k$ can be expressed as a linear combination of eigenvectors:

$$\hat{\boldsymbol{x}}_k = \sum_{r=1}^{n} \lambda_r c_{k,r} \boldsymbol{u}_r \tag{4.5}$$

with $c_{k,r} = \boldsymbol{u}_r^T \boldsymbol{x}_k$.

In order to determine the number m of PC features to use for vector compression, we first rank the eigenvalues of $\boldsymbol{W}$: $\lambda_1, \lambda_2, ..., \lambda_n$, in a non-decreasing order. The residual mean-square error of using only a subset of m PC features ($m < n$) is:

$$\sum_{i=m+1}^{n} \lambda_i. \tag{4.6}$$

We let m satisfy

$$\frac{\sum_{i=m+1}^{n} \lambda_i}{\sum_{i=1}^{n} \lambda_i} < P. \tag{4.7}$$

Usually for $P = 5\%$ a good reduction ratio (using only 25% of all PC's) while retaining the most important variance, which is present in the original image, is achieved.

Although the KL projection is well-suited to object representation we shall show that the produced features are not reliable for discriminating between classes, defined by the sample set of vector data. The PCs describe only some major variations in class, such as those due to lighting direction. But these observations may be irrelevant to the problem, how the classes are divided.

4.1.3 Discriminant analysis in classification space

Let $\boldsymbol{D}$ be a projection matrix onto the DA space:

$$\boldsymbol{Z} = \boldsymbol{D}\boldsymbol{Y}. \tag{4.8}$$

$\boldsymbol{Z}$ is a new feature vector from C classes, with class means located at $M_i, 1 = 1, 2, ..., C$. The *within–class* scatter matrix is defined as:

$$\boldsymbol{S}_w = \sum_{i=1}^{c} \sum_{j=1}^{n_i} (\boldsymbol{Y}_j - \boldsymbol{M}_i)(\boldsymbol{Y}_j - \boldsymbol{M}_i)^T \tag{4.9}$$

for n_i samples from class i. The *between–class* scatter matrix is defined as:

$$\boldsymbol{S}_b = \sum_{i=1}^{c} (\boldsymbol{M}_i - \boldsymbol{M})(\boldsymbol{M}_i - \boldsymbol{M})^T, \tag{4.10}$$

where $\boldsymbol{M}$ is the grand mean vector. In discriminant analysis we want to determine the projection matrix $\boldsymbol{D}$ that maximizes the ratio [KUL72]:

$$\frac{\det(\boldsymbol{S}_b)}{\det(\boldsymbol{S}_w)}. \tag{4.11}$$

The discriminant analysis breaks down when the within–class scatter matrix $\boldsymbol{S}_w$ becomes degenerate, when the number of samples is smaller than the dimension of input vectors, i.e. for a small image 64×64 the number of training samples should be greater than 4096. Due to the previous vector compression step, the discriminant analysis is

performed in the KL-transformed space, and in practice the degeneration does not occur (the number of samples is usually sufficient).

The value m should be chosen such that for s training samples from c classes it holds: $(m + c \leq s)$. At the same time m should be less than the rank of matrix $\boldsymbol{S}_w$ in order to avoid a degenerate matrix case. On the other hand, obviously it holds that: $(m \geq c)$.

Since there are at most $c - 1$ nonzero eigenvalues of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$, we choose the value k $(k \leq c - 1)$ to be the final dimension of the discriminant feature space, i.e:

$$k + 1 \leq c \leq m \leq s - c. \tag{4.12}$$

4.2 AN ADAPTIVE APPROACH TO IMAGE COMPRESSION

The main purpose of this section is to develop and to test fast learning algorithms for neural network-based PCA and PSA, in order to select the most reliable and fast algorithm for the image compression task in our image frame classification system. The selected algorithm should converge for every signal of given class and it should precisely extract an arbitrary number of principal components with high convergence speed.

4.2.1 Basic methods of neural PCA

Independently of the neural network learning algorithm that is applied for *principal component analysis* (PCA) [AMA77, TAY93, CIC94], a higher order principal component m can be estimated if, and only if, all the previous components $(1, 2, ..., m - 1)$ are already extracted or exactly estimated. This means that we are not able to extract all the required principal components in a fully *parallel way* (i.e. simultaneously).

The standard PCA, called also the Karhunen–Loeve transformation (KL), determines an optimal linear transformation of an input vector $\boldsymbol{x}$:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}, \tag{4.13}$$

where $\boldsymbol{x} = [x_1(t), x_2(t), ..., x_n(t)]^T \in \mathcal{R}^n$ is a zero–mean input vector, $\boldsymbol{y} = [y_1(t), y_2(t), ..., y_m(t)]^T \in \mathcal{R}^m$ is the output vector, called *principal components*, and $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_n] \in \mathcal{R}^{m \times n}$ is a desired transformation matrix. The orthogonal vectors $\boldsymbol{w}_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$, are called *principal vectors*.

In the standard numerical approach for estimating the principal components, the autocorrelation matrix $\boldsymbol{R}_{xx} = \langle \boldsymbol{x}\boldsymbol{x}^T \rangle$ is computed first and then its eigenvectors and associated eigenvalues are determined next by one of the known numerical algorithms, e.g. the *QR algorithm* or the *SVD algorithm* [JOL86]. A standard numerical approach for computing $\boldsymbol{W}$ is:

$$\boldsymbol{W} = \boldsymbol{D}^{-1/2}\boldsymbol{E}^T, \tag{4.14}$$

with $\boldsymbol{D} = diag[\lambda_1, ..., \lambda_m]$ and $\boldsymbol{E} = [\boldsymbol{c}_1, ..., \boldsymbol{c}_m]$, where λ_i is the i-th largest eigenvalue of the data covariance matrix $\boldsymbol{R}_{xx} = E\{\boldsymbol{x}(t)\boldsymbol{x}^T(t)\}$ and $\boldsymbol{c}_i$ is the respective principal eigenvector.

However, if the input data vectors have a large dimension, the correlation matrix $\boldsymbol{R}_{xx}$ is very large and this may lead to expensive computations. The neural network approach enables us to find the eigenvectors and the associated eigenvalues directly from the input vector $\boldsymbol{x}$ without the need to estimate the correlation matrix $\boldsymbol{R}_{xx}$.

Extraction of First PC – Oja's Rule

In order to find the optimal value of the vector $\boldsymbol{w}_1$ one can formulate (define) a suitable robust *loss (cost) function*, given as [CIC94]:

$$\rho(\boldsymbol{e}_1) = \rho(\boldsymbol{x} - y_1\boldsymbol{w}_1) = \sum_{i=1}^{n} \rho_i(e_{1i}), \tag{4.15}$$

where the $\rho_i(e_{1i})$-s are the individual loss functions.

The minimization of the loss (cost) function according to the standard *gradient descent* approach leads to the generalized learning algorithm:

$$\frac{d\boldsymbol{w}_1(t)}{dt} = -\eta_1(t)\frac{\partial\rho(\boldsymbol{w}_1)}{\partial\boldsymbol{w}_1} = \eta_1(t)\left[y_1(t)\psi(\boldsymbol{e}_1) + \boldsymbol{x}(t)\boldsymbol{w}_1^T\psi(\boldsymbol{e}_1)\right], \tag{4.16}$$

with $\boldsymbol{w}_1(0) \neq O$, where $\eta_1(t) > 0$ is the learning rate and $\psi_i(e_{1i})$-s $(i = 1, 2, ..., n)$ are activation (influence) functions with $\psi_i(e_{1i}) = \frac{\partial\rho(e_{1i})}{\partial e_{1i}}, (i = 1, 2, ..., n)$.

The standard loss function is a quadratic one (the 2–norm criterion) defined as

$$\rho(\boldsymbol{e}_1) = \frac{1}{2}||\boldsymbol{e}_1||^2 = \sum_{i=1}^{n}\rho_i(e_{1i}) = \frac{1}{2}\sum_{i=1}^{n} e_{1i}^2. \tag{4.17}$$

For this function the un-supervised learning rule (4.16) can be simplified to

$$\begin{array}{rcl} \frac{d\boldsymbol{w}_1(t)}{dt} & = & \eta_1(t)\left[y_1(t)\boldsymbol{e}_1(t) + \boldsymbol{x}(t)\boldsymbol{w}_1^T(t)\boldsymbol{e}_1(t)\right] = \\ & = & \eta_1 y_1(\boldsymbol{x} - y_1\boldsymbol{w}_1) + \eta_1\boldsymbol{x}(1 - \boldsymbol{w}_1^T\boldsymbol{w}_1)y_1 = \\ & = & \eta_1 y_1(\boldsymbol{I} - \boldsymbol{w}_1\boldsymbol{w}_1^T)\boldsymbol{x} + \eta_1\boldsymbol{x}(1 - \boldsymbol{w}_1^T\boldsymbol{w}_1)y_1. \end{array} \tag{4.18}$$

Taking into account that $\boldsymbol{w}_1^T(t)\boldsymbol{w}_1(t) \to 1$ as $t \to \infty$ the second term tends quickly to zero and it can be neglected without any influence on the final solution. Thus the learning rule can be simplified as follows:

$$\frac{d\boldsymbol{w}_1(t)}{dt} = \eta_1(t)y_1(t)\boldsymbol{e}_1(t) = \eta_1(t)y_1(t)[\boldsymbol{x}(t) - y_1(t)\boldsymbol{w}_1(t)]. \tag{4.19}$$

For discrete time data or a discrete time realization the learning algorithm in eq. (4.19) can be directly transformed to the well known *Oja's learning rule*: [OJA82]

$$\boldsymbol{w}_1(k+1) = \boldsymbol{w}_1(k) + \eta_1(k)y_1(k)[\boldsymbol{x}(k) - y_1(k)\boldsymbol{w}_1(k)]. \tag{4.20}$$

FOR signal samples $\boldsymbol{x}(k)$: FROM $k = 1$ TO N
(1) set error signal $\boldsymbol{e}_0(k) = \boldsymbol{x}(k)$
FOR every neuron: FROM $j = 1$ TO m
(2) set $\boldsymbol{w}_j(0)$ randomly
(3) set η_j in accordance with input variance
FOR epoch index s: FROM $s = 1$ TO MAX_S
FOR input samples: FROM $k = 1$ TO N
(4) $y_j(k) = \boldsymbol{w}_j^T(k-1)\boldsymbol{x}(k)$
(5) $\boldsymbol{w}_j(k) = \boldsymbol{w}_j(k-1) + \eta_j y_j(k)[\boldsymbol{e}_{j-1}(k) - y_j(k)\boldsymbol{w}_j(k-1)]$
IF $\lvert\boldsymbol{w}_j(k-1) - \boldsymbol{w}_j(k)\rvert < \epsilon$
THEN (6) $\boldsymbol{w}_j = \boldsymbol{w}_j(k)$; GO TO STABLE
(7) decrease η_j exponentially
STABLE:
FOR input samples: FROM $k = 1$ TO N
(8) set $y_j(k) = \boldsymbol{w}_j^T\ \boldsymbol{x}(k)$
(9) set error $\boldsymbol{e}_j(k) = \boldsymbol{e}_{j-1}(k) - y_j(k)\ \boldsymbol{w}_j$

Figure 4.2 Sanger's GHA algorithm implementation.

The GHA, SAMHL and RLS PCA methods

For many applications (e.g. image compression), it is important to extract or to estimate higher principal components in an exact and fast way. One of the first who proposed how to extend the Oja's rule for learning many principal components was Sanger [SAN89] (*Figure* 4.2). From the viewpoint of search for a general method two main drawbacks of his algorithm exist: (1) the initialization of the learning rate η_j was done heuristically using a trial and error approach, and (2) the applied Gram–Schmidt orthogonalization usually tends to de-converge the calculations for higher principal components corresponding to small eigenvalues.

An extension of Sanger's algorithm, called *SAMHL*, was proposed in [ABB93]. Its main feature is a successive (sequential) application of the generalized Hebbian learning rule for each next higher principal component. The advantage of the *SAMHL* method over the Sanger's method is also the explicit calculation and adaptation of the learning rate. The parameter η_j is initialized according to the variance of the whole original input sequence σ^2_{all} and to the set of eigenvalues λ_i, corresponding to the previous eigenvectors (step (3)):

$$\eta_j(0) = \frac{1}{\sigma^2_{all} - \sum_{i=1}^{j-1}\lambda_i}\ ,\ \lambda_i = \frac{1}{N}\sum_{k=1}^{N} y_i(k)^2. \tag{4.21}$$

The learning rate is adapted after some specified number of iteration steps q. This is due to the fact, that the adaptation equation is relatively complex and requires the computation of current signal variances σ^2 and $\Delta\boldsymbol{w}(k)$ over the whole input set. The rule is based on the minimization of some heuristic cost function and it has the following form

(step (7)):

$$\eta_j(q_{k+1}) = \eta_j(q_k) + cN[1 - \eta_j(q_k)\sigma^2(q_k)] \left[\sigma^2(q_k) + \frac{2}{N}\sum_{i=1}^{N} y_i(\boldsymbol{e}_j(q_k)\Delta\boldsymbol{w}_j(q_k))\right], \quad (4.22)$$

where c is the heuristic constant which is set to: $c = 0.01/\lambda_{j-1}$, $(k = 1, 2, ...)$.

In [BAN95] the *recursive least square learning* (*RLS*) was proposed for Sanger's GHA algorithm. A dynamic Kalman gain K was introduced in the updating equation for determining the learning coefficient η_j. For this purpose an RLS algorithm was applied. The learning rate $\eta_j(k)$ is now modified for each new image sample according to current gain $K_j(k)$ (step 7):

$$K_j(k) = \eta_j(k)y_j(k) = \frac{P_j(k-1)y_j(k)}{[1 + P_j(k-1)y_j(k)^2]}, \quad (4.23)$$

$$\text{where} \quad P_j(k) = [1 - K_j(k)y_j(k)]P_j(k-1). \quad (4.24)$$

Step (5) takes now the following form:

$$\boldsymbol{w}_j(k) = \boldsymbol{w}_j(k-1) + K_j(\boldsymbol{e}_j(k) - \boldsymbol{w}_j(k-1)y_j(k)). \quad (4.25)$$

Let us note that the adaptation of the learning rate (step 7) is now shifted to the iteration cycle and is placed between steps (4) and (5). But the authors applied the Gram–Schmidt orthogonalization (GHA Sanger's algorithm), which again prohibits a reliable extraction of all components (due to the accumulation of errors).

4.2.2 The proposed CRLS PCA method

The proposed algorithm, called *CRLS* (cascade recursive least square) combines the advantages of both previous methods, *SAMHL* and *RLS*, and eliminates their drawbacks.

The task of principal component extraction can be easily accomplished by using a *cascade neural network*. The learning algorithm for the estimation of the next largest principal component is performed in the same way as for the first component but the extraction process works not directly on the original input data $\boldsymbol{x}(t)$ but on the available errors remaining after previous extractions (this procedure is called *deflation*). Depending on the number of required principal components m an appropriate number of m cascades can be stacked one after the other. The training of this network continues until all weight vectors $\boldsymbol{w}_j, (j = 1, ..., m)$ are stabilized.

The algorithm of the *CRLS* (cascade recursive least square) method is given in *Figure* 4.3. The main differences to the Sanger's method appear in steps 3,4 and 7.

For the calculation of the j-th eigenvalue, $\lambda_j = E\{y_j^2\}$, the reduced input signal $\boldsymbol{E}_j = [\boldsymbol{e}_j(1), \boldsymbol{e}_j(2), ..., \boldsymbol{e}_j(N)]$ is applied (and not the original input signal) (step 4):

$$y_j(k) = \boldsymbol{w}_j(k-1)\boldsymbol{e}_j(k), \quad (4.26)$$

where $\boldsymbol{e}_j(k) = [e_{j1}(k), e_{j2}(k), ..., e_{jn}(k)]^T$, $(k = 1, 2, ..., N)$. This, at first view, slight modification has an important influence on the orthogonality of the computed weights

FOR signal samples $\boldsymbol{x}(k)$: FROM $k=1$ TO N		
(1) set error signal $\boldsymbol{e}_0(k) = \boldsymbol{x}(k)$		
FOR every neuron: FROM $j=1$ TO m		
(2) set $\boldsymbol{w}_j(0) = \mathbf{1}$		
(3) set $1/\eta_j(0) = \sigma^2[\boldsymbol{e}_{j-1}]/N$		
FOR epoch index s: FROM $s=1$ TO MAX_S		
FOR input samples: FROM $k=1$ TO N		
(4) $y_j(k) = \boldsymbol{w}_j(k-1)\boldsymbol{e}_{j-1}(k)$		
(7) $1/\eta_j(k) = y_j(k)^2 + 1/\eta_j(k-1)$		
(5) $\boldsymbol{w}_j(k) = \boldsymbol{w}_j(k-1) + [y_j(k)\eta_j(k)][\boldsymbol{e}_{j-1}(k) - \boldsymbol{w}_j(k-1)y_j(k)]$		
IF	$\lvert\boldsymbol{w}_j(k-1) - \boldsymbol{w}_j(k)\rvert < \epsilon$	
THEN	(6) $\boldsymbol{w}_j = \boldsymbol{w}_j(k)$; GO TO STABLE	
STABLE:		
FOR input samples: FROM $k=1$ TO N		
(8) set $y_j(k) = \boldsymbol{w}_j\boldsymbol{e}_{j-1}(k)$		
(9) set error $\boldsymbol{e}_j(k) = \boldsymbol{e}_{j-1}(k) - y_j(k)\boldsymbol{w}_j$		

Figure 4.3 The CRLS algorithm for the adaptive PCA.

and consequently on their exactness. In contrast to other known algorithms the extraction of higher components does not tend to zero and the successive extractions are provided in a numerically stable way.

The learning rate η_j is now initialized according to the variance of the whole original input sequence σ^2_{all} only (step 3): $1/\eta_j(0) = \sigma^2[\boldsymbol{e}_{j-1}]/N = E[y^2_{j-1}]$.

The learning rate is adaptively and optimally modified during a training process. The Kalman form of the RLS model (4.23) is transformed to an equivalent and simple formula (step (7)). This form of the learning rate ensures a considerable improvement in convergence speed.

4.2.3 RLS technique in neural PSA

An alternative approach to signal (or image) compression and feature extraction is the *principal subspace analysis* (PSA) [OJA92]. Its advantage is a fully parallel working ability, i.e. a simultaneous calculation of the subspace spanned by a specified number of principal components. But instead of a relatively simple scalar algebra, like in sequential PCA, a computationally more expensive but compact matrix algebra is required for PSA. From a parallel method we usually expect that good quality results will be available in a very fast manner. This is not automatically guaranteed by the original PSA algorithm which converges very slowly. A class of quasi–parallel PCA algorithms, called *ordered* PSA, can also be considered in this context [BRO91, OJA92, SAN89]. These methods are given in matrix form, like the PSA method, but they do not work fully parallel due to additional control by specific *ordering operators* (matrix).

After computational tests, the author claims, that the neural PSA methods are not able to learn *all* subspace components in a reasonable amount of time [KAS96b]. But at least, the major subspace vectors can be learned in a fast manner, and they are sufficient for our needs - to provide a compression of the image vector.

PSA learning rules

The adaptive algorithm of *principal subspace analysis* (PSA) was developed by Oja and Karhunen [OJA92]. It can be written in a generalized (modified) form as

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \boldsymbol{\eta}(t)[\boldsymbol{y}(t)\boldsymbol{x}^T(t) - \boldsymbol{y}(t)\boldsymbol{y}^T(t)\boldsymbol{W}(t)], \tag{4.27}$$

where $\boldsymbol{y}(t) = \boldsymbol{W}(t)\boldsymbol{x}(t)$ and $\boldsymbol{\eta}(t)$ is a suitable positive–definite matrix.

The PSA algorithm is able to learn only a rotated basis of the PC's subspace, i.e. PSA determines the subspace spanned by the first m ($m < n$) principal eigenvectors with imposed constraint

$$\boldsymbol{w}_j^T \boldsymbol{R}_{xx} \boldsymbol{w}_i = 0; \quad \text{for } i \neq j; \quad \text{where } \boldsymbol{R}_{xx} = E\{\boldsymbol{x}\boldsymbol{x}^T\}. \tag{4.28}$$

In order to extract the true principal components some non-symmetry must be introduced in the learning rule or some nonlinearity must be incorporated. Two learning algorithms from a class called *ordered PSA* will be tested: the Brockett subspace algorithm (BSA) and Sanger's generalized Hebbian algorithm (GHA). The Brockett learning rule differs from the PSA rule by the introduction of a nonsingular diagonal matrix $\boldsymbol{D}$ [BRO91]:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \boldsymbol{\eta}(t)[\boldsymbol{D}\boldsymbol{y}(t)\boldsymbol{x}^T(t) - \boldsymbol{y}(t)\boldsymbol{y}^T(t)\boldsymbol{D}\boldsymbol{W}(t)], \tag{4.29}$$

$$\boldsymbol{D}(t) = diag(d_1, d_2, ..., d_m); \quad \text{where } 1 > d_1 > d_2 > ... > d_m > 0. \tag{4.30}$$

The second modified form of ordered PSA, used in experiments, is the generalized Hebbian algorithm (GHA), proposed by Sanger [SAN89]:

$$\boldsymbol{W}(t+1) = \boldsymbol{W}(t) + \boldsymbol{\eta}(t)[\boldsymbol{y}(t)\boldsymbol{x}^T(t) - LT(\boldsymbol{y}(t)\boldsymbol{y}^T(t))\boldsymbol{W}(t)]. \tag{4.31}$$

$LT(.)$ means the Lower Triangular operation, i.e. it sets the above diagonal entries of the matrix to zero.

RLS technique in matrix form

Let us apply the *recurrent least squares* (RLS) learning rate adaptation for parallel *principal subspace analysis* (PSA) and for *principal component analysis* (PCA) algorithms with a quasi–parallel extraction ability. The purpose is to provide a useful tool for applications, where the learning process has to be repeated in an on–line self–adaptive manner. In such a case it is important to provide the fast learning convergence of parallel PSA and quasi–parallel PCA methods.

The initial learning rate is a diagonal matrix with values on the diagonal equal to $\boldsymbol{\eta}(0) = (\sigma_{all}^2)^{-1}\boldsymbol{I}$; where σ_{all}^2 is the variance of the input signal. After t steps the learning rate can be computed as follows:

$$\boldsymbol{\eta}(t) = [\sum_{k=1}^{t} \boldsymbol{y}(k)\boldsymbol{y}^T(k)]^{-1} = \boldsymbol{\eta}(t-1) - \frac{\boldsymbol{\eta}(t-1)\boldsymbol{y}(t)\boldsymbol{y}^T(t)\boldsymbol{\eta}(t-1)}{\boldsymbol{y}^T(t)\boldsymbol{\eta}(t-1)\boldsymbol{y}(t)}. \tag{4.32}$$

Let us note that in the above formulation $\boldsymbol{\eta}(t)$ is a matrix and not a scalar. The elements of this matrix converge to zero with different speed during the learning process according to the above rule.

$(a) PSNR = 24.46\ dB$ $(b) PSNR = 28.12\ dB$

Figure 4.4 The reconstructed image *Lenna* after the 1-st (a) and 3-d (b) PC.

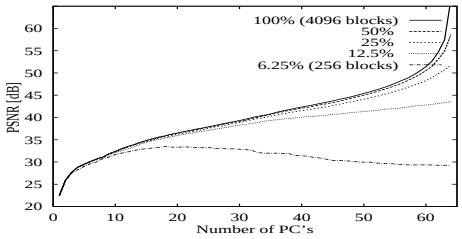
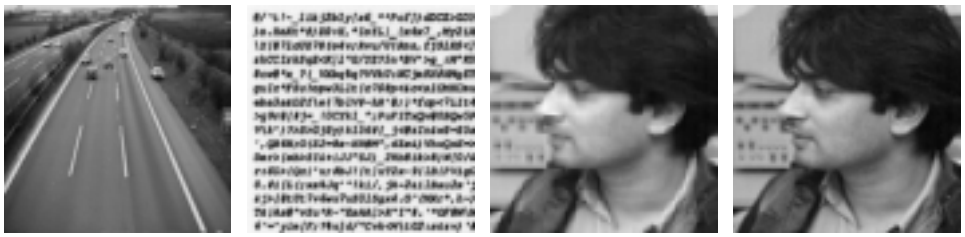


Figure 4.5 The PSNR of the reconstructed image depending on the part of the image used for training in the *CRLS*–PCA method.



(a) $PSNR = 33.41dB$ (b) $PSNR = 21.39dB$ (c) $PSNR = 29.73dB$ (d) $PSNR = 34.71dB$

Figure 4.6 Image compression–reconstruction with components learned on *Lenna*: applying 8 principal components of *Lenna* for (a) *Road* and (b *Text* compression–reconstruction, and (c) 1 and (d) 3 principal components of *Lenna* for *Ali* compression–reconstruction.

4.2.4 Test results of neural PCA/PSA

Several grey scale images (an 8 bit per pixel representation) were used for tests of the PCA and PSA adaptive learning algorithms. Their sizes range from 512 x 512 pixel for *Lenna* and the NIST sub-image *Text* over 384 x 288 pixel for the traffic scene image *Road*, 350 x 400 for image *Ali* down to 128 x 100 pixel for the MIT Media Lab. images *Brad* and *Plant*. Usually the images are divided into blocks of 8×8 pixels each and converted into vector samples of 64 elements.

The evaluation of the PCA/PSA methods is performed according to the quality of the image compression–reconstruction procedure based on extracted principal components or subspaces. In practice the extracted PC-s or PS-s constitute the weight matrix $\boldsymbol{W}_j$ (of size $j \times 64$, where $j = 1, 2, ..., 64$) and the quality assessment procedure consists of the three following steps: 1) $\boldsymbol{y}(t) = \boldsymbol{W}_j \boldsymbol{x}(t)$, 2) $\widehat{\boldsymbol{x}}(t) = \boldsymbol{W}_j^T \boldsymbol{y}(t)$, 3) compute $PSNR[\widehat{\boldsymbol{x}}, \boldsymbol{x}]$, where $\boldsymbol{W}_j \in R^{j \times 64}$, $\boldsymbol{x}(t) \in R^{64}$, $\boldsymbol{y}(t) \in R^j$, $j = 1, 2, ..., 64$.

CRLS PCA

In *Figure* 4.4 some results of processing the well known image *Lenna* are provided. It can be observed visually that already after the computation of three principal components the reconstructed image is of high quality. This quality is proved by the drawings in *Figure* 4.5. The peak signal to noise ratio (PSNR) is shown for all 64 principal components for several sets of training data from the same image. No more than one epoch

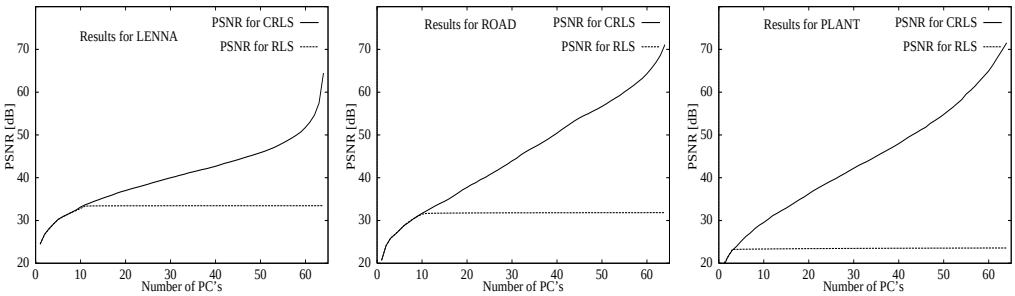


Figure 4.7 Comparison of image compression–reconstruction performances using the CRLS and RLS algorithms: PSNR values of reconstructed images *Lenna*, *Road* and *Plant*.

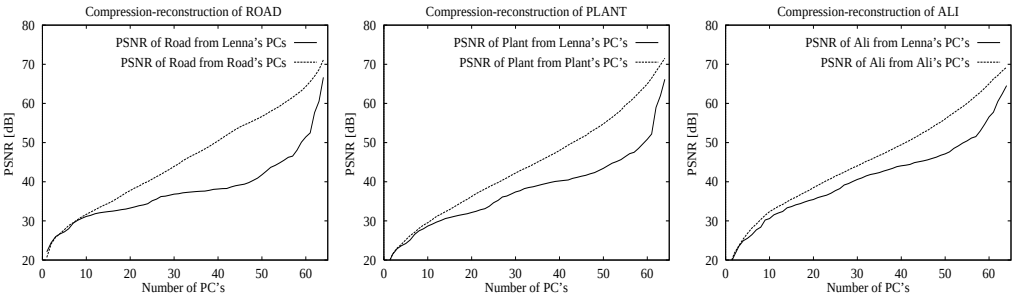


Figure 4.8 Comparison of performances for compression–reconstruction of test images, alternatively using principal components learned for the image *Lenna* or the image–own principal components.

of the learning sample was required in order to fulfill the weight convergence condition $\Delta \boldsymbol{w}(k) < 10^{-5}$. Even the use of a part of training data (incomplete image) still allowed a reliable extraction of the first 16 components – the differences for the first 16 principal components are negligible.

In order to compare our **CRLS** method with the **RLS** approach we have implemented the RLS–algorithm described in [BAN95]. CRLS and RLS both converge with similar speed and determine the first several principal components with similar quality, but RLS does it only up to a certain number of components. For the *Lenna* image the RLS algorithm after extraction of the first 10 components was not able to extract higher components (*Figure* 4.7). For the image *Plant* the situation was much worse – RLS found four components only. In contrast to RLS our CRLS algorithm monotonically increased the quality of reconstructed images as we added the next components.

One of the most important features of the PCA approach to data representation is its generalization power. It is expected that when the principal components obtained for one image I_1 are applied to another image I_2, they should provide similar quality of reconstruction as the components obtained directly for the image I_2. In *Figure* 4.8 this generalization power of PCA is illustrated. The reconstructions of different images on the basis of PC's obtained from the image *Lenna* were compared to reconstructions of these images on the basis of the original image–own principal components. These ex-

(a) 8 PCs in 8 epochs best: 24.92 *dB* (CRLS) (b) 8 PCs in 1 epoch best: 22.83 *dB* (BSA)

Figure 4.9 Reconstruction quality (PSNR) of image *Girl* using its 8 PC/PS-s, learned in (a) 8 epochs or (b) 1 epoch of data (optimum quality was 24.96 *dB* after 22.51 epochs of CRLS) [KAS96b].

(a) 8 PCs in 8 epochs best: 33.11 *dB* (CRLS) (b) 8 PCs in 1 epoch best: 25.79 *dB* (GHA)

Figure 4.10 Reconstruction quality (PSNR)of image *Road* using its 8 PC/PS-s, learned in (a) 8 epochs or (b) 1 epoch (optimum quality was 33.41 *dB* after 42.9 epochs of CRLS learning) [KAS96b].

periments document that there is a negligible difference between PSNR plots for both cases for the first 16 components.

Comparison of RLS PSA and CRLS PCA

In *Figure* 4.9 and 4.10 two image reconstruction results are shown, by applying 8 principal components or 8 subspace vectors respectively extracted by three methods with RLS learning (CRLS, PSA, BSA). Two cases of the learning time are considered: 1) the epoch number is equal to the number of requested PC/PS-s (e.g. 8 epochs for 8 PS/PC-s) and 2) there is only one epoch independent of the number of requested PS/PC-s.

For quality judgment the best possible reconstruction quality should also be known. This optimum was found by the CRLS method in a relatively long learning process. Usually more than one epoch of the image data for every PC is required in order to fulfill both the weight stability condition $\Delta \boldsymbol{w}_j < 10^{-5}$ and the normalization to unit length condition, i.e. $|1 - ||\boldsymbol{w}_j||| < 10^{-2}$.

Quantitative results of image reconstruction related to above tests are provided in *Figure* 4.11, 4.12. The peak signal to noise ratio (PSNR) is shown in all drawings. From the above figures it is clearly evident that the sequential extraction of the principal components by the cascaded PCA method ensures very high quality. The sequential CRLS method achieves the best performance among all of the tested methods if the training time is one epoch for each component. As far as the first 4 PC–s are concerned using this method they can also be learned in one epoch (i.e. 1/4 of the image data can be used for learning one component) with good quality. A further time shortage for one component by requesting more than four PC-s in one epoch, leads to lower quality of the CRLS results. If the neural network has to learn in an on–line manner in one epoch of image data and more than 6.25 % PC/PS-s (i.e. more than 4 from the set of 64) are needed then a quasi–parallel PCA method gives better results.

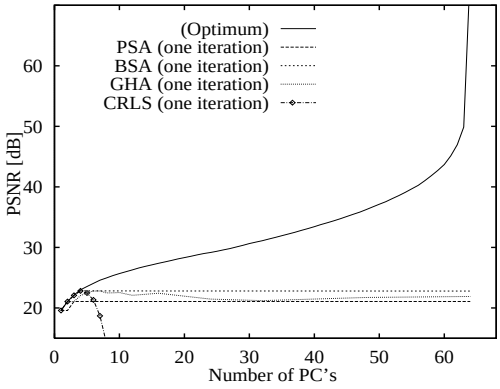


Figure 4.11 The PSNR of the reconstructed image *Girl* while learning in one epoch j PC/PS's.

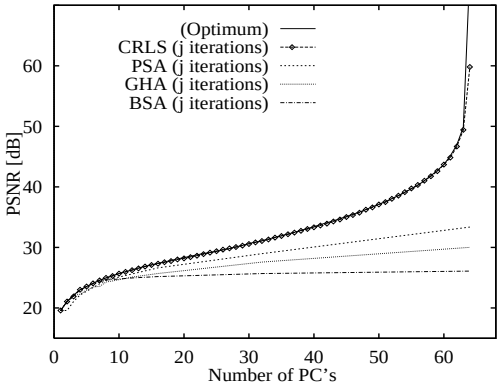


Figure 4.12 The PSNR of the reconstructed image *Girl* while learning in j epochs j PC/PS's.

4.3 AN ADAPTIVE APPROACH TO DA TRANSFORMATION

Initially, some study on the applications of linear auto-associating networks and PCA (both have similar linear function theory) for image classification, especially as applied to face images, has been provided [TUR91, VAL96]. The results and our experience in PCA-like methods bring us to the conclusion that it is not the major principal components that have a discriminating power, but rather the intermediate-level and minor principal components that are more useful for differentiation between image classes.

Hence back-propagating networks have been studied more intensively for face image classification [COT91, VAL94]. In this section, the discrimination analysis (DA) is applied for image classification and a nonlinear LVQ network is proposed as an adaptive approach to DA. Some related research deals with nonlinear PCA networks that may be applied for nonlinear image feature extraction (e.g. modelling of 2-D nonlinear equations by principal curves) [HAS89].

4.3.1 Standard approach

As we already know from equation (4.11) in discriminant analysis we want to determine the projection matrix $\boldsymbol{D}$ that maximizes the ratio of the between–class scatter to the within–class scatter. It is known that this ratio is maximized when the column vectors of projection matrix $\boldsymbol{D}$ are the eigenvectors of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$, associated with the largest eigenvalues [CHE98, SWE96]. The scalar components in $\boldsymbol{Z}$ are feature values of given sample and the column vectors of $\boldsymbol{D}$ are the MDF feature vectors. Because the matrix $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ need not be symmetric, the eigensystem computation could be unstable. The numeric method described in *Table* 4.1 diagonalizes two symmetric matrices [CHE98]. Here, using the symmetric properties of the component scatter matrices, we define a method for finding the eigensystem of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ that is stable.

1. Compute $\boldsymbol{H}$ and Λ such that $\boldsymbol{S}_w = \boldsymbol{H}\Lambda\boldsymbol{H}^T$, where $\boldsymbol{H}$ is orthogonal and Λ is diagonal. Then $(\boldsymbol{H}\Lambda^{-\frac{1}{2}})^T\boldsymbol{S}_w\boldsymbol{H}\Lambda^{-\frac{1}{2}} = \boldsymbol{I}$.
2. Now compute $\boldsymbol{U}$ and $\boldsymbol{V}$ such that $(\boldsymbol{H}\Lambda^{-\frac{1}{2}})^T\boldsymbol{S}_b\boldsymbol{H}\Lambda^{-\frac{1}{2}} = \boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^T$, where $\boldsymbol{U}$ is orthogonal and $\boldsymbol{V}$ is diagonal. Then:

$$\boldsymbol{S}_b = \boldsymbol{H}\Lambda^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^T\Lambda^{\frac{1}{2}}\boldsymbol{H}^T, \boldsymbol{S}_w = \boldsymbol{H}\Lambda^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{I}\boldsymbol{U}^T\Lambda^{\frac{1}{2}}\boldsymbol{H}^T. \tag{4.33}$$

3. Define: $\Delta = \boldsymbol{H}\Lambda^{-\frac{1}{2}}\boldsymbol{U}$. Δ diagonalizes $\boldsymbol{S}_b$ and $\boldsymbol{S}_w$ at the same time.
   Since $\boldsymbol{S}_w^{-1} = \boldsymbol{H}\Lambda^{-1}\boldsymbol{H}^T$, equation (4.33) gives

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b = \boldsymbol{H}\Lambda^{-1}\boldsymbol{H}^T\boldsymbol{H}\Lambda^{\frac{1}{2}}\boldsymbol{U}\boldsymbol{U}^T\Lambda^{\frac{1}{2}}\boldsymbol{H}^T = \boldsymbol{H}\Lambda^{-\frac{1}{2}}\boldsymbol{U}\boldsymbol{V}\boldsymbol{U}^T\Lambda^{\frac{1}{2}}\boldsymbol{H}^T = \Delta\boldsymbol{V}\Delta^{-1}. \tag{4.34}$$

4. That is Δ consists of the eigenvectors of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ and $\boldsymbol{V}$ contains the eigenvalues of $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$.

Table 4.1 Numerical computation method for discriminant analysis [SWE96, CHE98].

1. N output neurons exist. Each output neuron i represents a class label C_i.
2. A learning sample p consists of an input vector $\boldsymbol{x}^p$ and its correct class label d^p.
3. Two winners are determined - the best one k and the second best l, by using distance measures between weights $\boldsymbol{w}_i$ of $i-$th output neuron and input $\boldsymbol{x}$:

$$|\boldsymbol{x} - \boldsymbol{w}_k| < |\boldsymbol{x} - \boldsymbol{w}_l| < |\boldsymbol{x} - \boldsymbol{w}_i|, \forall i, i \neq k, l. \tag{4.35}$$

4. The class labels C_k and C_l are compared with d^p and two weight update rules are used according to the following strategy: let $C_k \neq d^p$ and $d^p = C_l$, and

$$\text{If} \quad |\boldsymbol{x}^p - \boldsymbol{w}_k| - |\boldsymbol{x}^p - \boldsymbol{w}_l| < \epsilon \tag{4.36}$$

$$\text{then} \quad \boldsymbol{w}_l(t+1) = \boldsymbol{w}_l(t) + \eta[\boldsymbol{x}(t) - \boldsymbol{w}_l(t)] \tag{4.37}$$

$$\text{and} \quad \boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) - \eta[\boldsymbol{x}(t) - \boldsymbol{w}_k(t)]. \tag{4.38}$$

Table 4.2 Neural supervised LVQ method for discriminant analysis.

4.3.2 Supervised LVQ approach for DA

For an adaptive solution of the DA problem we have developed the following adaptive algorithm, which can be called a *supervised LVQ* algorithm (see *Table* 4.2). During learning, the vector $\boldsymbol{w}_l$ corresponding to the correct label is moved towards the input vector, whereas the vector $\boldsymbol{w}_k$ with the incorrect label, which may even be nearest the input, is moved away from it. In the active classification phase the network works according to the equations 2.9 and 2.10, specified in chapter 2.

4.4 TESTS AND APPLICATIONS

4.4.1 Test example

In our test example the network learned 4 classes of images of size 64×64, reduced from original size 256×256. The system was trained with 75 images from 4 classes: face

Figure 4.13 Examples of four image classes: face, natural, text and synthetic, used during learning (top row) and active work (bottom row) of the DKL-classification system.

(F), natural (N), text (T) and synthetic (S) images (*Figure* 4.13). Nearly the same set was used for testing, i.e. only 35 images have been exchanged by new images.

The PCs showed a tendency to capture major variations such as lighting direction. The MDFs – have the ability to discount features unrelated to classification, like for example imaging artifacts, and to tolerate within–class variations. In the DA space (the MDF features) objects of the same class are clustered much more tightly than in the KL space (PC features) (*Figure* 4.14). Already two most important PCs allow a fairly good clustering of all class images together, whereas already the two most discriminating features nearly completely separate the classes. The distance between clusters cannot directly be regarded as some closeness measure. Following performance ratios for face images were obtained during testing:

- 95% of the MDF subspace variance was covered by 15 features.
- In order to retain 95% of the KL subspace variance at least 36 features were needed.
- By using 95% of the KL variance - a recognition rate of 90% was reached, and it was not improved by using more features.

4.4.2 Other applications

Classification of hand movements

Swets et al. [SWE96] have applied a similar system (designed in traditional, numeric computation mode) for the classification of N image frames extracted from a short image sequence "by-hand". The system can be applied to image sequences, after proper preprocessing and input vector determination. The goal is to track the motion of hand and to classify the dynamic change of its gesture as indicating a hand sign. In a test example [SWE96] 28 hand sign classes were used for training and testing, as defined in the *American Sign Language*. Two representations are needed, one for spatial recognition and one for spatio-temporal recognition from the fovea vectors (frames). Each sub-image that includes the hand is stacked along three axes, i.e. it gives a vector of $(s_x s_y s_t)$ dimensions. The global motion vectors of the hand are represented in the $x-y$ coordinates. Thus the combination of cropped sequence and the global motion vector

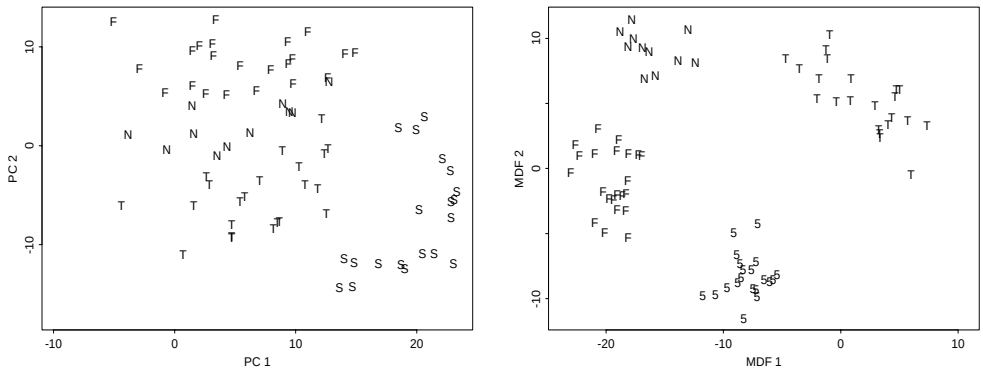


Figure 4.14 Images from the learning set represented in the subspaces spanned by: the first two PCs (left) and by the first two MDFs (right).

defines the input *fovea vector*, which is of dimension $(s_x s_y s_t + 2)$. For each of the 28 hand sign classes, 36 sample sequences were available, half of which was used for tracking and the other half for testing. The system correctly recognized 98% of the test sequences.

Autonomous robot navigation

On the basis of a conventional DKL classification system, Chen et al. [CHE98] designed a system for vision-based control of in–door robot navigation. In the reported test 318 images were collected for training, at each position a set of five images was obtained with different necessary corrections of the heading direction: two left heading corrections by $5^o, 10^o$, two right heading corrections by $-5^o, -10^o$, one straight ahead direction (*Figure* 4.15). In the working phase, in more than 30 test drives along 3 straight hallways and 2 turns, including 2 trained hallways and 1 trained turn, the robot successfully navigated, even when there were people walking along the hallways [CHE98].

4.5 CONCLUSIONS

In this chapter we have proposed an adaptive solution to the image window classification task. Two steps have been solved by the use of ANN-learning algorithms: the image compression step and the discriminant analysis step.

Firstly, a fast and reliable, adaptive learning algorithm for the principal component analysis (CRLS PCA) was proposed and tested on real images. This algorithm uses a normalized self–adaptive Hebbian rule and it constitutes an improvement on previous GHA, RLS and SAMHL methods for PCA. The architecture of the proposed neural network is a cascade one, i.e. the first processing unit drives the second one which in turn drives the third one and so on.

Secondly, an RLS based adaptation of the learning rate for PSA and associated quasi–parallel PCA methods was proposed and tested. A speed to quality trade–off was sought between the sequential and parallel working manner, depending on the number

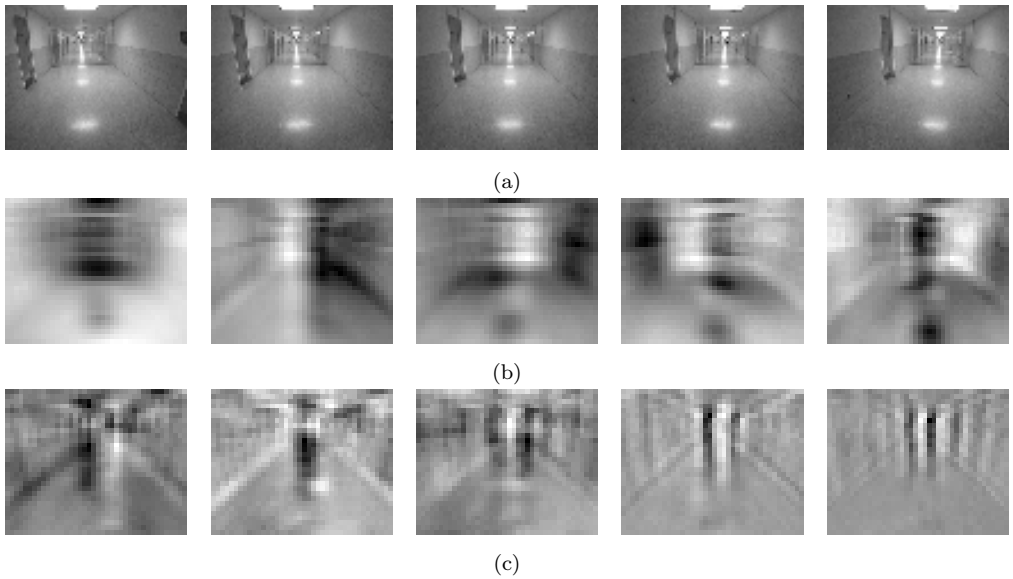


Figure 4.15 The images appearing during learning of robot heading correction commands: (a) some learning images of straight corridor hallways, (b) the "images" of the first learned PCs, (c) the images of the first five MDFs ([CHE98]).

of extracted PC/PS-s, while learning on natural image data. In our experiments the sequential CRLS method outperformed the parallel PSA and quasi–parallel ordered PCA methods for any number of PC/PS-s if the time of learning was in proportion to the number of PC/PS-s and a subset of at least 25% of the image data was used for the learning a single PC.

In the case of limited learning time (on-line learning required), it was found that the quasi–parallel PCA methods are a better choice than the PSA method. In the Brockett algorithm the vector $\boldsymbol{D}$ controls the *parallel behavior* of learning (i.e. if $\boldsymbol{D} = \boldsymbol{I}$ it simplifies the PSA algorithm). This matrix can always be set according to the number of requested PC-s in such a way that in a longer learning case BSA will perform with at least the same quality as the parallel PSA method does.

Hence, the leading performance of the CRLS method in comparison to other known PCA adaptive algorithms was documented. The convergence of this method was recently proved theoretically in [SKA99].

For the second step a learning vector quantifier method with supervised learning was proposed. It was proved that the method performs a discriminant analysis. This kind of adaptive technique is especially useful for classification in image spaces, as these spaces are usually very large. Although due to the initial compression step the final classification space can be reduced, but even then the size of matrices can cause computational problems for the conventional numeric approach to discriminant analysis.

5 VISUAL MOTION DETECTION AND ESTIMATION

The problem of visual motion detection or motion estimation can be solved in different ways, at various data abstraction levels. In the case of images from a stationary camera the iconic- and segmentation-level methods are quite robust. There are various visual motion detection and estimation methods available, working on these levels, i.e. on the iconic data level - single pixel or pixel block intensity-based approaches (e.g. difference image, adaptive difference image, block correlation methods, simulated annealing - global optimization) [AGG88, ENK88, FLE90, HEE88].

5.1 METHODS OF VISUAL MOTION DETECTION/ESTIMATION

Firstly, several conventional approaches to dense *visual motion* detection and *optical flow* estimation are implemented and tested. An adaptive ANN-based method of visual motion estimation is also proposed.

5.1.1 Typical methods

Visual motion detection

The simplest way of visual motion detection is to apply the so called *difference image* method, i.e. to obtain the pixel gray level differences between two consecutive images.

In the difference-image method simply the difference between intensities of corresponding pixels in two consecutive images are computed, and if the difference is larger than a pre-defined threshold a moving pixel is detected, i.e:

$$o(x,y) = \begin{cases} 1 & \text{, if } |I^{t+1}(x,y) - I^t(x,y)| > T, \\ 0 & \text{, if } |I^{t+1}(x,y) - I^t(x,y)| \leq T. \end{cases} \tag{5.1}$$

An *adaptive difference image* was proposed in [KAR90] for the detection of the stationary background in image sequences from a stationary camera. This method is specified in *Table* 5.1. Here, the motion mask is adapted from image to image. The data structures of our implementation of this method are denoted as follows:

Input: I - the source image.

Data: B - the motion image, dB - image of first derivatives in time.

Output: M - the motion mask image.

Optical flow estimation

The main class of pixel based methods for visual motion estimation is called *optical flow* detection [BAR94]. Among them we have implemented two approaches: the gradient-based optical flow [SCH86, NIE90] and the hierarchical block-based optical flow [KIR93].

We apply a gradient-based optical flow algorithm for estimation of motion vectors for each pixel, as given below in *Table* 5.2. For completeness of this algorithm let us denote the data structures as follows:

Input: $\boldsymbol{I}^k, \boldsymbol{I}^{k+1}$ – two consecutive input images.

Data: $\boldsymbol{f}^x, \boldsymbol{f}^y, \boldsymbol{f}^t$ – discrete derivative images; $\boldsymbol{u}, \boldsymbol{v}$ – discrete Laplacian images.

Output: of^x, of^y – two component images of the optical flow (visual motion).

The block-matching approaches to visual motion estimation are mainly used for image sequence compression, i.e. they allow a bandwidth reduction during the transmission of video signals. The main goal of block-matching in such an application is to predict

1. Initialize the system parameters.
   (a) Set the threshold for motion detection: $DIF = 20.0$.
   (b) Expectations of background dynamics ($\boldsymbol{A}$) and its measurability ($\boldsymbol{H}$):

$$\boldsymbol{A} = \begin{bmatrix} 1.0 & 0.7 \\ 0.0 & 0.7 \end{bmatrix}; \quad \boldsymbol{H} = \begin{bmatrix} 0.99 \\ 0.01 \end{bmatrix} \tag{5.2}$$

   (c) Adaptive properties of background extraction: $\alpha = 0.005$, $\beta = 0.05$.
2. Initialize the motion image and first derivative image:

$$I_{min} = min\{I(x,y)\} \quad , \quad I_{max} = max\{I(x,y)\}, \tag{5.3}$$

$$B(x,y) = (Imax - I_{min}) * 0.5 \quad , \quad dB(x,y) = 0, \forall x, y. \tag{5.4}$$

3. Initialize the output mask image:

$$M(x,y) = \begin{cases} 1 & \text{, if } (I(x,y) \geq B(x,y)) \\ 0 & \text{, if } (I(x,y) < B(x,y)) \end{cases} \tag{5.5}$$

4. The update rule:

$$B(x,y) = B(x,y) + g_{x,y} * dm_{x,y}, \tag{5.6}$$

$$dB(x,y) = dB(x,y) + g_{x,y} * dm_{x,y}, \tag{5.7}$$

$$\text{where: } dm_{x,y} = I(x,y) - [H[0] * B(x,y) + H[1] * dB(x,y)], \tag{5.8}$$

$$\text{and } g_{x,y} = \begin{cases} \beta & \text{, if } (M(x,y) = 0) \\ \alpha & \text{, if } (M(x,y) = 1) \end{cases} \tag{5.9}$$

5. Motion detection:

$$M(x,y) = \begin{cases} 0 & \text{, if } (|I(x,y) - B(x,y)| \leq DIF) \\ 1 & \text{, if } (|I(x,y) - B(x,y)| > DIF) \end{cases} \tag{5.10}$$

6. The prediction rule:

$$\begin{bmatrix} B(x,y) \\ dB(x,y) \end{bmatrix} = \boldsymbol{A} * \begin{bmatrix} B(x,y) \\ dB(x,y) \end{bmatrix}. \tag{5.11}$$

7. Read next image. Repeat from step 4.

Table 5.1 The adaptive difference image method of visual motion detection.

1. Initialize: $k = 0$, $a = 3$; $\forall_{i,j}$: $of^x_{(i,j)} = 0$, $of^y_{(i,j)} = 0$.
2. Read images $\boldsymbol{I}^k$ and $\boldsymbol{I}^{k+1}$.
3. Compute discrete derivatives:

$$\begin{array}{llll} f000 = \boldsymbol{I}^k_{[i,j]}; & f001 = \boldsymbol{I}^{k+1}_{[i,j]}; & f010 = \boldsymbol{I}^k_{[i,j+1]}; & f011 = \boldsymbol{I}^{k+1}_{[i,j+1]}; \\ f100 = \boldsymbol{I}^k_{[i+1,j]}; & f101 = \boldsymbol{I}^{k+1}_{[i+1,j]}; & f110 = \boldsymbol{I}^k_{[i+1,j+1]}; & f111 = \boldsymbol{I}^{k+1}_{[i+1,j+1]}. \end{array} \tag{5.12}$$

$$\begin{array}{rl} f^x_{[i,j]} = & [(f100 + f110 + f101 + f111) - (f000 + f010 + f001 + f011)] * \frac{1}{4}, \\ f^y_{[i,j]} = & [(f010 + f110 + f011 + f111) - (f000 + f100 + f001 + f101)] * \frac{1}{4}, \\ f^t_{[i,j]} = & [(f001 + f101 + f011 + f111) - (f000 + f100 + f010 + f110)] * \frac{1}{4}. \end{array} \tag{5.13}$$

4. Compute Laplacian images:

$$\begin{array}{rl} u_{[i,j]} = & (of^x_{[i-1,j]} + of^x_{[i+1,j]} + of^x_{[i,j-1]} + of^x_{[i,j+1]}) * \frac{1}{6} + \\ & +(of^x_{[i-1,j-1]} + of^x_{[i-1,j+1]} + of^x_{[i+1,j+1]} + of^x_{[i+1,j-1]}) * \frac{1}{12}; \\ v_{[i,j]} = & (of^y_{[i-1,j]} + of^y_{[i+1,j]} + of^y_{[i,j-1]} + of^y_{[i,j+1]}) * \frac{1}{6} + \\ & +(of^y_{[i-1,j-1]} + of^y_{[i-1,j+1]} + of^y_{[i+1,j+1]} + of^y_{[i+1,j-1]}) * \frac{1}{12}. \end{array} \tag{5.14}$$

5. Compute optical flow components:

$$of^x_{[i,j]} = \boldsymbol{u}_{[i,j]} - \boldsymbol{f}^x_{[i,j]} * val_{i,j} \quad , \quad of^y_{[i,j]} = \boldsymbol{v}_{[i,j]} - \boldsymbol{f}^y_{[i,j]} * val_{i,j}. \tag{5.15}$$

$$\text{where: } val_{i,j} = \frac{\boldsymbol{f}^x_{[i,j]} * \boldsymbol{u}_{[i,j]} + \boldsymbol{f}^y_{[i,j]} * \boldsymbol{v}_{[i,j]} + \boldsymbol{f}^t_{[i,j]}}{(\boldsymbol{f}^x_{[i,j]})^2 + (\boldsymbol{f}^y_{[i,j]})^2 + a}. \tag{5.16}$$

6. IF optical flow $\boldsymbol{of}$ is not stable THEN repeat from step 4.
7. Set $k = k + 1$ and repeat from step 2.

Table 5.2 The algorithm of gradient-based optical flow estimation.

the next image frame with the smallest error, which is slightly different than the main goal of visual motion estimation in image sequence analysis: to achieve a global optimality of the estimated visual motion field. This can be achieved by minimizing costs of block correspondence expressed via popular measures of difference between two pixel blocks: normalized cross-function, squared error or absolute error, called also *displaced frame difference* (DFD).

In order to resolve competitive correspondences of pixel blocks that appear during matching, a block-matching approach with continuity of motion in local neighborhoods was proposed [KIR93]. In this method the measure of correspondence quality considers both the similarity of blocks and the similarity of motion for neighbor blocks:

$$C = C_{DFD} + \alpha C_S, \tag{5.17}$$

where C_{DFD} are costs of similarity between two blocks, measured via the DFD measure, C_S are costs of motion field continuity and α is a weighting coefficient. The *displaced frame difference* (DFD) is defined as:

$$DFD_{i,j}(u, v) = \sum_{(x,y) \in B_{i,j}} |I^t(x, y) - I^{t+1}(x + u, y + v)|. \tag{5.18}$$

1. Set $\alpha = 0$ (no continuity). Compute the best motion field for initially large block size (e.g. 16×16), which is optimal with respect to C_{DFD}.
2. Set $\alpha \neq 0$ and minimize $C = C_{DFD} + \alpha C_S$ by a sequential of all block motion vectors $\boldsymbol{u}_{i,j}$.
3. IF the final block size is not reached
THEN divide each block into four sub-blocks AND repeat from step 2.

Table 5.3 The block-based optical flow.

The block similarity costs C_{DFD} are given as:

$$C_{DFD}(\boldsymbol{U}) = \sum_{i=1}^{n} \sum_{j=1}^{m} f_d[DFD_{i,j}(\boldsymbol{u})], \tag{5.19}$$

where for example ($f_d(d) = min\{d_{max}, |d|\}$). The continuity of motion field $\boldsymbol{U}$ is expressed as the similarity of motion field of current block and of its four neighbors:

$$c_s(\boldsymbol{u}_{i,j}) = \sum_{k=-1,0,1;l=-1,0,1} |\boldsymbol{u}_{i,j} - u_{i-k,j-l}|. \tag{5.20}$$

Hence the global costs of (non-)similarity are given as:

$$C_S = \sum_{i=1}^{n} \sum_{j=1}^{m} c_s(\boldsymbol{u}_{i,j}). \tag{5.21}$$

The considered method performs an iterative hierarchical minimization of the cost function (5.17) [KIR93]. A hierarchy of block sizes is assumed, where the process starts with the largest block size. The correspondences are computed on the basis of optimal matches between the first image block and the shifted blocks in the second image. Then each block is decomposed into a set of four smaller blocks, and the matching process is repeated at a more detailed level (see *Table* 5.3).

5.1.2 An ANN-based optical flow

A relaxation-like process in a feedback neural network is proposed that can compute optical flow for two consecutive images. Let a network with $(n \times m \times k^2)$ output neurons be given, where the neuron $o_{(x,y,D)}$ represents the hypothesis, that the pixel (x, y) in the first image corresponds to the pixel $(x + v, y + u)$ in the second image.

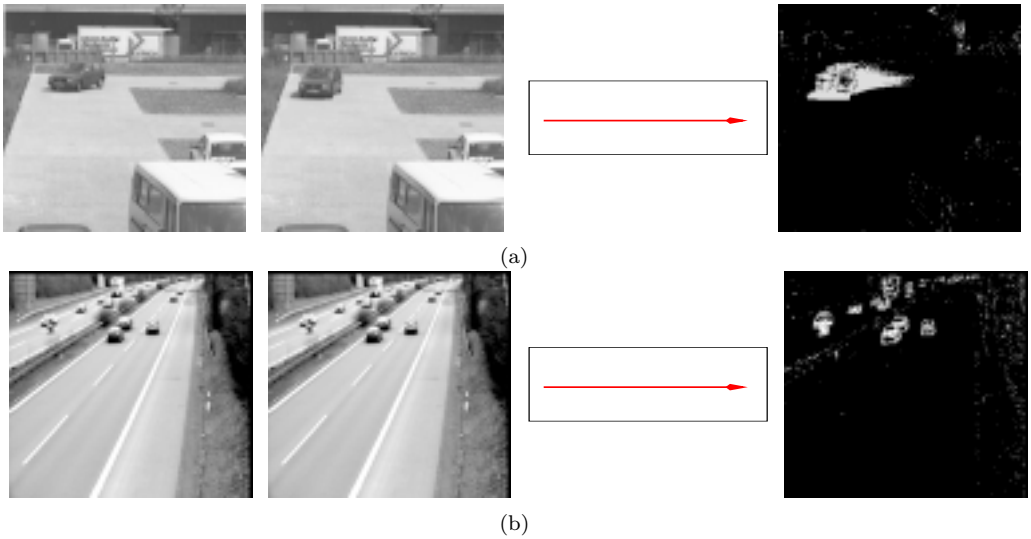
Let us denote by $D = fun(u, v)$ a linear mapping from motion vector (x, y), to an index D, where $\{v, u \in < -k/2 + 0.5, k/2 - 0.5 >\}$. The relaxation rule is given as:

$$y^{(t+1)}_{(x,y,D)} = \sigma[\sum_{S(x,y,D)} y^{(t)}_{(a,b,c)} - \eta \sum_{H(x,y,D)} y^{(t)}_{(a,b,c)} + y^{(0)}_{(x,y,D)}]. \tag{5.22}$$

Here η is a constant and σ is a threshold function. The local excitatory neighborhood $S(x, y, D)$ and the local inhibitory neighborhood $H(x, y, D)$ are defined as:

$$S(x, y, D) = \{(a, b, c)|(a = x \text{ or } a = x - v) \text{ and } (b = y - u \text{ or } b = y) \text{ and} c = D\}, \tag{5.23}$$

$$H(x, y, D) = \{(a, b, c)|(|(a, b) - (x, y)| \leq N) \text{ and } c = D\}. \tag{5.24}$$



(a)

(b)

Figure 5.1 Example of visual motion detection (stationary camera case): (a) adaptive difference image-based visual motion for nearly planar object motion, (b) difference image-based visual motion for mostly longitudinal object motion.

Initially the network contains the cross-correlation of the images

$$y^{(0)}_{(x,y,v)} = I^{(0)}_{(x,y)}\ I^{(1)}_{(x+v,y)}. \tag{5.25}$$

Thus the initial state represents the full set of possible pixel motions. During learning this set is steadily reduced until a stable state of the network is reached. The motion of some pixel $(x1, y1)$ is represented by one particular active output neuron from the set:

$$O_{x1,y1} = \{o_{(x1,y1,i)} | i \in < 0, k^2 - 1 >\}.$$

Only a single neuron in such set should be active at the end of the relaxation process.

5.1.3 Test examples

A comparison of the results of implemented visual motion detection methods for the stationary camera case are provided in *Figure* 5.1 - 5.5. The simple single pixel-based methods of difference- and adaptive difference-image perform well in both cases of object motion: if the object motion is approximately parallel to the image plane (planar motion) or mostly perpendicular to the image plane (longitudinal motion) (*Figure* 5.1). Let us simulate a more intensive longitudinal motion, by processing an image sequence taken from a moving camera. In such a case, in nearly the whole image a visual motion pointing towards a single or several focuses of expansion points should be detected/estimated (both stationary background and moving obstacles show a visual motion in the image). The example in *Figure* 5.2 demonstrates that an adaptive difference image stabilizes very well on the moving pixels, starting from some default motion image. Obviously the really moving objects cannot be distinguished from the stationary background.

(a) (b) (c)

Figure 5.2 The adaptation effect of adaptive difference image detection for longitudinal object motion case: (a) image sample, (b) the first (initial) motion mask, (c) motion mask after 25 images [KAS97].

(a) (b) (c)

Figure 5.3 The y–component of gradient-based optic flow after 25 images: (a) image sample, (b) pixel with positive-valued y-components, (c) pixel with negative-valued y-components [KAS97].

Similarly, the gradient-based optical flow also focuses quite well on the "synthetic" longitudinal movement of the background and objects (*Figure* 5.3). Moreover, this is a motion estimation approach, which nearly properly identifies that everything is moving toward the camera, i.e. along the negative Y-axis.

Both previous methods work in a local fashion and require only a few parameters. An opposite conclusion must be drawn for the block-based method. This is a highly global optimizing method and it seems that the parameters are very sensitive to the size of moving objects and to the type of motion. It can be seen in *Figure* 5.4 that the results of this method are not reliable in case of a strong longitudinal motion. Some differently moving objects belong to the same motion field in the block-based visual motion image.

In the adaptive (ANN relaxation based) method a relatively high quality-mapping between motion pixels and moving objects was achieved. The results are strongly dependent on the threshold function (σ) used during relaxation (*Figure* 5.5).

5.2 MOTION-BASED IMAGE SEGMENTATION

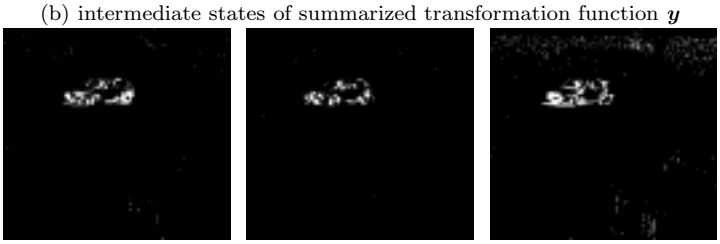
The pixel motion can be applied for image segmentation into moving and non-moving regions. This type of segmentation is used very often for outdoor scene analysis, as the projected objects are relatively small and detailed object structure is not detectable [BOB94, BOU93]. This approach is especially suited for object detection if

(a) (b) (c)

Figure 5.4 The y–component of hierarchical block-based optic flow after 25 images: (a) image sample, (b) blocks with positive-valued y-components, (c) blocks with negative-valued y-components [KAS97].

(a) 3 images are organized in pairs: 1-2, 2-3, 3-1

(b) intermediate states of summarized transformation function $\boldsymbol{y}$



(c) final state of summarized output activations $\boldsymbol{z}$

Figure 5.5 Example of ANN-based motion estimation. A 3-image sequence with one moving object is available. Three motion images are estimated for three pairs of input images: 1-2, 2-3 and 3-1.

there is a homogeneous background. Some related field is the compression of image sequences [DOM98], which usually exploits a short-term tracking of similar blocks.

5.2.1 The image segmentation module

Our contribution to image sequence segmentation can be summarized by the presentation of a (nearly application-independent) 2–D system module, which was developed by the author and described in detail in [KAS97]. Firstly, this module contains mostly

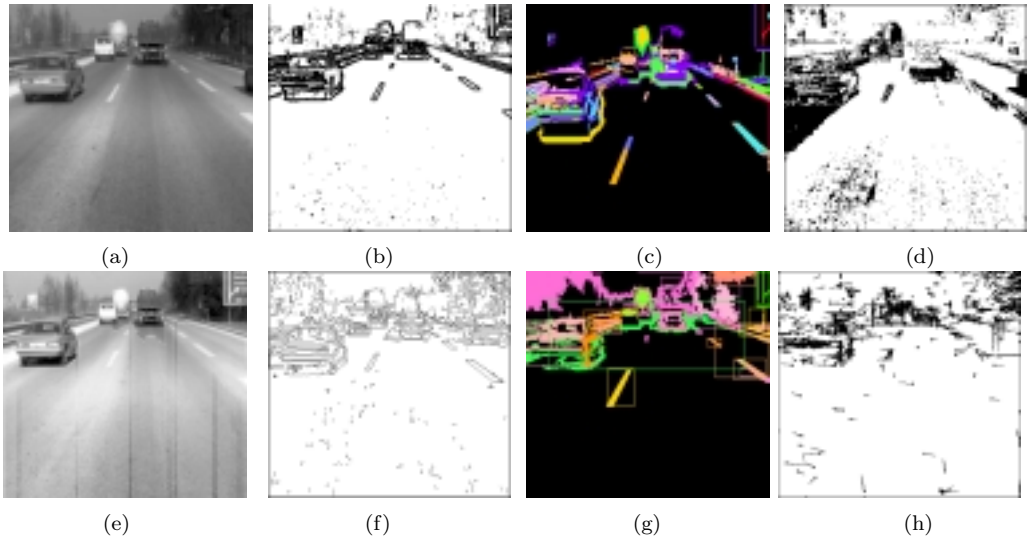


Figure 5.6 Result samples of analysis steps from the 2–D module: (a) normalized image, (b) edge image, (c) region image, (d) the motion mask image, (e) free region, (f) line segments, (g) segment groups (contours), (h) contour motion vectors [KAS97].

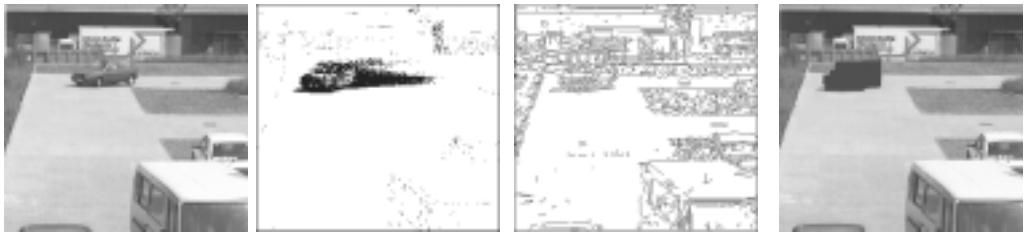
conventional methods for iconic processing, like: image normalization, edge- and region image detection, and motion mask estimation (*Figure* 5.6(a)-(d)). Secondly, the image segmentation steps follow: a "free" region detection (large region in front of the camera), line segment detection, segment grouping (linking lines to regions), represented by closed boundary contours, and contour motion estimation (*Figure* 5.6(e)-(h)).

5.2.2 Visual motion for segment classification

We have also considered the question if a visual motion detector can be helpful for segment classification. In a stationary camera case a positive answer can be given. Any visual motion detector can be applied to two or more consecutive images. For example, a so called *dynamic mask* is generated by the method of adaptive difference image (*Figure* 5.7). The segments are classified into "moving" or "stationary" depending on the amount of covered "moving" pixel.

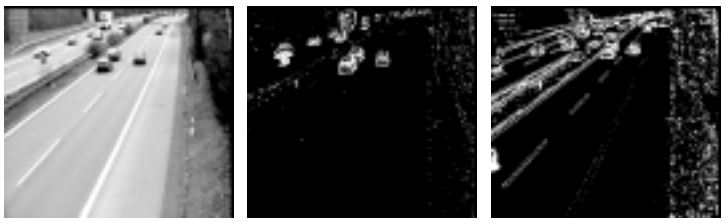
5.3 THE MOVING CAMERA CASE

In a moving camera case due to partially unknown ego–motion and unknown environment the simple visual motion detection method does not work properly (see *Figure* 5.8). This is even more evident in the case of a truly moving camera. From *Figure* 5.9 it is clear that the optic flow in image regions representing moving obstacles and the stationary surrounding area very frequently changes its sign, whereas the regions representing the road surface have approximately always the same number of upwards



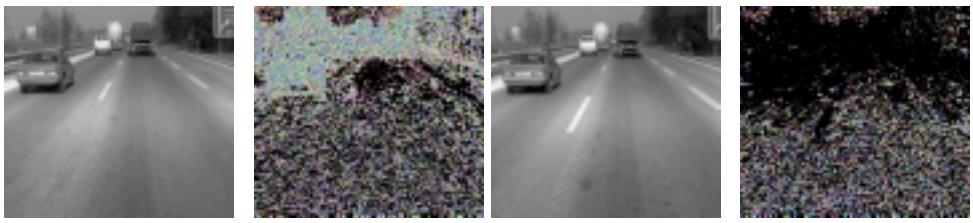
(a) (b) (c) (d)

Figure 5.7 Motion based segment classification in the stationary camera-case: (a) a normalized image, (b) the motion mask image, (c) segmented image, (d) detected moving object.



(a) (b) (c)

Figure 5.8 Examples of wrong visual motion estimation if the stationary camera is performing a small nodding movement: (a) one original image, (b) relatively good visual motion, (c) relatively wrong visual motion after the camera has moved.



(a) (b) (c) (d)

Figure 5.9 Wrong optical flow in moving camera case due to unknown nodding camera movement: (a),(c) original images - $t = 14$ and $t = 20$; (b),(d) pixels with negative y-component of gradient-based optic flow at $t = 14$ and $t = 20$ [KAS97].

and downwards moving pixels - thus these regions seems to be more stable in the optic flow images. But in any case the obtained optic flow does not properly separate the stationary background from moving objects.

5.3.1 Corrected visual motion under ego–motion

In order to overcome the influence of camera motion some model-based schemas can be considered. Let us distinguish two development stages: visual motion in the road plane (assuming known camera movement) (*Figure* 5.10) and ego-motion corrected visual motion (moving camera with unknown nodding movement) (*Figure* 5.11).

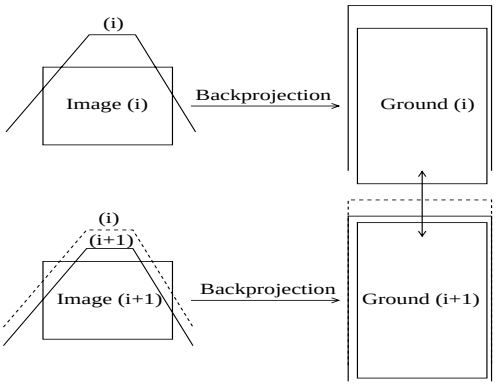


Figure 5.10 The principle of visual motion in the synthetic plane under known ego-motion.

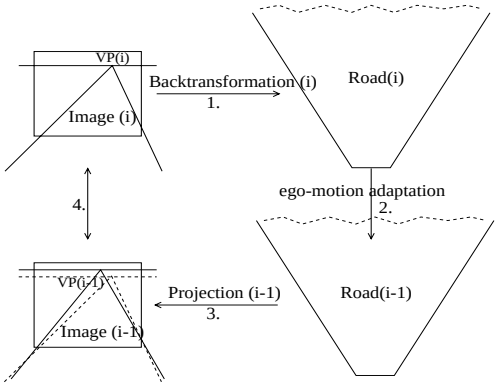


Figure 5.11 The principle of visual motion with unknown ego–motion correction.

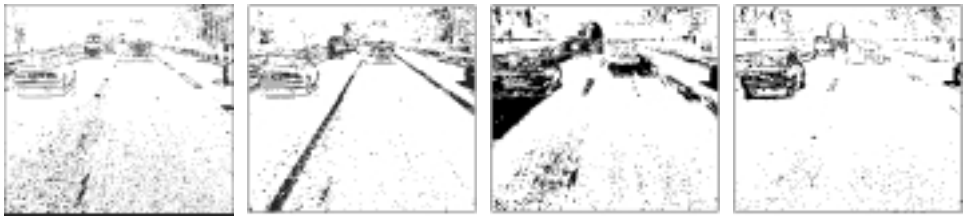
Let us first assume that the camera is moving with constant and known velocity and that no nodding movement occurs. This is approximately true for an indoor environment and for a relatively slow moving car or robot. The camera is turned toward the ground, i.e. the vanishing point is located in the image plane over the visible image area. Hence the vanishing point can be fully back–projected onto the assumed ground plane. As all points are assumed to be located on the ground, a transformation of the whole image into the "synthetic" road plane can be computed. Now a standard visual motion estimation can be applied for two consecutive ground images. This leads to motion vectors in the ground plane (*Figure* 5.10). The known ego-velocity of the vehicle provides a threshold for obstacle detection. If an obstacle violates the planarity assumption it corresponds to high displacement values in the estimated motion field.

In the case of unknown camera nodding movement we try to recognize the instantaneous camera position, i.e. assuming that we have an outer orientation point or we can measure the relative position against the ground and/or horizon. Then we can eliminate the influence of the movement of camera between two consecutive images. Finally we establish a correspondence between current image and a corrected previous image, from which the influence of estimated current ego–motion is eliminated (*Figure* 5.11).

The example results shown in *Figure* 5.12 verify that the performance of obtained visual motion is much higher in our two ego-motion correction approaches than in standard motion detection. In both correction methods the relative motion of (stationary) road stripes was nearly eliminated, whereas the motion of really moving objects (but of small relative image motion) was magnified (*Figure* 5.12(c,d)) (consider the characteristic disturbances in the backs of moving objects (c)). Unfortunately even small errors in camera projection conditions, ego-motion detection and environment modelling lead to significant errors of visual motion detection.

5.3.2 FOE-based segment classes

When the camera is performing a translating movement and the scene is planar or sufficiently distant, then still the 2-D motion may be sufficient for moving object detection. The dynamic *focus of expansion* point FOE (or epipole) is estimated, which is



(a) (b) (c) (d)

Figure 5.12 Results of visual motion estimation under ego-motion: (a) standard difference image, (b) standard adaptive difference image, (c) adaptive difference image in the ground–plane approach, (d) difference image with unknown ego–motion correction.

defined as the image plane point in which the motion vectors of (hypothetical) stationary background points vanish. An object moving differently from the camera induces its own individual FOE point (denoted by C_FOE) (like for contour segments in *Figure* 7.2). By comparing the general FOE point with image segment specific C_FOE points the image segment classification into "stationary' background and different "moving" objects is possible [IRA98].

In practice, under partially unknown camera ego-motion and with many moving scene objects it is very difficult to perform a reliable tracking of stationary single image segments. The author has experimentally verified, that the estimation of the focus of expansion point FOE is of much worse performance than the detection and estimation of the road's geometry-based vanishing point [KAS94a].

5.4 CONCLUSIONS

In the case of a stationary camera visual motion detection allows an easy image segmentation into moving objects and stationary background. A motion mask image can also be used for segment classification into stationary background or moving object.

In the case of a moving camera in an unknown environment these methods fail, because of the unknown transformation between the 3–D world and 2–D image. Even if the background of the scene is completely known and the influence of ego-motion on each pixel is computed, only small errors in ego-motion estimation lead to large errors in visual motion estimation.

In this chapter it was shown that reliable visual motion detection requires images from a stationary camera. Nevertheless, two correction approaches for motion detection under a moving camera were proposed, in order to extend the applicability of visual motion based image segmentation. Generally, in the case of an unknown moving camera a model–based approach to moving object detection should be preferred (see chapter 8).

6 AN ADAPTIVE OBJECT RECOGNITION SCHEME

The characteristic feature of single object tracking is the use of control techniques for a dynamic system with feedback [DIC88, WUE88]: at first a suitable initialization of object state is performed, then for each next image current object state is projected onto the image plane and it is matched against the new image features leading to the modification of the object state.

If the image sequence maps a many-object scene with a changing environment, competitive object hypotheses have to be generated, tracked and resolved. Besides multiple processes of single object tracking, the many-object *correspondence* task has to be solved [REI79, COX96]. As this problem is of exponential complexity, a trade–off between statistical property and heuristic pruning is sought for in most of the applications. Optimal search methods, like A^*-graph search [NIL82, PEA84], require a precise estimation of statistical properties of the application problem. On the other hand fast suboptimal methods can sometimes terminate with a suboptimal or a bad solution.

In this chapter an adaptive recognition scheme for modeling the symbolic analysis of infinite image sequences is developed, summarizing the author's previous work [KAS91, KAS97,KAS98]. The scheme combines two main paradigms: (A) single object tracking, (B) many-object description search.

6.1 ESTIMATION OF COMPETITIVE DYNAMIC OBJECTS

The second adaptive analysis scheme, developed in this work, is summarized in *Figure* 6.1. A (stationary or moving) object of given class (i.e. a segment, a token, a 2-D or 3-D object) which should be recognized in the image, is modeled further by a dynamic state vector in discrete time $\boldsymbol{s}(k)$. The image analysis procedure generates an object hypothesis, if there is sufficient evidence that an object of given class appears in the image data. Only some outer behavior of an object (represented by its system state) can be observed in the image, and this fact is summarized and modeled by the so called *measurement* vector $\boldsymbol{m}(k)$.

Due to errors in single image analysis, alternative object hypotheses may be generated. In order to find the proper hypotheses and to give a proper description of their dynamic behavior, a parallel tracking of many competitive *hypotheses*, combined with their state estimations $\boldsymbol{s}_i^*(k), i = 1, ..., n)$, is performed. The tracking process is supported by the *recursive estimation* mechanism, that allows the stabilization of the state estimations $s_i^*(k)$ due to an on–line evaluation of consecutive individual *measurements* (detections in the image) $\boldsymbol{m}_i(k)$.

A consistent subset of object hypotheses can be selected if a reliable judgement of each hypothesis is possible (*consistency search*). A hypothesis included in the selected consistent set can further change its status, i.e. it can be shifted from the *hypothesis*

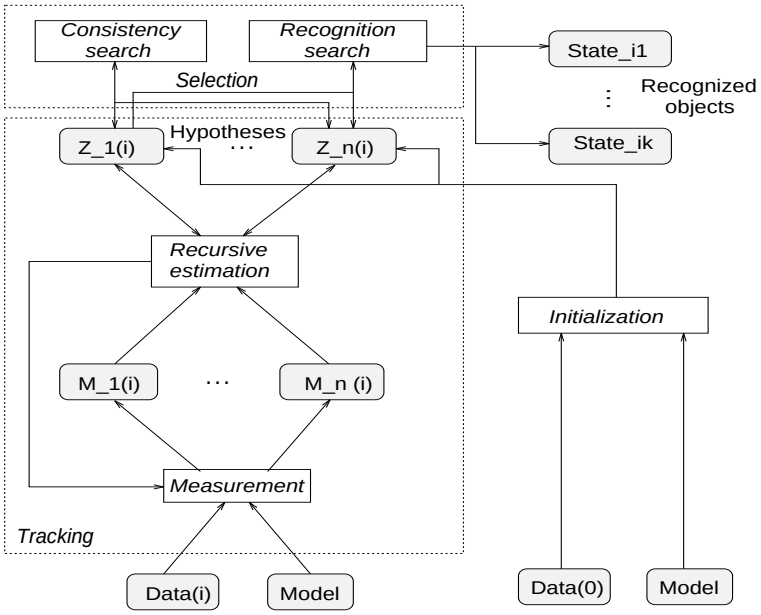


Figure 6.1 Overview of the adaptive object recognition scheme.

tracking state to the *recognized object* state, if its stability is of good performance and detailed object features are detected in the image.

6.1.1 The object model

Let $\boldsymbol{s}(k)$ be the *state (parameter)* vector at discrete time k and let $\boldsymbol{m}(k)$ be an associated observable *measurement* vector. The dynamic behavior of both vectors may be modeled by the following discrete–time dynamic system with free variables:

$$\boldsymbol{s}(k+1) = \boldsymbol{F}(k)\boldsymbol{s}(k), \tag{6.1}$$

$$\boldsymbol{m}(k) = \boldsymbol{H}(k)\boldsymbol{s}(k), \tag{6.2}$$

where: $\boldsymbol{F}(k)$ is called the state transition function (in matrix form) and $\boldsymbol{H}(k)$ is the state projection function (in matrix form). The goal of the *observation* task is to make a state estimation $\boldsymbol{s}^*(k)$ on the basis of measurements $\boldsymbol{m}(k)$ only. Let us denote the estimation error at time k by

$$\tilde{\boldsymbol{e}}(k) = \boldsymbol{s}(k) - \boldsymbol{s}^*(k) \tag{6.3}$$

and the predicted measurement at time k by

$$\boldsymbol{m}^+(k) = \boldsymbol{H}(k)\widehat{s}(k). \tag{6.4}$$

By adding a feedback component, i.e. the difference $(\boldsymbol{m}(k) - \boldsymbol{m}^+(k))$ between real and predicted measurements to the predicted state $\widehat{s}(k)$ one is able to minimize the estimation error:

$$\boldsymbol{s}^*(k) = \boldsymbol{s}^+(k) + \boldsymbol{K}(k)\Big\{\boldsymbol{m}(k) - \boldsymbol{H}(k)\widehat{s}(k)\Big\}, \tag{6.5}$$

where $\boldsymbol{K}(k)$ is some gain matrix. The associated dynamic behavior of the estimation error is:

$$\begin{aligned} \tilde{\boldsymbol{e}}(k+1) &= \boldsymbol{F}(k)\ \tilde{\boldsymbol{e}}(k) - \boldsymbol{K}(k+1)\Big\{\boldsymbol{H}(k+1)\boldsymbol{F}(k)\boldsymbol{s}(k) - \boldsymbol{H}(k+1)\boldsymbol{F}(k)\boldsymbol{s}^*(k)\Big\} \\ \tilde{\boldsymbol{e}}(k+1) &= \Big\{\boldsymbol{F}(k) - \boldsymbol{K}(k+1)\ \boldsymbol{H}(k+1)\ \boldsymbol{F}(k)\Big\}\ \tilde{\boldsymbol{e}}(k). \end{aligned} \tag{6.6}$$

Now it is evident that an appropriately designed $\boldsymbol{K}(k)$ matrix plays a crucial role in the minimization of the error $\tilde{\boldsymbol{e}}(k)$. However, it is not trivial to properly set all the $n \times m$ parameters $K_{ij}(k) \in \boldsymbol{K}(k)$ for all k-s without further assumptions about the nature of the measurement data. One way to do this is to model the measurement error by use of statistics.

A dynamic system with stochastic noise tries to model the uncertainty in system behavior by adding stochastic noise and measurement noise vectors to its equations. The mathematical model of a dynamic system with stochastic distortions in discrete time is:

$$\boldsymbol{s}(k+1) = \boldsymbol{F}(k)\ \boldsymbol{s}(k)\ +\ \boldsymbol{n}_s(k),\ k \geq 0, \tag{6.7}$$

$$\boldsymbol{m}(k) = \boldsymbol{H}(k)\ \boldsymbol{s}(k)\ +\ \boldsymbol{n}_m(k), \tag{6.8}$$

where $\boldsymbol{n}_s(k)$ denotes the system noise and $\boldsymbol{n}_m(k)$ means the measurement noise. The state estimation of such a system is the goal of the *filtering* task. Contrary to the previous *observation* task now we should be able to set the gain matrix $\boldsymbol{K}(k)$ optimal with respect to statistical features of the system. In the previous case we had no such optimality criterion.

6.1.2 Recursive estimation of a single object state

A consecutive solution of the filtering task is achieved by *recursive* methods, where the old estimate of $\boldsymbol{s}^*(k)$ is updated after new measurements $\boldsymbol{m}(k+1)$ are available. For this purpose the well known *Kalman filter* (KF) [BRA75] could be applied to filter the time discrete system with stochastic noise (6.7). In the Gauss–Markov model of random processes, the noise vectors $\boldsymbol{n}_s(k)$, $\boldsymbol{n}_m(k)$ are assumed to be white stochastic processes in discrete time, i.e. they have symmetrical, positive semi-definite covariance matrices and only the state noise vector $\boldsymbol{n}_s(k)$ has a non–zero expected value. It was proved elsewhere that the Kalman filter is an optimum filter for Gauss–Markov random processes, and for all other processes it is the optimum linear filter [BRA75]. The optimum in this context means that no other solution exists which gives in this case a smaller state estimation error than the linear Kalman filter does. For the estimation of a nonlinear dynamic system an *extended* Kalman Filter (EKF) can be used instead of the KF [WUE88, WUR88].

A natural criterion for judging the quality of state estimation $\boldsymbol{s}^*(k)$ is given by its covariance matrix $\boldsymbol{P}^*(k)$:

$$\boldsymbol{P}^*(k) = E\Big\{\Big[\boldsymbol{s}^*(k) - E\big\{\boldsymbol{s}^*(k)\big\}\Big]\Big[\boldsymbol{s}^*(k) - E\big\{\boldsymbol{s}^*(k)\big\}\Big]^T\Big\}, \tag{6.9}$$

where E is the expectation operator. Similarly, the covariance matrix $\boldsymbol{R}(k)$ of the measurement vector $\boldsymbol{m}(k)$ is:

$$\boldsymbol{R}(k) = E\Big\{ \Big[\boldsymbol{m}(k) - E\big\{\boldsymbol{m}(k)\big\}\Big] \Big[\boldsymbol{m}(k) - E\big\{\boldsymbol{m}(k)\big\}\Big]^T \Big\}. \tag{6.10}$$

During the measurement step it is important not only to obtain the current measurement vector $\boldsymbol{m}(t)$ itself but also to estimate the quality of individual measurements contained in $\boldsymbol{m}(k)$. The quality estimations lead to an estimation of the covariance matrix $\boldsymbol{R}^*(k)$ of the measurement vector. The quality of currently estimated state is the result of combining the previous estimation quality, expressed by $\boldsymbol{P}^*(k-1)$, with current measurement quality $\boldsymbol{R}^*(k)$ and current system error, expressed by the covariance matrix $\boldsymbol{Q}(k)$ of $\boldsymbol{n}_s(k)$.

This quality information is reflected by the component values of the gain matrix $\boldsymbol{K}(k)$.

Unfortunately, in some tasks of object tracking in image sequences the Kalman Filter can not be used. This is the case when a reliable quality judgement of detected image structures (or individual measurement error) cannot be provided by visual algorithms.

The measurement process means here the detection of appropriate image features, structures and object parameters, which all are combined to individual measurement vectors. The Kalman filter expects that two measurements of the same object in two different images are mutually independent, i.e. the judgement of first measurement data does not influence the judgement of the next one.

But in many object tracking systems this requirement is not satisfied, as a nearly optimal hypothesis is initialized first [KOL93] and it is tried repeatedly to detect this hypothesis in the next images. The error appears if the quality judgement of image measurements is directly controlled by the tracked object hypothesis. A proper solution (i.e. if individual measurement vectors of the same object are independent from each other) would be to apply general object class (model) expectations, instead of exploring the similarity of the measurement vector to the expected state projection of the tracked object.

6.1.3 Many-object description search

Let us assume, that our image sequences contain many moving objects. The ultimate goal of an object recognition system in such a case is to provide a many-object scene description. In our general model the process of searching for such a scene description is decomposed into two steps: (a) the *consistency search* - it selects the best consistent subset of hypotheses (if possible), and (b) the *recognition search* - it recognizes more detailed object features of selected object hypotheses.

Every object hypothesis corresponds to a dynamic state vector. During the analysis many (and possibly competitive) object hypotheses are tracked, as long as the proper consistent subset of hypotheses cannot be determined. The selection conditions are determined by two parameters: the minimum tracking time T_{min} of a hypothesis and the maximum variance threshold Var_{max} of combined quality judgement of a hypothesis. A selection test between two competitive hypotheses is performed only if their tracking times are both longer than T_{min} or if one of them is already in the recognition phase.

1. Initialization: Application–dependent computation of initial estimation $\boldsymbol{s}^*(k_0)$ and covariance matrix $\boldsymbol{P}^*(k_0)$ of estimation error. Go to step 6.
FOR every next image $k > k_0$
2. Detection of new measurement $\boldsymbol{m}(k)$. The covariance matrix of the system noise $\boldsymbol{Q}(k)$ is also available.
3. Estimation of current gain $\boldsymbol{K}$ - equation (6.11) or (6.17).
4. State modification (innovation): $\boldsymbol{s}^*(k) = \boldsymbol{s}^+(k) + \boldsymbol{K}(k)\{\boldsymbol{m}(k) - \boldsymbol{H}(k)\boldsymbol{s}^+(k)\}$
5. Modification of matrix $\boldsymbol{P}$ - equation (6.12) or (6.16).
6. Prediction of next state: $\boldsymbol{s}^+(k+1) = \boldsymbol{F}(k)\ \boldsymbol{s}^*(k)$
7. Prediction of next matrix $\boldsymbol{P}$: $\boldsymbol{P}^+(k+1) = \boldsymbol{F}(k)\boldsymbol{P}^*(k)\boldsymbol{F}^T(k) + \boldsymbol{Q}(k)$
8. $k \leftarrow k+1$

Figure 6.2 The recursive object state estimator.

A hypothesis has first a *tracking* status, but later this status can change to a *recognized* status. After the tracking time of given hypothesis exceeds T_{min} or its combined estimation variance falls below the Var_{max} threshold, the status of this hypothesis will change to a recognized status.

Let us further distinguish between two object specialization levels: *coarse* and *fine* object representation. Usually for a successful consistency test between two competitive object hypotheses a coarse representation of objects is sufficient, i.e. it is not required to recognize all details of these objects in the image. Only after an object hypothesis has changed to the recognized status, a more detailed recognition of this object, related to some fine model may be performed. Generally, more than one fine specialization level can be specified. This is an optional solution, which can be considered for objects with complex structure that are projected to large areas in the image, i.e. a sufficient amount of object details is visible and the available computational resources satisfy the real–time processing constraint.

6.2 RECURSIVE ESTIMATION OF SINGLE OBJECT

6.2.1 The developed recursive estimator

The developed recursive estimator is given in (*Figure* 6.2). It is similar to an extended Kalman filter (EKF) except the main estimation steps 3 and 5. These two steps may be alternatively performed in two ways:

1. the original EKF filter can be used or
2. an RLS-filter can be used, which minimizes the tracking error ($\boldsymbol{e}_T(k) = \boldsymbol{m}(k) - \boldsymbol{H}(k)\boldsymbol{s}^+(k)$) (and which solves a similar problem to an adaptive learning rate computation in ANN-learning of non-stationary signals).

From the state modification rule it is evident that a crucial role in the minimization of the estimation error ($\boldsymbol{s}(k) - \boldsymbol{s}^*(k)$) is played by an appropriately designed $\boldsymbol{K}(t)$ matrix. Originally, in the EKF the gain matrix depends on the estimation error covariance

matrix and on noise covariances only. Hence, the Kalman gain in EKF is estimated as:

$$\boldsymbol{K}(k) = \boldsymbol{P}^{+}(k)\boldsymbol{H}^{T}(k)\left\{\boldsymbol{H}(k)\boldsymbol{P}^{+}(k)\boldsymbol{H}^{T}(k) + \boldsymbol{R}(k)\right\}^{-1}, \tag{6.11}$$

where $\boldsymbol{R}(k)$ is the covariance matrix of the measurement. The original modification equation of the error covariance matrix $\boldsymbol{P}$ in EKF is given as:

$$\boldsymbol{P}^{*}(k) = \boldsymbol{P}^{+}(k) - \boldsymbol{K}(k)\boldsymbol{H}(k)\boldsymbol{P}^{+}(k). \tag{6.12}$$

Hence, both steps in EKF are computed independently from the current *tracking error* ($\boldsymbol{e}_T(k) = \boldsymbol{m}(k) - \boldsymbol{H}(k)\boldsymbol{s}^{+}(k)$). This requires the existence of a proper judgement scheme for detected ("measured") image data in a given application, i.e. where one can reliably judge the correspondence between object class elements and image segments. But in most of the applications one is not able to compute a judgement about how well the segment data fits to some object class. If such a scheme is not available, all the $n \times m$ parameters $K_{ij}(t) \in \boldsymbol{K}(t)$ for all t should be set by default - this will usually result in large object tracking errors. In such case, the desired solution requires a direct inclusion of the tracking error in the gain estimation equation.

The alternative recursive filter tries to directly minimize the current tracking error $\boldsymbol{e}_T(k)$ - two new rules for the computation of the gain and state covariance matrices are proposed that are similar to rules for self–adaptation of learning rates in neural network learning [AMA67]. Let us consider a neural network described in matrix form as: $\boldsymbol{y}(t) = \boldsymbol{W}(t)\boldsymbol{x}(t)$, where $\boldsymbol{y}(t)$ is the output vector, $\boldsymbol{W} = [w_{ij}] \in R^{n\times n}$ is the synaptic weight matrix, $\boldsymbol{x}(t)$ is the input vector. Recently it was proposed [CIK96b, MUR97], that each synaptic weight w_{ij} has its own (local) learning rate $\eta_{ij}(t)$ and that this rate is adjusted during the learning process according to a set of differential equations. A learning rule performs the minimization of expected error, i.e.

$$\frac{\partial w_{ij}(t)}{\partial t} = -\eta_{ij} J_{ij}(\boldsymbol{W}, \boldsymbol{y}, \boldsymbol{x}), \tag{6.13}$$

where $\eta_{ij}(t) > 0$ is a local adaptive learning rate, and J_{ij} is some loss function. The rules for the adaptive learning rate can be derived as:

$$e_{ij}(t+1) = (1-\delta)e_{ij}(t) + \delta J_{ij}(t), \tag{6.14}$$

$$\eta_{ij}(t+1) = (1 - \eta_{ij}(t)\delta_1)\eta_{ij}(t) + \alpha\eta_{ij}(t)|e_{ij}(t)|, \tag{6.15}$$

where δ, δ_1 and α are some positive scalars. δ and δ_1 control the stabilization speed of the expected error $\boldsymbol{e}(t)$ and the learning rate matrix $\boldsymbol{\eta}(t)$. α controls the influence of the expected error in one iteration, which is normalized by the maximum expected error.

Now let us come back to the dynamic system model and let us assume that only default values of measurement vector judgments can be provided. The quality of predicted state vector $\boldsymbol{s}^{+}(k)$ will be expressed by the tracking error ($\boldsymbol{e}(k)_T = \boldsymbol{m}(k) - \boldsymbol{H}(k)\boldsymbol{s}^{+}(k)$), via its associated covariance matrix ($\boldsymbol{R}_e(k) = \mathrm{E}\{\boldsymbol{e}(k)\boldsymbol{e}^{T}(k)\}$). To assure a non-zero minimum error threshold the default matrix $\boldsymbol{R}(t)$, corresponding to measurement noise, is

also added. After a derivation similar to the derivation of the adaptive learning rate rules in eq. (6.14), the alternative estimation of matrix $\boldsymbol{P}^*(k)$ is given as:

$$\boldsymbol{P}^*(k) = (1-\delta)\boldsymbol{P}^+(k) + \delta\boldsymbol{P}^+(k)\boldsymbol{H}^T(k)(\boldsymbol{R}_e(k) + \boldsymbol{R}(k))(\boldsymbol{H}(k)\boldsymbol{P}^+(k)\boldsymbol{H}^T(k))^{-1} \quad (6.16)$$

and the alternative estimation of a single gain matrix element is:

$$K_{ij}(k+1) \quad = (1 - K_{ij}(k)\delta_1)K_{ij}(k) + \alpha K_{ij}(k)\textstyle\sum_l P_{il}(k)H_{ij}(k). \quad (6.17)$$

6.2.2 The measurement step

During the measurement step at first it is important to detect and to select the current measurement vector $\boldsymbol{m}(k)$ itself. It is clear that various image segments may have different certainty factors associated with them. The criteria for selection of the best one should be as much independent as possible from the currently tracked object hypotheses. The search area in the image for new measurement data may be restricted to the area explained by the given hypothesis. But other detection criteria should follow model specific assumptions and should not be related to the current hypothesis. For example, many competitive measurements may be performed, according to different model views, that are possible in the image. Among the competitive measurements one is determined to be the best one, which best satisfies the model constraints. In this way the statistical independence of consecutive measurements for a given hypothesis is satisfied in practice.

The use of an EKF-like recursive estimator (with equations 6.11 and 6.12) can only be allowed in applications which include reliable measurement judgement schemas. For example, we shall propose a short-time tracking of corresponding measurements in some applications (in chapter 7 and 8) in order to estimate the errors of individual measurements, i.e. by means of their differences from the expected average values. Without such specific measurement schemas, one should work with default measurement errors and after some tracking time every covariance matrix $\boldsymbol{P}$ should be stable filled with low covariance values, giving no indication about the real quality of its associated hypothesis.

If the tracking-error based estimator (second recursive estimator case) is used, according to equations 6.16 and 6.17, the measurement error can be modeled by a constant matrix $\boldsymbol{R}$. Obviously, for the task of optimal segment selection itself, different certainty and quality judgements can be used. The object tracking error now directly influences the state estimation. Hence the estimator is able to follow highly non–stationary signals, i.e. to track abrupt changes. Additionally, the state covariance matrix expresses the quality of the associated hypothesis, as required by the hypothesis selection procedure.

6.3 MANY-OBJECT DESCRIPTION SEARCH

In this section a general solution for the tasks of consistency- and recognition search is developed. Both problems will be expressed in terms of a *model-to-image matching* scheme, which again will be implemented in the form of a graph search-based control.

In a simple object model there is only one representation level, i.e. an object has no explicit structure and it corresponds directly to a group of image segments. In such a case, the consistency search process deals with a set of homogeneous hypotheses, i.e. no decomposition of the search process into sub-search processes between competitive sub-objects is possible. A trivial case appears if additionally, there exist no fine object models, i.e. no specialization relations between objects are specified in the model. Consequently, the process of recognition search would not be necessary.

Let us further focus on a general object model case, in which two hierarchies of perpendicularly and horizontally structured (heterogeneous) object classes exist. These hierarchies are expressed by *is-part-of* and *is-specialization-of* dependency relations between objects. Such a general object model is represented by a labeled tree or graph structure, which contains different sub-objects as its nodes and dependency relations between objects - as its edges. Hence, both consistency- and recognition-search processes can be seen as a *matching* process between the model graph and current image segments.

6.3.1 Graph search-based control

In this subsection, a general approach to model-to-image matching in terms of graph search is developed [NIL82,PEA84,NIB90]. Obviously, different approaches to this problem, like a relaxation-like approach by means of a truth maintenance system [DEK86], could be developed as well. The advantage of our approach is, that a clearly application-independent framework can be specified in terms of graph search, which requires only few control elements to be supplied by the application (user).

Let us explain the approach with the help of an illustrative example (*Figure* 6.3). At first, a complex object model in terms of a hierarchical graph (*Figure* 6.3(a)) is given (two objects E, F represent the most abstract objects, whereas the low-level objects, like D, correspond to segmentation data). In a real model of a vehicle the concepts can represent for example following object classes: E - vehicle, F - car, A - front view, B - lower front region, C - cabin region, D - rectangle segment.

After the image segmentation process many data items exist in the data base (*Figure* 6.3(b)) (for example, three of them are instances of the low-level object class D - $I_1(D), I_2(D), I_3(D)$). The goal of the model-to-image matching can be expressed as follows: find the best (according to some criteria specified by the user) consistent subset of instances of (most specialized) top-level objects, that include current low-level data items (instances of model concepts).

The overall goal will be achieved in a step-by-step manner, through the generation of data items (so called *modified concepts* Q or *instances* I of model concepts) by applying several *inference rules* to already existing data and model concepts (*Figure* 6.3(c)). At least the following five inference rules should exist: (1) selection of current goal concept (for example E), (2) top-down modification (for example, generating the modified concepts $Q(B)$ and $Q(C)$ based on previously obtained $Q(A)$), (3) bottom-up modification (for example, obtaining the modified concept $Q_1(E)$ on base of instance $I(A)$), (4) bottom-up instantiation (for example from $I_1(D)$ the instance $I(B)$ can be generated), (5) derivation of a more specialized modified object (for example, after $I(E)$ is available a modified concept $Q(F)$ can be generated). Let us note, that in each inference

step many alternative results (instances of given concepts or modified concepts) may be generated.

The history of performed inference steps is stored in the form of a decision tree (*Figure* 6.3(d)). With every new inference step, the decision tree is expanded by adding tree nodes, corresponding to the alternative results of such a step. The shape of a particular decision tree and the optimal path, corresponding to the best model-to-image match, cannot be foreseen (this is non-deterministic control). Along each path of the decision tree, its own (local) net of inference results (data items) can be associated. The generation of n competitive data items of one model concept in a single inference step leads to the creation of n successor nodes of the current node in the decision tree.

For example, the search path specified in (*Figure* 6.3(d)) represents the following history of inference steps with unique, noncompetitive inference results: 1) selection of goal node A (a modified concept of A is added), 2) top-down expansion of A (including the modified concepts $Q(B)$ and $Q(C)$), 3) top-down expansion of B (a modified concept Q(D) is generated), 4) instantiation of D (among others the instance $I_1(D)$ is generated and added to the data net of the considered path), 5) instantiation of B (among others the instance $I(B)$ is generated and added), 6) top-down expansion of C (the modified concept $Q_2(C)$ is added), 7) instantiation of D (the instance $I_2(D)$ is added). Similarly, with each path in such tree a consistent inference net (representing a partial solution of the matching problem) is associated.

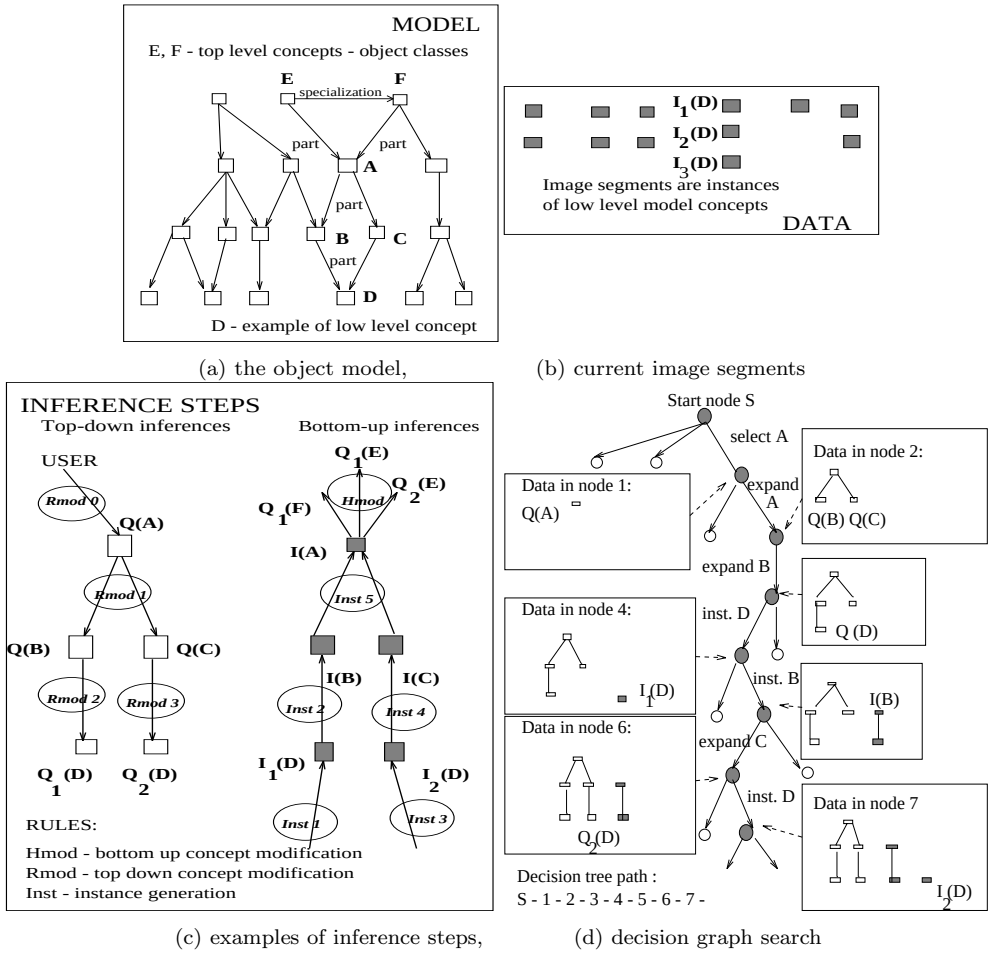
6.3.2 Model-to-image sequence matching

In the context of image sequence analysis, a solution path in the decision graph need not be found immediately during the analysis of the first image. The final results (data items, decision graph) for a previous image constitute the starting point for the next-image analysis. An additional inference step (called *refreshing*) is necessary, that performs the next state estimation of an already tracked object. Different strategies for how to explore the previous decision tree and how to refresh the previous data items (objects) are possible. One solution to this consecutive control will be given in chapter 9.

The consecutive control should satisfy two (to some extent contradictory) conditions: (1) it should avoid an exhaustive analysis flow by providing a sufficient *selectivity* of analysis, and (2) it should provide a *stable* tracking of concept instances, i.e. the consistency decisions made for an earlier image should not be changed during next image analysis.

If the decision about competitive instances may be postponed to a later time (if necessary after several images) a parallel tracking of competitive hypotheses will be possible. This is a desired solution, as a selection decision that is made after a longer tracking phase is usually more proper, than a decision made immediately after first appearance of an object hypothesis.

On the other hand, the selectivity of analysis is an important requirement for the analysis of every image from the sequences. Here, the use of an optimal graph search-based control naturally reduces the computational effort, as it provides a small number of consistent solutions.



(a) the object model, (b) current image segments

(c) examples of inference steps, (d) decision graph search

Figure 6.3 The model-to-image matching process in terms of graph search.

6.3.3 A^*–graph search

The decision graph is constructed incrementally during the model-to-image matching process, and the graph is expanded as long as the ultimate goal is not reached. If a graph node judgement function is available, some strategies for the best expansion of this graph can be proposed. An "intelligent" strategy combined with a robust judgement function will prohibit the expansion of all possible alternative solutions. This demand is equivalent to the problem of finding the best path in a finite graph, from the root node to one of its terminal nodes. After Nilsson [NIL82] let us denote by the A-search the *best node* search algorithm in finite graphs. The A-algorithm employs an additive cost function:

$$f(n) = g(n) + h(n), \tag{6.18}$$

A , A^*:
(1) put s in OPEN; set $g(s) \leftarrow 0, f(s) \leftarrow 0$
(2) IF OPEN is nonempty
THEN STOP with error message at output
(3) remove from OPEN the node n with the smallest f–value and put n in CLOSED (equalities are resolved according to the better terminal node)
(4) IF n is a terminal node
THEN STOP with current $g(n)$-value and solution path from s to n at output
(5) expand the node n (if no successor node is found then $\longrightarrow$ step 2)
(6) FOR every successor n_i of n
$g_i \leftarrow g(n) + c(n, n_i)$
(7) FOR every successor n_i of n
(7a) IF $n_i \notin$ (OPEN $\cup$ CLOSED)
THEN put n_i in OPEN, $g(n_i) \leftarrow g_i$, $f(n_i) \leftarrow g_i + h(n_i)$
(7b) IF $n_i \in$ (OPEN $\cup$ CLOSED) $\wedge$ $(g(n_i) > g_i)$
THEN $g(n_i) \leftarrow g_i$, $f(n_i) \leftarrow g_i + h(n_i)$; remove the former path $(s - n_i)$
IF $n_i \in$ CLOSED
THEN put n_i in OPEN
(8) $\longrightarrow$ step 2

Figure 6.4 The A (A^*) algorithm for optimal graph search.

where $g(n)$ means the current traversal costs from start node s to current node n, and $h(n)$ are (expected) remaining costs of the solution path from node n to some terminal node (heuristic part). Consider following classes of the heuristic $h(n)$ for a graph $\mathcal{G}$:

- a heuristic $h(n)$ is *admissible* if $h(n) \leq h^*(n), \forall n \in \mathcal{G}$;
- a heuristic $h(n)$ is *consistent* if $h(n) \leq c(n, n') + h(n')$, $\forall$paths $(n, n') \in \mathcal{G} \times \mathcal{G}$, where $c(n, n')$ are costs of the path from n to n';
- a heuristic $h(n)$ is *monotonic* if $h(n) \leq c(n, n') + h(n'), \forall$ edges $(n, n') \in \mathcal{G} \times \mathcal{G}$.

Let us observe, that *monotonic* and *consistent* heuristics are equivalent terms. The A^*–algorithm is such an A–algorithm, for which the condition

$$h(n) \leq h^*(n) \text{ for each node } n \in \mathcal{G}, \tag{6.19}$$

is satisfied, i.e. h is an admissible heuristic [NIL82]. Hence, there is no algorithmic difference between A and A^*, as the above requirement restricts the processed data only, but it is not related to the computation procedure itself. In both cases the processing pattern is identical (*Figure* 6.4). A^* is an optimal graph search algorithm but only under *consistent* heuristics [NIL82, PEA84].

In practice a graph search algorithm has sometimes to deal with "worse" heuristics. If the heuristic is not admissible then the A–search proceeds as follows: if two paths lead to the same node then the better path is selected on the basis of function g instead of f (step 7.b). The function f determines the best node from the OPEN set (step 3).

A modification of A^* is possible, in which, after a repeated selection of some already expanded node n_i (step 7.b), the new evaluation is immediately propagated without moving this node from CLOSED to OPEN [NIL82]. For a tree search this modification

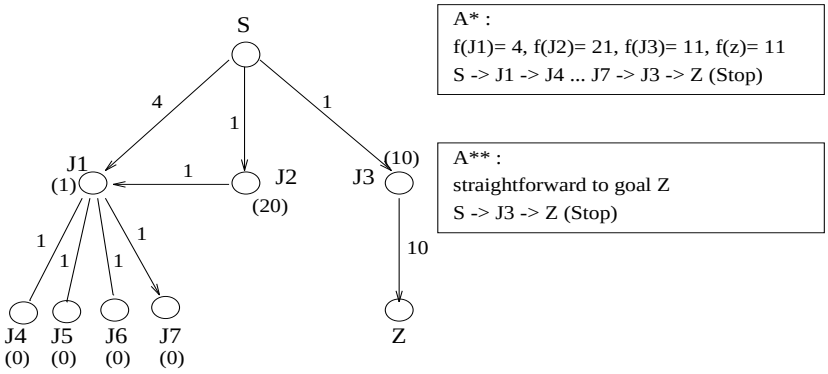


Figure 6.5 Search under admissible and pathologic heuristics.

has no relevance, as a node n_i is always a new one (step 7.b never happens). This step is never performed in a graph search with consistent heuristic, as the relation $g(n_i) \leq g_i$ is always true.

6.4 MODIFICATIONS OF A^*-GRAPH SEARCH

Let us review some generalizations of the A-algorithm, with better behavior than A in the case of graphs with inconsistent or non-admissible heuristics, e.g. A^{**} [DEC88], B [MAR77], C [BAG83] and D [MAH88]. The complexity of search is expressed by $\mathcal{N}$, the number of graph nodes with lower costs than the costs of the optimal solution node.

6.4.1 Pathologic heuristics

Let us first assume the case of admissible heuristic. We define the specific *non-pathologic* problem as [DEC88]: there exists at least one optimal solution path P in graph $\mathcal{G}$ for which the h is not fully informed, i.e. $\forall n \in P : h(n) < h^*(n)$ (except for the goal node). If this condition does not hold a pathologic graph search problem appears.

Let an algorithm A^{**} be defined, which differs from A^* only by the use of the following judgement function f:

$$f'_{P(s-n)}(n) = \max\{f(n') = g_{P(s-n)}(n') + h(n') \quad | \quad n' \in P_{(s-n)}\}. \tag{6.20}$$

In contrast to A^*, for the selection of some node n, the algorithm A^{**} uses the ordinary $(g + h)$ value not only of the current node n, but the maximum judgement value of n and of all previous nodes included in the path from start node s to the current node n. This maximum $f'_P(n)$ replaces the previous $f_P(n)$ function value of the A^*-algorithm. It can be proved, that in the case of pathologic problems A^{**} dominates over A^*, i.e. it expands no more nodes [DEC88]. Let us demonstrate this fact by two examples.

Example 6.1 - In *Figure* 6.5, A^{**} does not expand the node J_1 further, which would be expanded by A^* (the edge costs represent $c(v_i, v_j) = g(v_j) - g(v_i)$ and the node costs

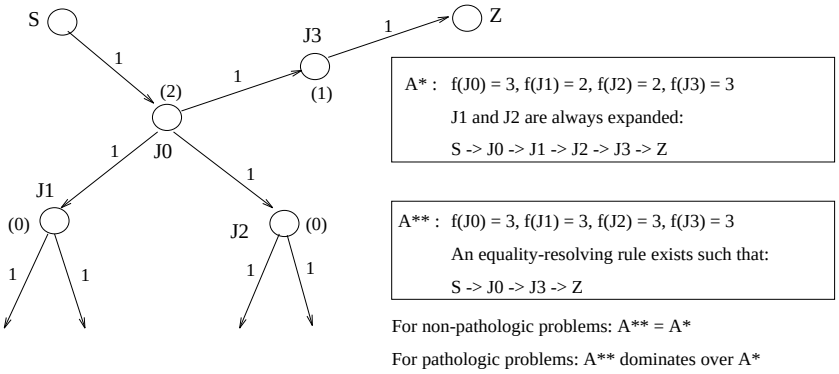


Figure 6.6 Equality-resolving rules under pathologic heuristics.

Algorithm B is similar to A^* except for the new steps 1 and 3, which are as follows:
B1: Put s in OPEN, set $g(s) \leftarrow 0, f(s) \leftarrow 0, F \leftarrow 0$.
B3: IF nodes exist in OPEN with $f < F$, THEN select the node n with the smallest g-value among them, ELSE select a node n from OPEN with the smallest f-value and set $F \leftarrow f(n)$.
(Resolve equalities with respect to the goal.) Move n from OPEN to CLOSED.

Table 6.1 B-graph search algorithm.

in parentheses are $h(v_i)$). If A^{**} has expanded the node J_2 then J_1 already has the costs $f(J_1) = 21$ due to the edge, linking it with J_2.

At the same time A^{**} does not expand any other node, which is not expanded by A^*. This is due to the fact, that for each equality-resolving rule in A^* there exists a "better" equality-resolving rule in A^{**}, i.e. where A^{**} expands a true subset of nodes expanded by A^*.

Example 6.2 - In *Figure* 6.6 the nodes n_1 and n_2 would be expanded by A^* under every equality-resolving rule, but A^{**} would be able to expand the path P_{s-z} only under some appropriate rule.

6.4.2 Non-monotonic heuristics

Let the node set V of graph G be given and let $\mathcal{N} = |V|$. For a consistent h up to all nodes from V will be expanded by A^*-search (and only once), i.e. there is a linear search complexity with respect to $\mathcal{N}$: $O(\mathcal{N})$. For a non-monotonic (inconsistent) h every node could be repeatedly expanded (if G is not a tree). It can be shown, that the worst-case complexity of A^* under admissible heuristics ($h(n) \leq h^*(n), \forall n$) is equal to $O(2^{\mathcal{N}})$ [MAR77]. A modification of the A^* algorithm was proposed, called B-search [MAR77] (*Table* 6.1).

The behavior of B is identical with A^* under consistent heuristics, as in such a case no nodes exist with $f < F$. Under inconsistent heuristics B is generally better than A^*. Another modification of A^*, called algorithm C was proposed in [BAG83] (*Table* 6.2).

Algorithm C is similar to A^* except for the new steps 1 and 3, which are as follows:
C1: Put s in OPEN, set $g(s) \leftarrow 0, f(s) \leftarrow 0, F \leftarrow 0$.
C3: IF nodes exist in OPEN with $f \leq F$, THEN select the node n with smallest g-value among them, ELSE look at nodes in OPEN with the smallest f-value and select from them the node n with the smallest g-value and set $F \leftarrow f(n)$.
(Resolve equalities with respect to the goal). Move n from OPEN to CLOSED.

Table 6.2 C-graph search algorithm.

B and C are of similar worst-case complexity $O(\mathcal{N}^2)$, but B can not provide a better solution (with lower costs) than C does [BAG83]. Let us assume that h-values are non-negative values. Obviously g-values are always non-negative values. Let C^* be the maximum cost of all expanded graph nodes: $g(n) + h(n) \leq C^*$. For all solution paths $P_1, P_2, P_3...$ in a graph their nodes with the largest costs M_i are given - one node per path ($Mi = \max_{n \in P_i}\{c(P_i, n) + h(n)\}$), where $c(P_i, n)$ are the costs of path P_i from s to n. By Q we denote the smallest cost among these maximum node costs: $Q = \min_{i \geq 1}\{M_i\}$.

Under admissible heuristics h it holds that: $Q = C^* = f(s)$, but under a non-admissible one it can happen that $Q = C^* = f(s)$. For $(Q = f(s))$ all algorithms A, B and C will find a solution with minimum costs. If condition $(Q > f(s))$ holds then such a solution cannot always be found.

Example 6.3- Let a graph G_8 be given in *Figure* 6.7(a) with admissible heuristics. Algorithm A makes 18 node expansions but B and C expand only 13 nodes each:
A : $s, n_6, n_7, n_1, n_2, n_1, n_3, n_1, n_2, n_1, n_4, n_1, n_2, n_1, n_3, n_1, n_2, n_1$;
B, C: $s, n_6, n_7, n_1, n_2, n_1, n_3, n_2, n_1, n_4, n_3, n_2, n_1$.
Obviously there are graphs for which A has exponential complexity, whereas the complexity of B, C remains polynomial.

In *Figure* 6.7(b) a graph G_9 with admissible heuristics is given, for which all three algorithms expand the same 8 nodes each: $s, n_7, n_8, n_5, n_4, n_3, n_2, n_1$. We conclude, that for some graphs all algorithms A, B or C require a search of linear complexity.

In *Figure* 6.7(c) non-admissible heuristics appear. For the given graph the algorithms B and C return a solution with costs equal to 3. Algorithm A can return the value 4 assuming an appropriate equality-resolving rule. Hence, B and C always return the solution with minimal costs, whereas the solution delivered by A may not be optimal for non-admissible heuristics.

A modification of algorithm C, called D-algorithm was proposed, in order to reduce the number of repeated node expansions [MAH88] (*Table* 6.3). Algorithm D will not return a better solution than algorithm C does, but in the case of repeated expansions of search nodes, the set of nodes $V(D)$ expanded by D is smaller than $V(C)$. The comparison of algorithms is summarized in *Table* 6.4.

6.4.3 Sub-optimal approaches

In [KAS91] the author studies the complexity of the optimal search for a search tree with a single goal node [HUY80, PEA83] or with more than one goal node [BAG88]. A very precise heuristic is required, i.e. the error may increase not faster than a logarithmical function of the path length N, in order to avoid an exponential growth of the

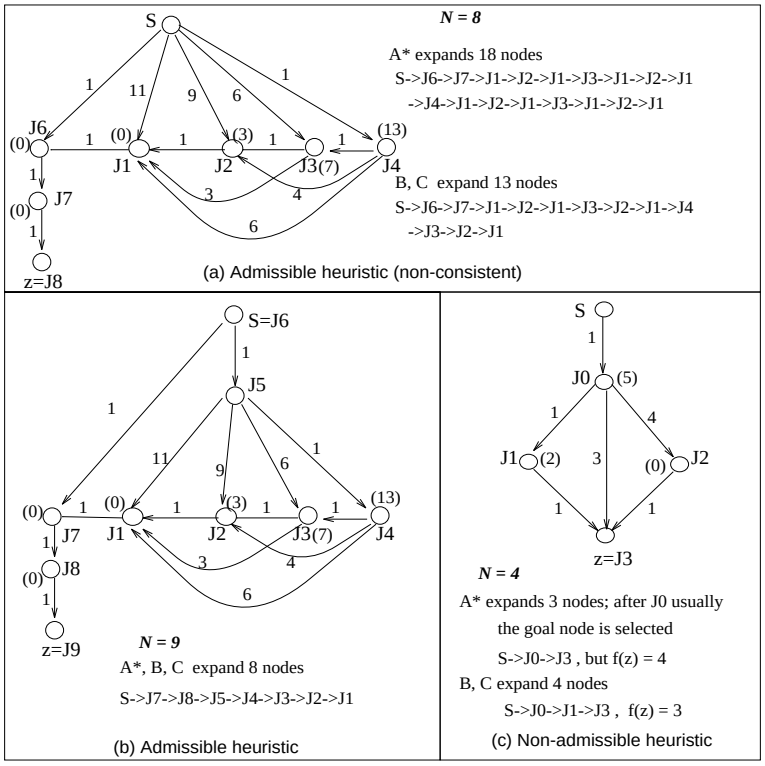


Figure 6.7 Comparison of algorithms A, B, C.

Algorithm D is similar to A^* except for the new steps 1 and 3:
$D3$: Look for nodes in OPEN with the smallest f-value and select among them the node n with the smallest g-value. (Resolve loops with respect to a goal node). Move n from OPEN to CLOSED.
$D5$: ... (IF n has no successor nodes THEN set $h(n) \leftarrow \infty$).
$D6$: FOR each successor n_i of n
DO: $g_i \leftarrow g(n) + c(n, n_i)$,
$e \leftarrow \min_i\{c(n, n_i) + h(n_i)\}$,
IF $h(n) < e$ THEN set: $h(n) \leftarrow e$,
ELSE FOR each n_i DO:
IF $h(n_i) < h(n) - c(n, n_i)$
THEN $h(n_i) \leftarrow h(n) - c(n, n_i)$.

Table 6.3 D-graph search algorithm.

A^*-tree search complexity. The number of goal nodes can be a polynomial function of N only. The interchange between heuristic quality and search complexity is not elastic. A very high precision of heuristic is required in order to reach a polynomial complexity [PEA84].

	Worst complexity	Result under: admissible h	non-admissible h	Comments
A	$O(2^N)$	$h(s)$	$f_A \leq Q$	
B	$O(\mathcal{N}^2)$	$h(s)$	$f_B \leq f_A \leq Q$	last_$F = Q$
C	$O(\mathcal{N}^2)$	$h(s)$	$f_C \leq f_B \leq f_A \leq Q$	last_$F = Q$ sometimes: $V(C) \supset V(B)$
D	$O(N^2)$	$h(s)$	$f_D = f_C = C_{min}$	$V(D) \subseteq V(C)$

Table 6.4 Comparison of optimal graph search methods.

If for some application the design of a "good" heuristic judgement is not possible, but one wants to keep the search complexity as low as possible, *sub-optimal* search methods (e.g. [PEA84]) or *locally optimal* search (e.g. [KOR90]) and *hybrid* methods (optimal search combined with depth-first search) (e.g. [KOR85, CHA89]) may be considered.

6.5 CONCLUSIONS

In addition to typical single object tracking problems, in this chapter the focus was on many object scene descriptions and on mechanisms for the selection of the best subsets of tracked hypotheses. In this context we identify image measurement problems that may prohibit the use of a Kalman filter (KF). The KF requires that individual measurements are statistically independent. This is often not the case in vision tracking systems, where one usually selects those segments which best fit the hypothesis expectations. Furthermore, KF does not require judgments of each single measurement, assuming every measurement in given channel to be equally probable. This does not allow one to track highly non-stationary signals, i.e. to follow abrupt changes of the tracked signal. We propose a necessary modification of the recursive estimator, that follows the idea of self–adaptation of learning rates in the unsupervised learning of neural networks [AMA67, CIK96b].

We shall apply the proposed adaptive object recognition scheme to different analysis problems at three abstraction levels. In chapters 7–9 computational solutions of the following tasks are described: 2–D segment tracking and recognition, model-based road and vehicle recognition, and finally knowledge-based scene description [KAS94b, KAS95a, KAS97, KAS98].

Currently, simplified schemas, compared to the general object recognition scheme, are used for segment tracking and road recognition. In both implementations the transition and projection functions ($\boldsymbol{F}(k)$ and $\boldsymbol{H}(k)$) are reduced to linear transformations, hence a linear estimator is sufficient for them. The number of required tokens (objects) is known in advance in some cases (one vanishing point and one road state), whereas the number of segments and consistent 3–D object hypotheses is generally unknown. Thus, the hypothesis selection search is simplified in the first case. Finally a complex object structure can be distinguished for 3–D vehicle objects, but the two remaining implementations deal with homogeneous objects only - a simplified recognition search task appears in the latter case.

7 IMAGE SEGMENT TRACKING AND RECOGNITION

Most of the work that has been done on visual motion estimation is limited to a stationary camera case (without ego-motion). In this chapter the focus is on the specific application of vision systems for traffic scene evaluation [KOL93,TAN93,COX96,IRA98]. The main goal of such systems is traffic flow analysis and evaluation. The characteristic features of such application systems that we assume here are:

- mostly longitudinally moving objects on the road plane appear,
- the perspective projection determines the projection conditions,
- a stationary camera provides the image data, but a small nodding movement of the camera can always appear.

Image segment tracking is another way to determine visual motion, in this case called - *discrete motion* map [DER90,SAL90]. This kind of visual motion allows one, for example, to perform a 3-D structure reconstruction process for known and restricted object classes [TSA84, SUB88].

In this context two implementations of the adaptive object recognition scheme are proposed: an image segment tracking method (for the estimation of translational motion of objects corresponding to segments) [KAS93a, KAS93c, KAS93b] and a vanishing point recognition method (for automatic on-line calibration of projection conditions) [KAS97, KAS98].

7.1 IMAGE SEGMENT TRACKING

In the past it was proposed to track different discrete image segments. Let us detect a block of pixels that contains a 2–D texture in a first image. The corresponding problem for this block in the next image from the given image sequence may be solved by *correlation methods* [EKL94]. The visual motion vector for this block is determined by tracking its position continuously from image to image.

In the *edge tracking* method [DER90], the image location and direction of a tracked edge is predicted. Around this expectation a search area is built. In this area the best corresponding edge is searched for. Most often this requires the use of heuristic methods for pruning the correspondence space [ZHA94].

If point features (like for example corners) can be distinguished in the image, the trajectory determination can give good results [SAL90, COX96]. Characteristic object boundary points can define an object contour. This contour can change its shape in an elastic way during the tracking process, called *active contour* tracking [KAW88, BLA93].

The correspondence of more complex image features like, for example, *local symmetries* can also be searched for [BER93]. In this approach the detection and tracking of complex objects, like vehicles, as groups of pairs of symmetrical features is attempted. In similar approaches the correspondence and tracking of elements of a 2-D object model

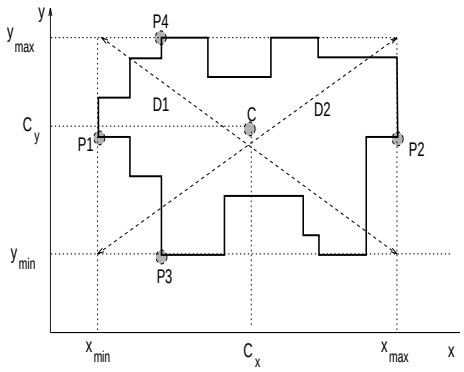


Figure 7.1 Discrete contour and its features.

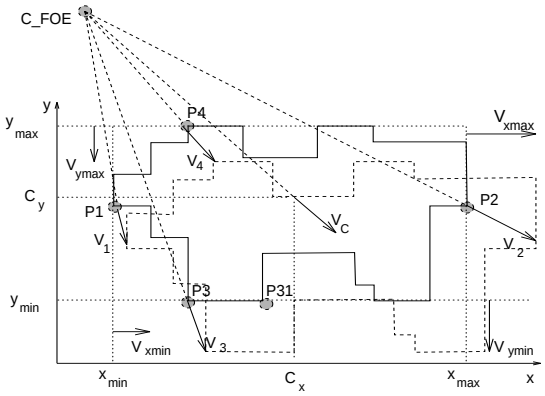


Figure 7.2 Dynamic features of a discrete contour.

is performed, on the assumption that nearly always a front or back view only of a 3-D object is available [REG94, SCH92]. But the main goal of these methods is the stable detection and tracking of the image position of the object and not the estimation of its visual motion.

By processing a short image sequence the ultimate goal may also be to detect the *global depth* of object features or the *3–D structure* of an object [SUB88]. It is assumed that some significant object views exist in consecutive images and that discrete object feature tracking in these images is performed [TSA84] or a dense optical flow is available [WAX88]. In both cases the image data is used to solve the non-regular inverse 3–D projection transformation of an object, belonging to a given class. Additional regularity constraints are usually necessary in order to propose robust methods. A simplification occurs if an orthographic projection can be applied [HUA94], although even in this case at least three significant views are required.

For outdoor scenes it was tried to detect short trajectories of many significant points of a ground plane moving object and to determine its 3–D structure [DRE82] or at least its depth and motion [TAN93, TAN94]. In practice these methods require a stationary camera, a nearly orthographic projection and a single object window of sufficiently large dimension.

7.1.1 Contour state vector

We represent image segments and segment groups by their closed boundary contours. The following geometric features of a contour are distinguished (*Figure* 7.1):

- mass center position $C = (C_x, C_y)$,
- points of maximum enlargement $\{P1, P2, P3, P4\}$ in each direction $x+, x-, y+, y-$,
- the boundary box $(x_{min}, x_{max}, y_{min}, y_{max})$,
- the discrete contour length l,
- the sum of two diagonal distances: $d = |\vec{D1}| + |\vec{D2}|$.

On the basis of the motion mask the ratio of "moving" pixel of the contour to the overall pixel number is computed as a "dynamic" contour feature.

The dynamic features of a contour are obtained due to the short time correspondence of this contour found in up to N consecutive images ($N = 2 - 5$). The following dynamic contour features are available (*Figure* 7.2):

- visual motion of specific positions - mass center motion $v_C = (v_{Cx}, v_{Cy})$, boundary box motion $(v_{xmin}, v_{xmax}, v_{ymin}, v_{ymax})$, up to four point motions - $v^{P1} = (v_x^{P1}, v_y^{P1})$, $v^{P2} = (v_x^{P2}, v_y^{P2})$, $v^{P3} = (v_x^{P3}, v_y^{P3})$, $v^{P4} = (v_x^{P4}, v_y^{P4})$;
- length change rate of contour length v_z^l and of diagonal distance v_z^d.

7.1.2 Measurement - weighted averaging

The continuous contour change is approximated by a weighted averaged change of a discrete contour. Let $[Pel]$ denote the pixel size and τ be the constant time interval between two consecutive images. The dynamic contour features are computed as follows:

$$\begin{array}{ll} v_x = \frac{1}{(N-1)\tau} \sum_{i=1}^{N-1} \phi_i (x(i) - x(0)), & v_y = \frac{1}{(N-1)\tau} \sum_{i=1}^{N-1} \phi_i (y(i) - y(0)) \ [Pel/\tau], \\ v_z^l = \frac{1}{(N-1)\tau} \frac{\sum_{i=1}^{N-1} \phi_i (l(i) - l(0))}{l(0)}, & v_z^d = \frac{1}{(N-1)\tau} \frac{\sum_{i=1}^{N-1} \phi_i (d(i) - d(0))}{d(0)} \ [1/\tau]. \end{array} \tag{7.1}$$

The length changes $\{v_z^l, v_z^d\}$ are relative motion measures, whereas $\{v_x, v_y\}$ is the usual visual motion vector. The weights $\{\phi_i | i = 1, ..., N-1\}$ should correspond to the depths $Z_C(i)$ of the object points, projected to contour points $(x_C(i), y_C(i))$:

$$\phi_i = \frac{x_C(1) - x_C(0)}{x_C(i) - x_C(0)} = \frac{y_C(1) - y_C(0)}{y_C(i) - y_C(0)} = \frac{Z_C(i)}{i \ Z_C(1)}. \tag{7.2}$$

Obviously these depths are not known in advance, but they can be roughly estimated from the contour change ratio v_z. If the contour plane is approximately parallel to the image plane the relation between v_z and Z_C is given as:

$$\Delta Z = Z_C(i+1) - Z_C(i) \simeq Z_C(0) \ \Big(\frac{1}{1 + v_z} - 1 \Big). \tag{7.3}$$

But v_z also depends on the same weights. An iterative simultaneous computation of v_z and $\{\phi_i | i = 1, ..., N-1\}$ is performed, with following initial data:

$$\phi_i(Init) = \frac{1}{i}, \qquad v_z(Init) = \frac{1}{N-1} \sum_{i=1}^{N-1} \phi_i(Init) \frac{(l(i) - l(0))}{l(0)}. \tag{7.4}$$

In each iteration step the weights are obtained from:

$$\phi_i = \frac{Z_C(i)}{i \ Z_C(1)} \simeq \frac{1 + i\gamma}{i \ (1 + \gamma)}, \qquad \text{where} : \gamma = \Big(\frac{1}{1 + v_z} - 1 \Big). \tag{7.5}$$

7.1.3 Recursive estimation

The state vector of a contour segment hypothesis consists of the geometric and dynamic features, as specified above, i.e.:

$$\boldsymbol{s}_C(k) = [l(k), C_x(k), C_y(k), ..., v_x(k), v_y(k), v_z(k)]^T. \tag{7.6}$$

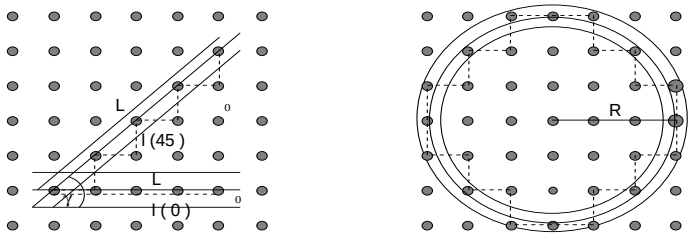


Figure 7.3 Digitalization errors for straight line segment and circle.

The appropriate variances along the diagonal of the covariance matrix for state $\boldsymbol{s}_C$ are denoted as:

$$diag(\boldsymbol{P}_C(k)) = [P_l(k), P_{Cx}(k), P_{Cy}(k), ..., P_{vx}(k), P_{vy}(k), P_{vz}(k)]^T. \tag{7.7}$$

The measurement vector is of similar structure to the state vector:

$$\boldsymbol{m}_C(k) \leftrightarrow \boldsymbol{s}_C(k), \tag{7.8}$$

although only the position is measured directly.

Due to the weighted averaged measurements a reliable judgement scheme for vector $\boldsymbol{m}_C(k)$ can be provided. The error of a geometric or dynamic contour feature is related to the difference of individual measurements from (motion corrected) average measurement in up to N images (or $N-1$ for dynamic features). For each single image feature its difference to an approximated ideal feature is computed. The set of N individual errors (or $N-1$ respectively) can be seen as an error variable and the variance of this variable is used for a quality judgement. For example, the error variance of a current averaged contour length measurement is given as:

$$R_l = \frac{1}{N}\sum_{i=0}^{N-1} l_i * V_(i), \quad \text{where} \quad V_(i) = \frac{(l_i - \widehat{l_i})^2}{l_i}, \tag{7.9}$$

and $\widehat{l_i}$ is the expected value of the individual $i-$th measurement.

Due to the above measurement judgements for the recursive estimation the Kalman filter can be applied. As there is in fact an identity projection function between the state and measurement vector a linear estimator is sufficient for this application.

7.1.4 Digitalization errors

Due to the discrete image space there can be two error sources of a discrete representation l of a continuous contour L in image plane: the *digitalization system* error and the *size tolerance* error [KAS93a, KAS97]. The system error strongly depends on the form of the contour and on its direction in the image coordinate system. This error could be avoided if the proper mapping scheme of the discrete contour to a continuous line, circle etc. is known. The size tolerance error is directly related to the discrete image mesh (resolution). Even if the continuous contour is properly derived from the discrete one, tolerances of the positions of vertices or center points must be still taken into account.

For a straight line segment of length L the system error depends on the direction of the line segment. If the line segment is parallel to one of the image axes ($\gamma = 0^o$), then there is no system error at all (*Figure* 7.3). The size tolerance error is assumed to be $\pm 1/2 \cos\gamma$ at each segment end (but the continuous length L must be $> 1/2$). Hence, L is given as [KAS93a]:

$$L = l\ \frac{\sqrt{1+\tan^2\gamma}}{1+\tan\gamma} \pm \frac{1}{\cos\gamma}\ =\ \frac{l}{\sin\gamma+\cos\gamma} \pm \frac{1}{\cos\gamma},\ (0 \le \gamma \le 45^o). \tag{7.10}$$

For a n-sided polygon of total length L_{poly}, with n line segments $L_i (i = 0, ..., n-1)$, the general relation between its continuous image length L_{poly} and the length $l_{poly} = \sum_{i=0}^{n-1} l_i$ of its discrete representation is as follows:

$$L_{poly} = \sum_{i=0}^{n-1} \left(\frac{l_i}{\sin\gamma_i + \cos\gamma_i}\right) \pm \sum_{i=0}^{n-1} \frac{1}{\cos\gamma_i},\ (0 \le \gamma_i \le 45^o; i = 0, ..., n-1). \tag{7.11}$$

The length l_c of a discrete mapping of a continuous circle with radius r is approximately given as ([VOS88], p.121): $l_c(r) = 8[ent(r)]$, where $ent(r)$ denotes the integer value nearest to r. This is the system digitalization error of a circle. The tolerance error for radius r or location of the center point can be assumed to be $\pm 1/2$. If $L_c = 2\pi r$ is the real boundary length, then for the total digitalization error it holds that:

$$\begin{array}{rll} \frac{\pi}{4} l_c - \pi & < L_c < & \frac{\pi}{4} l_c + \pi, \quad (l_c \ge 8) \\ \frac{\pi}{2} & < L_c < & 2\pi, \qquad\quad (l_c = 4). \end{array} \tag{7.12}$$

Let us assume that the continuous (real) visual motion component $\tilde{\boldsymbol{v}}_x$ (or $\tilde{\boldsymbol{v}}_y$) is of positive value. The maximum discrete overestimation v_x^{max} (or v_y^{max}) of $\tilde{\boldsymbol{v}}_x$ (or $\tilde{\boldsymbol{v}}_y$) is defined as [KAS93a]:

$$v_x^{max} = \tilde{\boldsymbol{v}}_x + \frac{2}{N};\ v_y^{max} = \tilde{\boldsymbol{v}}_y + \frac{2}{N}. \tag{7.13}$$

Similarly, the largest underestimation v_x^{min} (or v_y^{min}) of a negative $\tilde{\boldsymbol{v}}_x$ (or $\tilde{\boldsymbol{v}}_y$) is:

$$v_x^{min} = \tilde{\boldsymbol{v}}_x - \frac{2}{N};\ v_y^{min} = \tilde{\boldsymbol{v}}_y - \frac{2}{N}. \tag{7.14}$$

Clearly the digitalization error of v_x and v_y depends on the image mesh value only. The digitalization error of a continuous $\tilde{\boldsymbol{v}}_z$ depends on the tolerance digitalization error of a contour only. Hence, there is no need to approximate the set of elementary edges by polygons before the visual motion is computed.

The general equations for a discrete representation of a n-sided polygon lead to a general equation for the maximum overestimation $v_{z\_poly}^{max}$ of $\tilde{\boldsymbol{v}}_z$ [KAS93a]:

$$v_{z\_poly}^{max} = (1 + \mathcal{K}_{poly})\tilde{\boldsymbol{v}}_z + \frac{4}{N}\mathcal{K}_{poly}, \tag{7.15}$$

$$\mathcal{K}_{poly} = \frac{\frac{n}{\cos\gamma}}{L_{poly} - \frac{n}{\cos\gamma}} \quad , \quad \mathcal{K}_{poly}(\gamma = 0^o) < \mathcal{K}_{poly} < \mathcal{K}_{poly}(\gamma = 45^o).$$

Dually, the worst underestimation-case of the polygon motion is approximated by:

$$v^{min}_{z\_poly} = (1 - \mathcal{K}_{poly})\tilde{\boldsymbol{v}}_z - \frac{4}{N}\mathcal{K}_{poly}. \tag{7.16}$$

Similar assumptions for a circle with radius r (such that $l_c \geq 8$) lead to the following worst-case overestimation $v^{max}_{z\_circle}$ and underestimation $v^{min}_{z\_circle}$ of a positive-valued motion $\tilde{\boldsymbol{v}}_z$ [KAS93a]:

$$v^{max}_{z\_circle} = (1 + \mathcal{K}_{circle})\tilde{\boldsymbol{v}}_z + \frac{4}{N}\mathcal{K}_{circle}; \quad v^{min}_{z\_circle} = (1 - \mathcal{K}_{circle})\tilde{\boldsymbol{v}}_z - \frac{4}{N}\mathcal{K}_{circle}; \tag{7.17}$$

$$\text{with} \quad \mathcal{K}_{circle} = \frac{\pi}{L_c - \pi}.$$

Example 7.1 - (all calculations for N=5)

1. For a polygon with 4 sides and with continuous length $L_{poly} = 84$ the estimated motion v_z is related to continuous motion $\tilde{\boldsymbol{v}}_z$ by following inequalities:

$$0.928\tilde{\boldsymbol{v}}_z - 0.058 < v_z < 1.072\tilde{\boldsymbol{v}}_z + 0.058, \text{where} \quad \mathcal{K}_{poly} < \frac{4\sqrt{2}}{84 - 4\sqrt{2}} \simeq 0.072.$$

2. For a circle with boundary 21π ($r = 10.5$) it holds that:

$$0.95\tilde{\boldsymbol{v}}_z - 0.04 < v_{z\_circle} < 1.05\tilde{\boldsymbol{v}}_z + 0.04, \text{where} \quad \mathcal{K}_{circle} < \frac{\pi}{21\pi - \pi} = 0.05.$$

3. For both polygon and circle it holds that:

$$\tilde{\boldsymbol{v}}_x - 0.4 < v_x < \tilde{\boldsymbol{v}}_x + 0.4 \quad ; \quad \tilde{\boldsymbol{v}}_y - 0.4 < v_y < \tilde{\boldsymbol{v}}_y + 0.4.$$

For the circle in point 2. the estimation of v_z is of better quality than the estimation of v_x (or v_y) if: $\tilde{\boldsymbol{v}}_x < \frac{0.4\ \tilde{\boldsymbol{v}}_z}{0.04 + 0.05\tilde{\boldsymbol{v}}_z}$, i.e. if $\tilde{\boldsymbol{v}}_z = 0.1$ then it must be $\tilde{\boldsymbol{v}}_x < 0.889$; if $\tilde{\boldsymbol{v}}_z = 0.01$ then $\tilde{\boldsymbol{v}}_x < 0.099$.

From the above analysis of discrete mapping errors one can conclude that for a sufficiently large L and $\tilde{\boldsymbol{v}}_z$ the motion estimate v_z is more reliably given than than the length l itself.

Example 7.2 - The worst-case errors for a 4-sided polygon with $L_{poly} = 80$ and $|\tilde{\boldsymbol{v}}_z| = 0.2$ (if N=5) are: $0.95L_{poly} < l_{poly} < 1.44L_{poly}; 0.62|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.38|\tilde{\boldsymbol{v}}_z|$. If $L_{poly} = 160$ and $|\tilde{\boldsymbol{v}}_z| = 0.1$ (N=5) then: $0.975L_{poly} < l_{poly} < 1.427L_{poly}; 0.67|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.33|\tilde{\boldsymbol{v}}_z|$.

The above analysis underlines the importance of accumulated motion estimation. In contrast to the length digitalization error, which is relatively constant in a longer sequence and which depends mainly on the system digitalization error, the v_z-estimation error is a tolerance error. There is a low probability of a worst-case tolerance error in most of the long sequence-frames. Thus a better quality of an accumulated estimation of v_z than the worst-case quality of measured v_z is expected.

Even in the worst case of length digitalization, the digitalization error of the relative length change rate v_z is not affected by the length error, being a tolerance error only.

Figure 7.4 Real image (left) and the vector field for updated (v_x, v_y) motion.

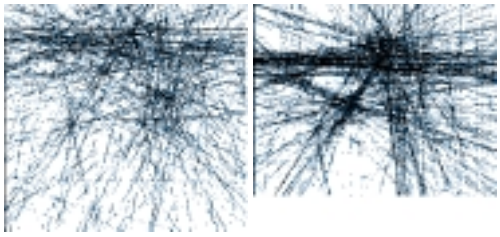


Figure 7.5 Line fields for initial (left) and updated (right)(v_x, v_y)-motion.

7.1.5 Interpretation of contour motion as object motion

In images of road scenes the contour motion in the image plane corresponds to some object motion in the road plane [KAS93b].

Let $\mathcal{L}$ be the length of a line segment between points P_1 and P_2, with depths Z_1 and Z_2 respectively, that is projected to image edges with lengths $L(k), L(k+1)$ at time points t_k, t_{k+1}. A *virtual* segment corresponding to L at time t_k has the length $\mathcal{L}_v(k)$, it is parallel to the image plane, it projects to the same edge in the image as $\mathcal{L}$ does and its depth is: $Z_C = (Z_1 + Z_2)/2$. The center point of a virtual line segment is projected to image points $p(k) = (x, y, 0)^T$ and $p(k+1) = (x + v_x, y + v_y, 0)^T$ respectively.

The lengths $L_v(k), L_v(k+1)$ are approximated by discrete edge lengths $l(k), l(k+1)$:

$$\frac{L(k+1)}{L(k)} \simeq \frac{l(k+1)}{l(k)} \quad = \quad \frac{l(k) + \Delta l}{l(k)} = 1 + v_z. \tag{7.18}$$

The relation between the measured relative length change rate v_z of a contour and the translational velocity $V^c = (\Delta X, \Delta Y, \Delta Z)^T$ of the line segment is specified as [KAS97]:

$$\mathcal{L}_v(k+1) = \theta(k)\mathcal{L}_v(k) \quad \Rightarrow \quad \Delta Z(k) \simeq Z_C(k) \left(\frac{\theta(k)}{1 + v_z(k)} - 1 \right) \tag{7.19}$$

$$\Delta X = \frac{1}{F}\big[\Delta Z(x + v_x) + v_x \; Z(k)\big]; \quad \Delta Y \quad = \frac{1}{F}\big[\Delta Z(y + v_y) + v_y \; Z(k)\big]. \tag{7.20}$$

If the line segment is nearly parallel to the image plane or $(Z_C(k) >> \mathcal{L}_v(k))$, we set $\theta(k) = 1.0$. In other cases the parameter $\theta(k)$ depends on the depth, length, and orientation of the line segment (in practical applications it varies between 0.95 and 1.05). During hypothesis generation the estimates for these parameters are provided by model–based restrictions. Hence the correction coefficient $\theta(k)$ can be estimated as well.

7.1.6 Test results

A single image from a real sequence of images with resolution 256x256x8 bit (left image) and the corresponding contour motion field $\{(v_{Cx}, v_{Cy})\}$ are presented in *Figure* 7.4. The example in *Figure* 7.5 verifies the stabilization ability of the (v_{Cx}, v_{Cy}) motion tracking procedure. For each (x, y)-motion vector its direction line going through the contour center point is drawn across the image. Two motion vectors are similar if their directions point toward the same *focus of expansion* point (FOE). A properly segmented

State parameter	Image k =						
	2	5	8	11	14	17	20
Contour length l $[Pel]$	34	40	48	74	78	92	106
Measurement:							
VP–based p^c_{SZ} [m]	-43.70	-39.40	-46.15	-38.95	-27.00	-25.95	-18.10
v_z–based V_S $[m/\tau]$	1.46	1.45	1.38	1.33	1.10	1.29	1.15
First stripe:							
p^{c+}_{SZ} [m]	-43.70	-39.35	-36.61	-33.99	-30.89	-27.40	-23.27
V^+_S $[m/\tau]$	1.46	1.44	0.91	0.87	1.04	1.16	1.38
	Image k =						
Second stripe:	23	26	29	32	35	38	41
p^{c+}_{SZ} [m]	-43.27	-39.18	-35.79	-32.81	-29.49	-26.13	-22.41
V^+_S $[m/\tau]$	1.38	1.36	1.13	0.99	1.11	1.12	1.24

Table 7.1 The measurements and estimations of depth positions and translatory motion of relatively moving road stripes (p_{SX}, V_S).

image should contain several FOE-points, each corresponding to a moving object or the stationary background.

In natural scenes the detection and tracking of segment groups is more reliably done than single contour tracking. In our application road stripe objects exist, which correspond nearly always to a single contour. We simulate a really moving object by tracking middle road stripes in image sequences under ego-motion. In *Table* 7.1 some results of road stripe tracking in real image sequence are provided. They show the quality improvement due to the recursive stabilization of the contour state in a longer image sequence. A single middle stripe is observed in a sequence of about 20 images. The next stripe is initialized with the relative motion estimated for its predecessor stripe. Obviously the motion estimation improves with each next stripe. Large measurements errors can be observed for the images from 7 to 12, as the lighting condition varies. Nevertheless the estimated values are always of good quality.

How large are digitalization errors usually contained in the test results? In order to answer this question we have simulated a moving object in a synthetic image of similar size to the real image. For a synthetic image the segmentation error is small and can be omitted from consideration. The results of tracking a middle stripe in ten *N-Images* of a synthetic traffic scene are summarized next. As it was expected, the main error appeared in the estimation of the continuous contour length L - 13.7% or 12.1 Pel. Nevertheless the performance of the visual motion was much higher than that of l, even without correction. The final error of updated v_z and v_C did not exceed 0.6 Pel/τ, i.e. 14%. Let us compare this result with the worst-case of digitalization error. For a contour of length $L = 80$ Pel and visual motions of $|\tilde{\boldsymbol{v}}_z| = 0.2$ and $|\tilde{\boldsymbol{v}}_x| = 2.0$ the following maximal errors are possible ($N = 2, 5$):

$$N = 2 \Rightarrow (0.50|\tilde{\boldsymbol{v}}_x| < |v_x| < 1.50|\tilde{\boldsymbol{v}}_x|, \qquad 0.802|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.198|\tilde{\boldsymbol{v}}_z|),$$
$$N = 5 \Rightarrow (0.80|\tilde{\boldsymbol{v}}_x| < |v_x| < 1.20|\tilde{\boldsymbol{v}}_x|, \qquad 0.910|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.090|\tilde{\boldsymbol{v}}_z|).$$

Similarly for $L = 40$ Pel, $|\tilde{\boldsymbol{v}}_z| = 0.2$ and $|\tilde{\boldsymbol{v}}_x| = 4.0$ it follows that:

$$N = 2 \Rightarrow (0.75|\tilde{\boldsymbol{v}}_x| < |v_x| < 1.25|\tilde{\boldsymbol{v}}_x| \quad ; \quad 0.597|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.403|\tilde{\boldsymbol{v}}_z|,$$

$$N = 5 \Rightarrow (0.90|\tilde{\boldsymbol{v}}_x| < |v_x| < 1.10|\tilde{\boldsymbol{v}}_x| \quad ; \quad 0.817|\tilde{\boldsymbol{v}}_z| < |v_z| < 1.183|\tilde{\boldsymbol{v}}_z|.$$

From above it is evident that the estimated values in the synthetic image are of much better quality than the possible worst-case measurements.

7.2 VANISHING POINT RECOGNITION

The purpose of vanishing point (VP) recognition in images of traffic scenes is to estimate the current orientation of the camera in relation to the road plane and the road trajectory. The state vector of a VP–hypothesis consists of the localization coordinates and a translation vector in the image plane:

$$\boldsymbol{s}_{VP}(k) = [x(k), y(k), \delta x(k), \delta y(k)]^T. \tag{7.21}$$

The appropriate variances (the diagonal of the covariance matrix for state $\boldsymbol{s}_{VP}$) are:

$$diag[\boldsymbol{P}_{VP}(k)] = [P_x(k), P_y(k), P_{\delta x}(k), P_{\delta y}(k)]^T. \tag{7.22}$$

The measurement vector consists of position and displacement parameters only:

$$\boldsymbol{m}_{VP}(k) = [m_x(k), m_y(k), m_{\delta x}(k), m_{\delta y}(k)]^T \leftrightarrow \boldsymbol{s}_{VP}(k), \tag{7.23}$$

although only the position is measured directly.

7.2.1 The recognition procedure

As there is in fact an identity projection function between the state and measurement vector, a linear estimator is applied. A reliable measurement judgement scheme is also defined. Hence a linear Kalman filter is applied in this implementation. The structure of the VP recognition procedure follows the general adaptive approach, i.e. it consists of measurement, initialization or modification, prediction and selection steps (*Table* 7.2).

7.2.2 The VP measurement

The VP–measurement process consists of two sub–steps *Figure* 7.6. The first sub–step performs a processing of the whole image at once, whereas the second sub–step is repeated for all predicted VP–hypotheses and it tries to match each detected density peak with each predicted VP–hypothesis.

From the set of all segments (*Figure* 7.6(b)) the assumed road stripe segments and road border segments are selected first. Such segments are mainly vertically elongated in the image (i.e. the relation of their width to height is larger than a given threshold value), the neighbor region intensities are "white" and these segments are located on the bottom of the image, below the current vanishing point hypothesis (*Figure* 7.6(c)).

Only those line segments are finally selected, that are located on lines, which are very near to at least one vanishing point hypothesis. In the above line selection step the reduction rate of line segments cannot be too high, eg. the elimination of more than 50%

For every image in the sequence the following steps are performed:

1. Image measurement:
   a) Detection of current vanishing point lines in the image.
   b) Addition of these lines to the accumulating density image and calculation of the whole sum of accumulated pixel values.
2. FOR each vanishing point hypothesis DO:
   a) Prediction of the current state in the image plane: $\boldsymbol{s}^+_{VP}(k) = [x^+, \delta x^+, y^+, \delta y^+]^T$ and of its covariance matrix $\boldsymbol{P}^+_{VP}(k)$.
   b) Direct measurement A
      i. Determine the search area around the currently expected position – the estimation variance induces the size of this area.
      ii. Perform a search in the selected area for a point area with the highest density value – a small point–like area is detected, where its center point is the current measurement $\boldsymbol{y}_{VP}(k)$.
      iii. Set the variances of current position detection in $\boldsymbol{R}(k)$ by relating the point density to the overall density.
   c) Current gain $\boldsymbol{K}_R = [K_x(k), K_y(k)]^T$ is obtained as:

$$K_x(k) = \frac{P^+_x(k)}{P^+_x(k) + R_x(k)} \quad ; \quad K_y(k) = \frac{P^+_y(k)}{P^+_y(k) + R_y(k)}. \tag{7.24}$$

   d) Estimation (modification, innovation) of location parameters and their variances:

$$\boldsymbol{s}^*_R(k) = \boldsymbol{s}^+_R(k) + \boldsymbol{K}_R(k)\left[(\boldsymbol{m}(k) - \boldsymbol{s}^+_R(k)\right] \tag{7.25}$$

$$\boldsymbol{P}^*_R(k) = [1 - \boldsymbol{K}_R(k)]\,\boldsymbol{P}^+_R(k) \tag{7.26}$$

   e) Measurement of image motion features:

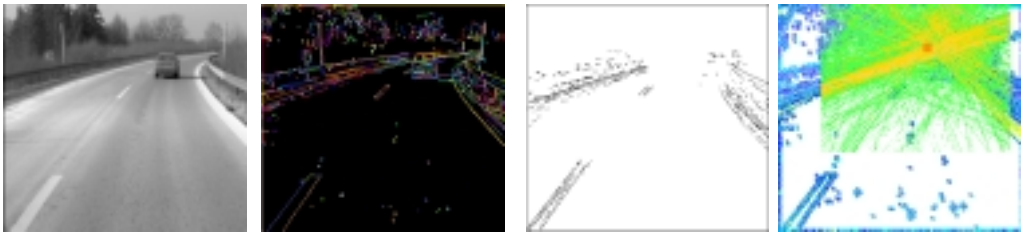
$$\delta x(k) = x^*(k) - x^*(k-1) \;\text{ and }\; \delta y(k) = y^*(k) - y^*(k-1). \tag{7.27}$$

   f) New estimation of motions $\delta x^*(k), \delta y^*(k)$ and of their variances - similar to c)-d).
3. For all significant peaks in the density image, that are not measurements of any current VP-hypothesis - initialize new VP-hypotheses.
4. Select the current best VP-hypothesis.

Table 7.2 The procedure of VP recognition.

of prospective line segments may indicate that we are concentrating on the wrong hypotheses. Hence, if the reduction rate violates some threshold then a new hypothesis is added, different to the existing ones.

The lines corresponding to selected line segments are added to the accumulating density image. The location and size of this accumulator are determined by the current prediction of the given hypothesis. In this accumulator image the process is looking for a small circular area with the highest density value. The center point of such an area constitutes the current location measurement $\boldsymbol{y}_{VP}(k)$ (*Figure* 7.6 (d)). The variance of position measurement in $\boldsymbol{R}(k)$ is determined by the relation of the selected point density to the sum of all densities in the accumulator.



(a) (b) (c) (d)

Figure 7.6 Steps in vanishing point measurement (detection): (a) normalized image, (b) edge image, (c) VP-line segments, (d) density search in the accumulated VP-line segment image, around the expected position of VP.

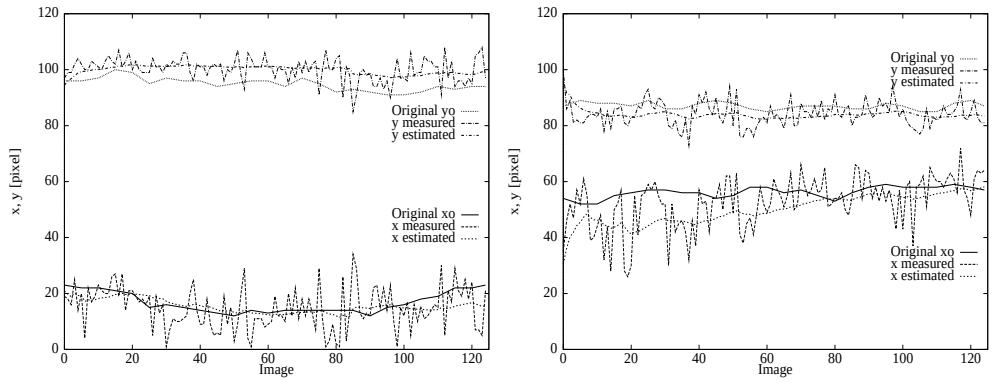


Figure 7.7 Vanishing point measurement (detection) and stabilization in two sequences – with linear (left drawing) and curved (right drawing) road elements.

During the final selection step a single, best hypothesis is selected, but some other hypotheses are still tracked and the remaining may be eliminated. This selection depends on the variances of states estimations, on the stability factor and on the tracking time of the given hypothesis.

7.2.3 Test example

The vanishing point recognition method was tested on several 8-bit image sequences with 125 – 500 non–interlaced images of resolution 351 x 283 pixel each. There were traffic scene images from a stationary camera and image sequences of motor-way scenes and federal road scenes provided (moving camera) [KAS97, KAS98].

In order to evaluate the quality of the vanishing point estimation, we manually detected the original, reference positions $\{(x_o, y_o)(k),\ (k = 0, ..., 124)\}$ in each frame of the sequence. The drawings in *Figure* 7.7 verify that the measurement (detection) error $(VP_y - y_o)$ was always below $\pm 10[Pel]$ and the detection error $(VP_x - x_o)$ was below $\pm 16[Pel]$. Due to recursive estimation the quality of VP point was improved - the error $(VP_y^* - y_o)$ was below $\pm 5[Pel]$ and the error $VP_x^* - x_o$ was below $\pm 10[Pel]$.

7.3 CONCLUSIONS

Two implementations of the adaptive object recognition scheme have been developed, that deal with segment tracking and vanishing point recognition.

The developed contour tracking method is specially suitable for longitudinal motion, i.e. if the objects or the camera are moving forward or backward along the longitudinal projection axis. A segment grouping process explores both visual motion mask and geometry information in order to group neighbor segments. The groups are represented by their boundary contours in the image [KAS93a, KAS93c, KAS93b]. A sub-pixel accuracy of motion vector components is achieved by two stabilization steps applied to the displacement vectors of image segments. Instead of a simple averaging of the displacements a weighted one, taking into account N consecutive images, is used. The weights depend on the translation of the observed object along the camera's depth axis. The recursive contour state estimation during a long image sequence constitutes the next stabilization step.

The second method performs an adaptive recognition of a specific virtual segment in the image plane, the road's vanishing point (VP) [KAS97, KAS98]. This allows the correction of projection errors caused by an unknown nodding movement of the camera. In road traffic analysis relative long edges of vertically elongated contours are supposed to be the *VP lines*. The highest density area of intersection points between these lines is detected as the VP point. The correct rotation angles $\alpha(k)$ and $\beta(k)$ of the camera orientation are directly derived from the estimated position of VP point. Up to several competitive vanishing points are tracked in parallel until the best one is recognized.

8 MODEL-BASED OBJECT RECOGNITION IN IMAGE SEQUENCES

Autonomous navigation and driver support [DIG88,MAS92,GRA93,THO93,WET94] is an attractive and challenging application field for image sequence analysis systems. The partly unknown ego-motion of the camera requires model-based approaches to the main two tasks of such vision systems: (1) the *road following* and *ego-state tracking* task [DIM92], and (2) the obstacle (on-road 3–D objects) detection task [REG94]. If a large piece of very distant road border is visible in the image, i.e. the curvature "is measurable" in the image, then the road following task can also include the estimation of *road curvature* [POL92]. There already exist solutions to the first task [DIM92, POM93, JOC96, MAU96] although they still seem not to be sufficiently robust. There is a problem of automatic orientation recognition (overcoming the camera nodding movement) and of recognizing the road if many obstacles exist in the scene. A reliable detection and classification of obstacles in images of many–object scenes is still a challenging problem [SCH92, REG94, WET94].

The complex nature of the subject makes it necessary to apply a dynamic model–based analysis, which usually performs object tracking and constrains the classes of recognized objects [GEN92, WUR88, KOL93]. A robust recognition system is required, that not only tracks already known image features (due to manual or sub–manual initialization) but also automatically detects new features and is able to adapt to changes of the scene environment.

In this chapter we apply the adaptive recognition approach (chapter 6.) for model–based road and 3–D object recognition tasks, performed in a vision system for car driver support (autonomous navigation) [KAS95a, KAS97, KAS98]. First a solution of the road border following task is defined, called the adaptive road recognition method. The method recognizes the number of road lanes and estimates the width and curvature of the road. It requires the tracking of ego-state, i.e. the camera's orientation and location and the ego–velocities. Next in this chapter, the adaptive recognition scheme is applied for a model-based recognition of 3–D vehicles moving in front of the camera car.

8.1 A VISION SYSTEM FOR DRIVER SUPPORT

The processing structure of an implemented image sequence analysis system under true camera motion is depicted in *Figure* 8.1. It consists of an application–independent module for 2-D image segmentation and visual motion estimation (bottom left scheme part), which is integrated with model-based road recognition/egostate tracking (the 2.5–D module) and 3–D object recognition modules (the 3–D module). The 2.5–D module obtains its input from the 2–D module (image segments) and supplies the 3–D object recognition module with current road hypotheses.

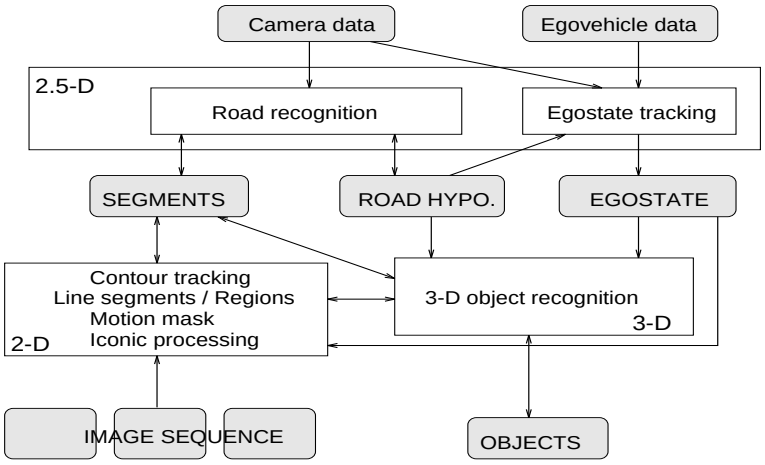


Figure 8.1 The processing structure of an implemented traffic scene analysis system under ego-motion.

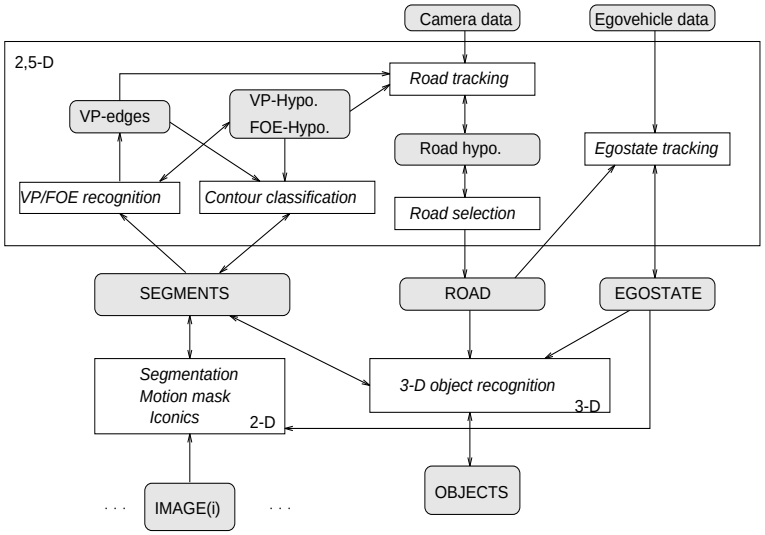
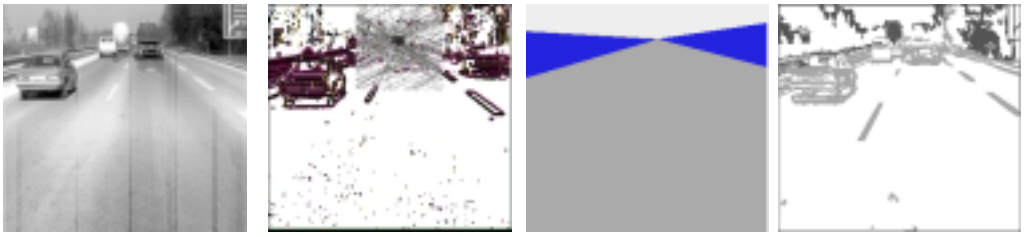


Figure 8.2 The 2.5–D module for road recognition and ego-state tracking, and its integration into a vision system for driver support [KAS98].

8.1.1 The 2,5-D module

The structure of the 2.5–D module is given in *Figure* 8.2. Road recognition requires that classified segments, the vanishing point and current ego-state data are already available.

Contour classification depends on previous detection of the *free road* region and the *vanishing point* in the image [KAS94a]. In the 2–D segmentation module a homogeneous free road region directly in front of the ego–car is detected and tracked leading to the free-road region estimation (*Figure* 8.3 (a)). This region determines the small-



(a) (b) (c) (d)

Figure 8.3 Geometry based segment classification: (a) the free-road region, (b) the vanishing point, (c) a three–class image decomposition, (d) a rule-based segment classification into road-, surround- and horizon-segments.

est road hypothesis extension in the image. Next, on the basis of the current vanishing point (VP) estimation (see chapter 7) (*Figure* 8.3 (b)) and the current free-road region estimation, the image is decomposed into three classes: "road", "surrounding area" and "sky" (*Figure* 8.3 (c)). Now the segments containing a number of "road" pixels are classified as "road" contours, the contours without such pixels but containing sufficient "surrounding" area pixels are classified as surrounding contours. The remaining contours are assumed to be placed above the horizon (*Figure* 8.3 (d)).

The estimation of the so called *ego–state* is of importance for both road recognition and 3–D object recognition. The ego-state contains the following relative camera position parameters: location relative to current road center plane B, location over the ground H, orientation parameters α and β, and the translational (longitudinal) and rotational ego–velocities V_c, ω_c. The orientation parameters of the ego-state are directly related to the vanishing point in the image. The longitudinal ego–velocity V_c is known if the speed measurement of the ego–vehicle is available. The ego–vehicle data streams are related to the speed $V_c(k)$ and steering angle $\delta_c(k)$, as well as the distance between the gravity center and the rear wheel axis L. This data is used for estimation of the rotational velocity of the ego–car $\omega_c(k)$. The dynamic trajectory of the ego–vehicle is approximated by a circular curve, where the tangential line of the trajectory is attached to the current position of the ego–car's center of gravity. For small $\delta_c(k)$ this angle can be computed as [KOL93]:

$$\delta_c(k) \simeq \frac{L}{R(k)}, \tag{8.1}$$

where $R(k)$ is the instantaneous trajectory radius. From $\omega_c(k) = \frac{V_c(k)}{R(k)}$ one finally concludes that

$$\omega_c(k) = \frac{V_c(k) * \delta_c(k)}{L}. \tag{8.2}$$

Although an approximation was applied, which is valid for a relatively large ego–trajectory radiu this kind of trajectory usually appears while driving along roads. We could also apply a more complex car motion model for the estimation of rotational ego-velocity (e.g. [KOL93]). In our tests it turned out that the quality of this simple curvature model is higher than the quality of approaches measuring the curvature directly in the image data. The height H is assumed to be known and constant. The

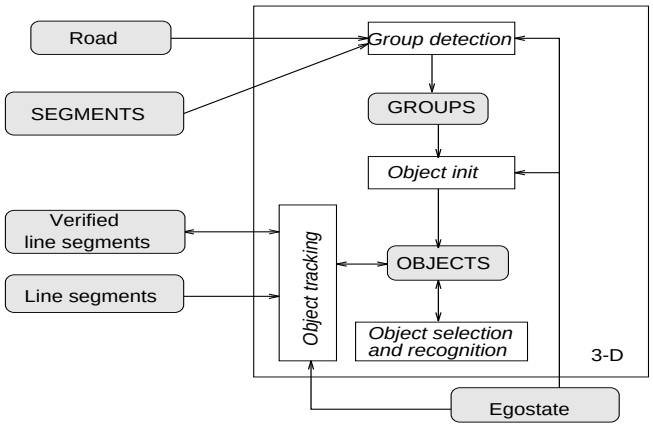


Figure 8.4 The processing structure of the 3–D object recognition module.

parameter B is a by-product of the road recognition step, locating the road's center line in the camera coordinate system.

Finally, the road recognition method itself consists of *road tracking* and *road selection* steps, i.e. it implements two steps of the general adaptive recognition scheme. Several road hypotheses can be generated which are characterized by different state vectors. Every road state consists of the road lane number T_i, the road width W_i, the camera position B_i and the road curvature ψ [KAS98]. During measurement line segments corresponding to the currently best vanishing point hypothesis are back-projected onto the road plane, and the analysis of the intersection densities between a cross section line, moving along the depth axis, and the back-projected segments leads to detections of probable road lane number and road center line (*Figure* 8.5(a)). Different road class hypotheses may be tracked in parallel, but for every image the best one is selected and supplied to other modules of the system.

8.1.2 The 3-D module

The 3–D module for model-based object recognition (*Figure* 8.4) consists of methods for segment group detection, object initialization, object tracking, object selection and object recognition. Let us note, that group detection is a model-based measurement step, which allows the initialization of a 3-D object hypothesis. With a given current road class hypothesis and ego-state (*Figure* 8.5(a)) appropriate road stripe objects are searched for in the image (*Figure* 8.5(b)). Stripe object recognition is reduced to a tracking case only. In contrast, competitive hypotheses of moving vehicle objects are allowed and they are tracked in parallel (*Figure* 8.5(c)) as long as a selection is not possible (*Figure* 8.5(d)). A more detailed recognition step using a more specialized vehicle model is performed after the selected object hypothesis changes to a "recognized object" status (*Figure* 8.5(e)).

(a) (b) (c) (d) (e)

Figure 8.5 Results of model-based object recognition: (a) road and ego-state are already given, (b) road stripe objects, (c) moving vehicle hypotheses, (d) selected consistent vehicle hypotheses, (e) detailed recognition of a particular object.

8.2 ROAD RECOGNITION

Related work, covering some aspects of the presented road recognition problem, has been performed since the late 80's. In the early stage the authors concentrated on the *road border tracking* (or *lane keeping*) problem and on real–time implementations of such solutions. In [FUJ90] a classification approach is proposed. With the help of a simple geometry model image regions are identified to be road parts. A neural network learning approach is proposed in [POM91, POM93]. Its purpose is a pixel–based classification of image regions into road parts under various lighting conditions. Other approaches are based on the detection of image edges. In [SCH92] such a road border following task is described. In [MOR90] a pure road border tracking method is proposed. A proper initialization should be made by hand. In [DIM92] a real–time implementation of the road border following task is described. Yet another approach is a feature tracking based one [KLU90].

Starting from linear road tracking the next problem was the detection of road curvature on the basis of image edges. There are several basic approaches known from the literature as far as road curvature estimation is concerned. The first one is a model–independent analysis of multiple vanishing points. For example, in [POL92] a method for vanishing point–based road curvature estimation is proposed. The image is divided into several horizontal cross sections. The detected image edges corresponding to road boundaries are grouped together if they belong to the same image section. Every group allows an estimation of some vanishing point. The curvature is estimated by analyzing the differences in locations of these vanishing points. The second approach consists of a model–based fitting of border segments to 2–nd order curves [SCH92]. Another approach tries to match a given curvature hypothesis with the bird's eye reprojection of the image [CHE93]. The fourth approach tries to fit differential geometry models into intensity histograms along both the horizontal and vertical axes. The important requirement for all these methods is good visibility of the distant road elements, i.e. the approaches will fail if many obstacles or objects exist in the scene. This means that a robust detection of the road curvature directly from the image data is possible only in the case of small occlusions of the road border.

Generally speaking, the above methods of road border following and road curvature estimation work well under some scene restrictions, i.e. if no obstacles occur in the scene, if no lane change is performed by the ego–vehicle and if no road class change occurs.

The recognition of other objects has been reduced so far to the detection of obstacles, i.e. to something rising above the ground in front of the vehicle.

Now the second development stage in the area of road recognition is under way, which means the design of recognition systems that not only track the known state but also make automatic adjustments to lane and road type changes [MAU96, JOC96]. They are also able to distinguish between a road lane marker and an obstacle [GRA93]. The combination of such systems with further developed traffic object recognition systems will result in relatively near future in computer vision applications like car driver assistance systems [WET94].

8.2.1 The road recognition procedure

The purpose of road recognition is the estimation of a road state vector. This is accomplished by a consecutive detection and recursive estimation of several competitive road hypotheses which is combined with the selection of the best road hypothesis. In our experiments up to three road hypotheses are tracked in parallel: R_2, R_3 and R_4. The state vector of one road hypothesis $R_i, (i = 2, 3, 4)$ (i means the number of lanes)

$$\boldsymbol{s}_{Ri}(k) = [W_i(k), B_i(k), \delta B_i(k), \omega_{Ri}(k)]^T \tag{8.3}$$

consists of the following parameters: the road width W_i, the position of the center line in ego–coordinates B_i and its change in time δB_i, and the rotational velocity of the road coordinate origin ω_{Ri} that is induced by the ego–car movement and the road curvature.

The road recognition procedure is described in detail in *Table* 8.1. For the recursive estimation of a single road hypothesis a linear recursive filter is applied. Again, a specific measurement judgement scheme was developed, which allows the use of the Kalman filter (as specified in section 6). In the remaining part of this section we concentrate on specific road detection, measurement judgement and hypothesis selection steps. A more detailed description of the road curvature detection method is also provided.

8.2.2 Detection and judgement of parameters W, B

The road model consists of the following data: (1) the minimum and maximum width of one road lane, and (2) the minimum and maximum width of one road stripe (an inner stripe or a border stripe). The initial road–to–camera transformation can be computed after the selection of the best vanishing point hypothesis. With known H and on the assumption - $B = 0$ - the angles α and β are derived directly from the VP point. The initial transformation is denoted as $EGO(-H, -\alpha(k), -\beta(k)))$.

From the previously detected VP lines a subset of line segments is selected that satisfy the current best vanishing point. Only those segments are selected, that point toward this best VP hypothesis (*Figure* 8.6(a, b)). These possible road stripe segments are back projected onto the road plane (due to the transformation $EGO(-H, -\alpha(k), -\beta(k)))$) (*Figure* 8.6(c, d)). The weights of back projected edges are also determined, depending on the length, the distance from the observer, and their elongation along the Z_w axis.

An accumulator vector is provided, that corresponds to a cross section of the road (perpendicular to the Z^c axis). The selected road line segments are projected onto the X^c axis, and the edge weights are added to the appropriate accumulator cells. Now a

A) *Initialization*
In the first three images the detection of measurements of three hypotheses only is performed. After the third image the three hypotheses are initialized with averaged measurements for the width and side position. The position change parameter is set to zero, whereas the rotational velocity is set to the current ego-car's rotational velocity. Thus for image $k_0 = 3$ the first estimations $\boldsymbol{s}^*_{Ri}(k_0)$ and its covariance matrices $\boldsymbol{P}^*_{Ri}(k_0)$ are computed.
A skip to cycle step B.9 follows.

B) For every next image $k > k_0$ the following cycle of processing steps is performed.
1. New *road region* detection and estimation.
2. *Direct measurement*
FOR each road hypothesis $R_i, (i = 2, 3, 4)$ DO:
– detect $\boldsymbol{m}_{Ri}(k), \boldsymbol{R}_{Ri}(k)$, where $\boldsymbol{m}_{Ri}(k) = [W_i(k), B_i(k)]^T$ is equivalent to a reduced state which consists of the road width and the ego–car's relative on–road position.
3. *Modification*
FOR each road hypothesis $R_i, (i = 2, 3, 4)$ DO:
– compute the gain $G_i(k)$
– modify estimated state $\boldsymbol{s}^*_{Ri}(k)$ and its covariance matrix $\boldsymbol{P}^*_{Ri}(k)$ appropriately.
4. *Measurement* of δB
FOR each road hypothesis $R_i, (i = 2, 3, 4)$ DO:
$\delta B_i(k) = \boldsymbol{s}^*_{Ri}(k) - \boldsymbol{s}^*_{Ri}(k-1)$.
This measurement evaluates the change between the previous and current B position.
5. *Modification* of δB.
6. *Measurement* of ω_R
FOR each road hypothesis $R_i, (i = 2, 3, 4)$ DO:
$\omega_{Ri}(k) = \boldsymbol{s}^*_{Ri}(k) - \boldsymbol{s}^*_{Ri}(k-1)$.
This dependent measurement means the detection of the road curvature ω_{Ri}.
7. *Modification* of ω_R.
8. Selection of the best road hypothesis.
9. Next state *prediction*
FOR every road hypothesis $R_i, (i = 2, 3, 4)$ DO:
– prediction of the state vector $\widehat{s}(k+1)$ and its covariance matrix $\boldsymbol{P}^+(k+1)$.
10. With $k \leftarrow k+1$ repeat the cycle.

Table 8.1 The road recognition procedure.

matching between a model grid M_i and the accumulator is performed for each road hypothesis R_i $(i = 2, 3, 4)$. The particular model grid M_i contains $i + 1$ elements. The best density grid with respect to the model grid is searched for. One individual measurement consists of the best $i + 1$–element grid, i.e. a single measurement for one road grid hypothesis M_i (*Figure* 8.7).

The selection of best measurement is directed by a quality function, which is computed by a weighted sum of accumulator elements that are covered by the particular grid. Some large penalty scores are also added to the judgment if one of the following situations occurs: some inner grid elements are missed (a null match for some element inside the road grid), the grid is partially outside the allowed maximum width section,

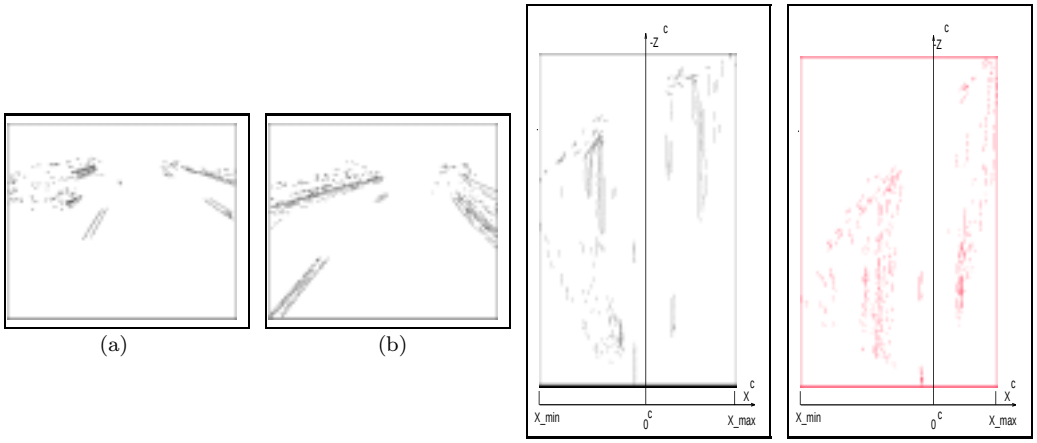


Figure 8.6 Line segments, that have been selected for road recognition (a, b) and their top view back-projections onto the road plane (c, d). Linear road case in images (a,c) and curved road case in images (b,d) [KAS98].

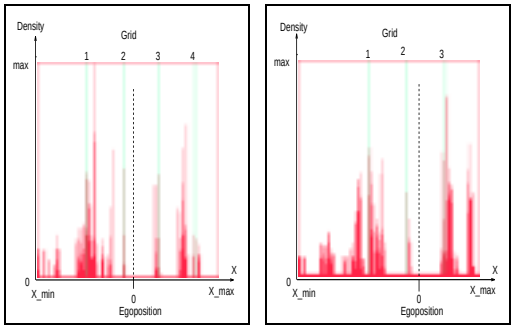


Figure 8.7 The detected road grids (one grid element corresponds to the location of a road stripe on the X^c–axis): for a 3–lane road (left) and a 2–lane road (right).

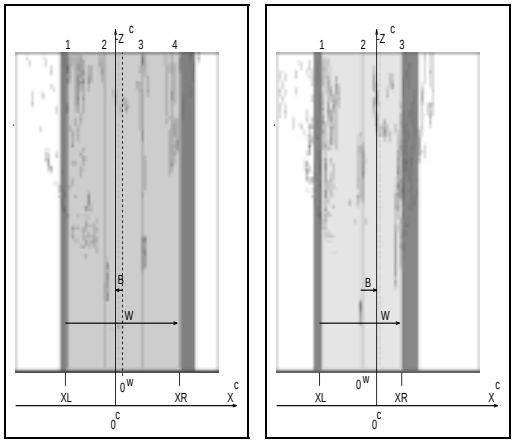


Figure 8.8 Top view of the estimated and selected road (width $W = X_R - X_L$, side position B and stripe number) for a straight road (left) and for a curved road (right).

the grid is fully inside of the minimum width section (this section is induced by current road region in front of the ego–vehicle).

As the best location of a single grid model is directly given in X^c–coordinates, the road width and the location of the camera (for a given grid hypothesis M_i) are immediately given.

Measurement judgement

The success of stabilizing each road parameter set $\boldsymbol{s}_{Ri}, (i = 2, 3, 4)$ during adaptive filtering mainly depends on the proper estimation of the measurement variances in $\boldsymbol{R}_i$.

The author found experimentally that a measurement variance, which depends solely on the above mentioned quality of measurements (quality of a grid model to density accumulator match)), is not reliable. Instead of working with measurements in a single image we apply a short time storage in which the current and up to $N-1$ ($N=3-5$) corresponding measurements in the last previous N images are stored. The variances of individual parameter measurements in this sequence of N images are combined with the quality of the current single measurement in order to provide current measurement variance of this parameter. A heuristic additive penalty score may modify this judgement, if no corresponding measurement is available in some image.

8.2.3 Road hypothesis selection

The quality evaluation of some road hypothesis $\boldsymbol{s}^*_{Ri}(k)$ depends directly on its covariance matrix $\boldsymbol{P}^*_{Ri}(k)$. In the simplest form a weighted sum P_i of the variances on the diagonal of matrix $\boldsymbol{P}^*_{Ri}(k)$ is calculated, or simply the variance of the parameter $B_i(k)$ is used. The selection of the best road hypothesis from a pair $\boldsymbol{s}^*_{Ri}(k)$, $\boldsymbol{s}^*_{Rj}(k)$ $(i \neq j, i, j = 1, 2, 3)$ is based on the relation between these combined variances P_i and P_j normalized over the road widths W_i and W_j, respectively. The following rule has been applied for the selection of the better hypothesis of the two hypotheses:

$$\text{IF}(P_j \leq \frac{W_j^2}{W_i^2}\, P_i) \text{ THEN select } \; T_j \text{ ELSE select } T_i. \tag{8.4}$$

The current best road hypothesis is used for ego–state tracking and for object recognition (*Figure* 8.8).

8.2.4 Road curvature detection

Let us define the road *curvature* at image k ($\mathcal{C}_R(k)$) as the direction change of the axis $Z^w(k)$ of the road coordinate system. The important requirement for known methods of road curvature detection is good visibility of the distant road elements, i.e. they will fail if many obstacles or objects exist in the scene. This means, a robust detection of the road curvature directly from the image data is possible only in the case of small occlusion of the road border. Our image data for road curvature consists alternatively from the VP point or the B–parameter estimations. In this estimation we use as many appropriate segments as possible and compute one instantaneous value of a combined entity (VP or B). Hence, we are not searching for a lateral correspondence between particular segments, which would be strongly affected by many obstacles and moving objects that are hiding the road boundaries in an unpredictable way.

Additionally, curvature measurement from image data is highly sensitive to noise and errors of digitalization, as it deals with very small angular values. In order to minimize these errors the measured entities should have some significant values. During experiments the author found that some short–term memory of an ego–car's direction and ego–velocity estimations, which allows one to work with "larger" angle change values, significantly increases the signal-to-noise ratio.

The VP-based method

The road curvature is detected on the basis of results of vanishing point tracking and of the knowledge of the rotational ego–velocity ω_c. At first the angular change of the ego–vehicle's direction Γ_S in the last n time intervals τ is approximated by:

$$\Gamma_S(k) = \sum_{i=k-N}^{k} \omega_c(i), \text{ where } N = \begin{cases} n & \text{, for } k \geq n; \\ k & \text{, for } k < n. \end{cases} \tag{8.5}$$

In this time the relative-to-road direction β has changed from $\beta(k-N)$ to $\beta(k)$, i.e. by: $\Delta\beta(k) = \beta(k) - \beta(k-N)$. Thus, the current road curvature $\mathcal{C}_R(k)$ is approximately given by:

$$\mathcal{C}_R(k) = \frac{\Gamma_S(k) - \Delta\beta(k)}{N\tau}. \tag{8.6}$$

The egoposition-based method

The second method of curvature estimation explores the tracking results of position B and the knowledge of rotational (ω_c) and translational ego–velocities (V_c). Again, let $\Gamma_S(k)$ denote the ego–car direction change in N time intervals. Let $\Gamma_V(k)$ be the approximated distance covered by the ego–car in this time:

$$\Gamma_V(k) = \sum_{i=k-N}^{k} V_c(i). \tag{8.7}$$

Hence the direction of this movement is approximately given by:

$$\gamma(k) = \arcsin\left[\frac{B(k) - B(t-N)}{-\Gamma_V}\right]. \tag{8.8}$$

The current road curvature is now approximated by:

$$\mathcal{C}_R(k) = \frac{\Gamma_S(k) - \gamma(k)}{N\tau}. \tag{8.9}$$

8.3 3-D VEHICLE RECOGNITION

By adapting control techniques a dynamic system with feedback for 3-D object tracking was first defined in [DIC88]. Applications of this 3–D model based object tracking approach were originally developed for flight and aerospace applications. It was later moved to applications in road scene analysis and for autonomous navigation in robotics. This approach is especially suitable for single object tracking if an exact object model is available, i.e. exact shape and dimensions.

A model based method for automatic satellite docking at space stations is proposed in [WUE88]. The image measurement contains only a small number of significant points,

which allows the recovery of the 3–D position of the station in space. The rotations are restricted to one plane and only the relative camera motion has to be recovered. In [GEN92] the same application field is assumed as in the previous paper. A tracking method for known objects in space is described, where the number of degrees of freedom of object motion is increased to six. The object can freely rotate and the camera is stationary. The measurement contains either points or edges, whereas the objects are defined either by wire frames or by points.

In [KOL93] an approach to single vehicle tracking on the ground plane with a stationary camera is described. A parametric shape model (with 12 lengths) is applied that enables the modeling of different vehicle types. It is assumed that all the recognized objects are moving forward. The model edges are projected into the image plane and a match between them and the image edges is performed.

8.3.1 The object model

In our approach the model-based single object tracking task is extended to the task of multiple object recognition. An object hypothesis is specified by its class and its state vector $\boldsymbol{s}$:

$$\boldsymbol{s}(k) = [\boldsymbol{s}^d(k), \boldsymbol{\xi}(k)], \tag{8.10}$$

where $\boldsymbol{s}^d$ is the *trajectory vector*, that specifies the object position and motion on the ground plane, and $\boldsymbol{\xi}$ is the *shape vector*.

Trajectory vector

A general trajectory vector of a rigid object, moving in 3–D space, consists in general of 6 localization parameters (3 translation and 3 rotation components) and 6 motion parameters (3 translation velocities and 3 angular velocities). In this work the object motion and ego-car motion are restricted to the road plane. Thus two trajectories are considered: the camera vehicle trajectory and the moving object trajectory, as shown in *Figure* 8.9. The two trajectories are approximated locally by circular arcs in the road plane with center points C^c and C^o respectively.

A *trajectory* vector $\boldsymbol{x}(k)$ at time point t_k is a five–dimensional vector

$$\boldsymbol{s}^d(k) = [p_X(k), p_Z(k), \Theta(k), (V(k), \omega(k))]^T, \tag{8.11}$$

that consists of the position $(p_X(k), p_Z(k))$ and orientation Θ of the translational motion, and of the magnitudes $V(k)$ and $\omega(k)$ of translational and angular velocities, respectively. The parameters of this vector can be of different semantics depending on the object class. The virtual trajectory vector of a *stationary object* on the road plane is given as:

$$\boldsymbol{s}^d_S(k) = [p^c_{SX}(k), p^c_{SZ}(k), \Theta^c_S(k), V_S(k), \omega_S(k)]^T \tag{8.12}$$

Let us note, that the first three parameters are expressed in relation to the ego-camera coordinates (the *ego-coordinates*). The trajectory vector of a *moving object* contains the following components:

$$\boldsymbol{s}^d_B(k) = [p^c_{BX}(k), P^c_{BZ}(k), \Theta_B(k), V_B(k), \omega_B(k)]^T. \tag{8.13}$$

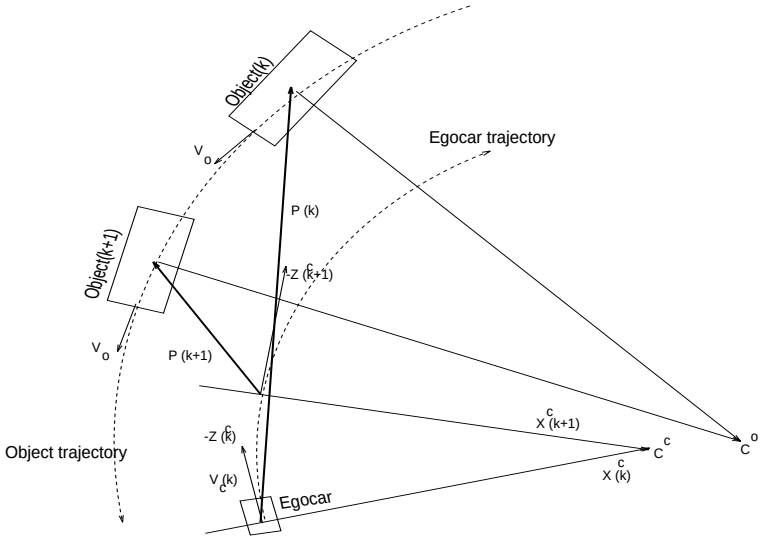


Figure 8.9 The moving object trajectory and the ego-trajectory on the road plane.

Now only the location is specified in the ego-coordinates, but the direction is given in relation to the global world coordinate system.

Shape vector

The parameters of the shape vector of an object hypothesis are the width $Width$ and several parameters κ_i:

$$\boldsymbol{\xi}(k) = [Width(k), \kappa_1(k), \kappa_2(k), ..., \kappa_j(k)]^T. \tag{8.14}$$

The number and meaning of the components κ_i depends on the object class and on the specialization of the shape representation. Model shapes for three object classes were implemented: stationary road *stripe* and *obstacle* object, and moving *vehicle*. The first form is inherently planar (in reality some constant height is assumed) and thus it is determined by two shape parameters only – width and length (i.e. $Width$ and $length_1$). For an obstacle object a 3–D bounding box only is specified, i.e. three shape parameters are necessary: $Width, length_1$ and $height_1$. Three specialization levels for vehicle objects are considered (*Figure* 8.10). The general vehicle shape at the *object* level consists of two boxes of equal width, i.e. five shape parameters are necessary: $Width, Length_1$, $Length_2, Height_1$ and $Height_2$. The vehicle shape vector is the most general one, whereas the $\boldsymbol{\xi}_S$ (stripe vector) and $\boldsymbol{\xi}_U$ (obstacle or surrounding object) vectors are specialized vectors.

Between the general model and the first specialization level, called *object_shape*, there is a difference in model activation and in the number of independent shape parameters. In the most general case the 3–D object hypotheses are repeatedly generated and a matching between previous and new hypotheses takes place. The two shape parameters $height_2$ and $length_2$ are dependent and are assumed to be related to the independent

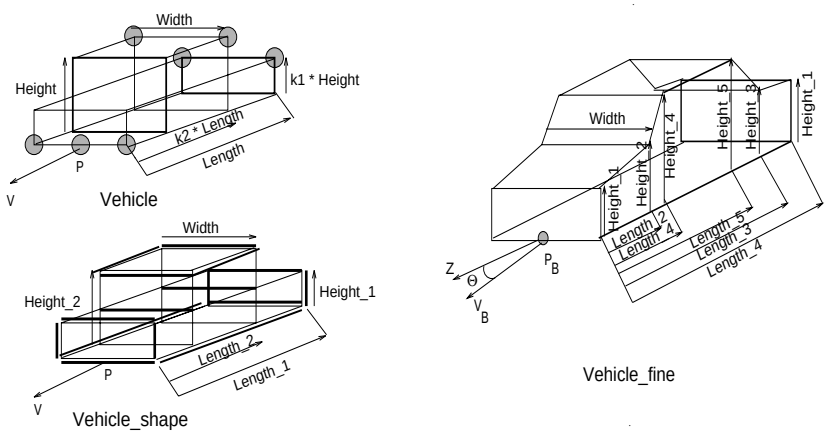


Figure 8.10 Three object specialization levels of the *vehicle* class.

parameters $height_1$ and $length_1$ by a constant. In the second case a goal oriented 2–D measurement takes place for each previous object hypothesis. The matching takes place on the level of 2–D edges. The object tracking step for the current image is done before the segment detection step. The new edge segments that support a tracked object belong to the *verified segments* and they are filtered out from further processing.

The models of the second specialization level, called *vwehicle_fine*, work with the edge based 2–D matching alone. Their goal is a detailed classification of the vehicle hypothesis, if the image dimension of the object allows it. This specialization level can contain such shapes like *car* or *truck*. The object shape is specified here by an extended parameter vector. As far as the activation of this model specialization level is concerned it differs from the *object_shape* level only by its greater detail level, but the measurement step and recursive estimation are similar.

Other stationary or moving objects that do not belong to the vehicle class are classified into *obstacle* (if they are inside the road) or *surrounding object* (if outside the road). The principle of vehicle recognition is the same as for example for the *bi-cyclist* recognition, if a volume model of a person and bicycle can be defined.

Hypothesis selection

An object hypothesis is either in its *tracking* or in one of its *recognition* phases. A tracking phase is given if the tracking time of this hypothesis is lower than T_{min} or its variance is greater than the $maximum\_var$. Otherwise the hypothesis is in one of its three recognition phases. These phases are closely related to the use of object specialization levels (from *Shape* over *Fine* to *Type*).

The consistency test takes place between pairs of hypotheses. If the tracking times of two competitive hypotheses are both larger than T_{min} or one of them is in the recognition phase (i.e. its tracking time is longer than some threshold time T_{min} and its combined variance is lower than some $Maximum\_Var$), then the consistency test among them is performed.

1. VP-based depth estimation of $p_Z^w(0)$ assuming that $p_Y^w = 0$.
   In *Figure* 8.11 (a) one contour group with its bounding box is presented. The surface which projects to the first contour of the group is touching the road plane. This implies that the back-projections $P^w, P1^w, P2^w$ of the points $P, P1, P2$ satisfy the constraint $Y^w \simeq 0$. With this restriction the depth can be estimated by back-projection.
2. A correction of the depth, if $p_X^w(0)$ is outside the road space.
3. For road stripes the translational motion magnitude $V(0)$ is set to the ego-motion; for vehicles this depends on their positions on the road (the lane number).
4. The direction angle $\Theta(0)$ is detected from the boundary boxes.
   As seen in *Figure* 8.11 (a) there are two bounding boxes provided for one contour group: the overall box and the included first contour box. The conjunction of these two detected image bounding boxes with the model–based restrictions about the length–to–width and height–to–width ratios makes the direction hypothesis possible.
5. The obstacle objects are assumed to be vertically elongated.
6. The rotational velocity $\omega(0)$ is determined by the current road curvature.
7. The shape parameters occur from projecting the image data back at assumed depth and direction.

Table 8.2 The procedure of object hypothesis initialization.

8.3.2 Object initialization

By the initialization of an object (or generation of an object hypothesis) we mean the detection of a segment *group*, the detection of the object class, and the initialization of an appropriate *state vector* of the object hypothesis.

Based on the classified segments (moving or stationary segments; road, obstacle, horizon or surrounding segment) and the current camera to road transformation, a segment grouping is performed in an expected 3–D space over the road. The contour grouping step aggregates contours from the same class together, although specific interclass groupings are allowed. Three classes of group are possible: *road stripe group*, *obstacle group* and *vehicle group*.

The segments are first classified into horizontal, vertical and others. For example, the search for an "obstacle" group starts with a "road", non–stationary and horizontal segment on the bottom of the image. It looks for neighbors located above it in the image. As the first segment is assumed to be placed on the road a depth value can be hypothesized. Now the real dimensions of the neighbor segments can be initialized also. The grouping is performed in a model dependent 3–D bounding box over the start segment. This box is always modified after adding a new segment to the current group. The group is completed if it cannot be extended, because of lack of contours or violation of the 3–D model restrictions (i.e. width, height, length or width–to–height ratio).

For each new group an object hypothesis is initialized. The *state vector* is derived from the 2–D features of the detected group and the 3–D model–dependent restrictions, by applying the knowledge about the camera–to–road transformation, ego-motion, and road origin trajectory (see *Table* 8.2).

8.3.3 Prediction – state transition

Two state transition functions are defined below: for stationary stripes and obstacles $\boldsymbol{F}_S(\boldsymbol{s}_S)$, and for moving objects $\boldsymbol{F}_B(\boldsymbol{s}_B)$. For other object types, like moving obstacles or stationary surrounding objects, similar state transition functions can be defined.

The full form of the state transition function for road stripe state vector is given as (on the right side the k-s were omitted):

$$\boldsymbol{s}_S(k+1) = \begin{pmatrix} p^c_{SX}(k+1) \\ p^c_{SZ}(k+1) \\ \Theta^c_S(k+1) \\ V_S(k+1) \\ \omega_S(k+1) \\ \boldsymbol{\xi}_S(k+1) \end{pmatrix} = \begin{pmatrix} p^c_{SX}\cos(\omega_S\tau) + p^c_{SZ}\sin(\omega_S\tau) - \frac{V_S}{\omega_S}(\cos(\omega_S\tau) - 1) \\ -p^c_{SX}\sin(\omega_S\tau) + p^c_{SZ}\cos(\omega_S\tau) + \frac{V_S}{\omega_S}(\sin(\omega_S\tau)) \\ \Theta^c_S + \omega_S\tau \\ V_S \\ \omega_S \\ \boldsymbol{\xi}_S \end{pmatrix} \quad (8.15)$$

The sum of the ego-motion and the object motion can be observed only (*Figure* 8.9). Let the unknown object motion be denoted by V_o and $\dot{\theta}_o = \omega_o$. The transition function for a moving object state $\boldsymbol{s}_B(k+1)$ is given as:

$$\begin{pmatrix} p^c_{BX}(k+1) \\ p^c_{BZ}(k+1) \\ \Theta_B(k+1) \\ V_B(k+1) \\ \omega_B(k+1) \\ \boldsymbol{\xi}_B(k+1) \end{pmatrix} = \begin{pmatrix} p^c_{BX}\cos(\omega_c\tau) - p^c_{BZ}\sin(\omega_c\tau) + \frac{V_c}{\omega_c}(C-1) + \frac{V_B}{\omega_B}(C2 - C1) \\ p^c_{BX}\sin(\omega_c\tau) + p^c_{BZ}\cos(\omega_c\tau) - \frac{V_c}{\omega_c}S + \frac{V_B}{\omega_B}(S1 - S2) \\ \Theta_B(k) + \omega_B(k)\tau \\ V_B(k) \\ \omega_B(k) \\ \boldsymbol{\xi}_B(k) \end{pmatrix} \quad (8.16)$$

$$\text{with:} \qquad S1 = \sin(\Theta^c_B - \omega_c\tau), \qquad C1 = \cos(\Theta^c_B - \omega_c\tau),$$
$$S2 = \sin(\Theta^c_B - \omega_c\tau + \omega_B\tau), \quad C2 = \cos(\Theta^c_B - \omega_c\tau + \omega_B\tau).$$

8.3.4 State modification

The state projection function is not directly dependent on the motion parameters V and ω. An independent measurement of the parameter Θ and the lengths l_1 and l_2 cannot be achieved. This problem occurs both for the 2–D and 3–D measurement. This is the reason why two sequential measurement and modification steps are necessary during object tracking. At first, after the measurement in the image, the new estimation of a reduced state ($\boldsymbol{s}^*_R(k) = \boldsymbol{s}^*(k) - [V, \omega)]^T$) takes place. From the vector $\delta\boldsymbol{s}^*_R(k) = \boldsymbol{s}^*_R(k) - \boldsymbol{s}^*_R(k-1)$, new synthetic measurements of $(V(k), \omega(k))$ are computed. From these measurements the border values for the dependent parameters occur. This is a second synthesized measurement:

$$\boldsymbol{s}_m(k) = [\Theta_m(k), l_{1m}(k), l_{2m}(k)]^T. \quad (8.17)$$

These two synthetic measurements lead to the second modification. The sub-state

$$\boldsymbol{s}_E(k) = [V(k), \Theta(k), \omega(k), l_1(k), l_2(k)]^T \quad (8.18)$$

1. A modification of the reduced state $\boldsymbol{s}_R(k)$ and of its covariance matrix $\boldsymbol{P}_R(k)$;

$$\boldsymbol{s}_R^*(k) \quad = \quad \boldsymbol{s}_R^+(k) + \boldsymbol{K}(k)\big\{\boldsymbol{m}(k) - \boldsymbol{H}(k)\boldsymbol{s}^+(k)\big\}, \tag{8.19}$$

   $\boldsymbol{P}_R^*(k)$ according to eq. (6.16) and $\boldsymbol{K}(k)$ according to equation (6.17).
2. A synthetic measurement of the motion parameters and the motion dependent border values for the dependent parameters, on the basis of the difference: $\boldsymbol{s}_R^*(k) - \boldsymbol{s}_R^*(k-1)$.
3. A modification of the sub-state $\boldsymbol{s}_E^*(k)$ on the basis of prediction $\boldsymbol{s}_E^+(k)$, the synthetic measurements V, ω and $\boldsymbol{s}_m(k)$.

Table 8.3 The two-step 3–D object modification procedure.

is modified next. The *two step modification* procedure is summarized in *Table* 8.3. Due to the non-linearity of state transition and projection functions, the problem is linearized at each time k by using the instantaneous values of these functions (given by their Jacobi-matrices). Despite many tests a reliable measurement judgement scheme could not be found. Hence, the state covariance matrix $\boldsymbol{P}$ and gain matrix $\boldsymbol{K}$ estimations are based on the tracking error, and they are computed according to equations (6.16) and (6.17).

8.3.5 State projection and measurements

The projection function $\boldsymbol{H}(\boldsymbol{s})$ depends on the complexity of the object model. Two types of measurement are distinguished:

1. the *3–D measurement* – a repeated object initialization for a full new image takes place. The subsequent modification of the hypothesis is equivalent to an object–to–object–update.
2. the *2–D measurement* – such image segments are searched for, that fit the model based expectations related to the current hypothesis. The modification of the object hypothesis is controlled by the difference between the predicted model projection and the detected 2–D segment set.

The 3–D measurement is applied during general vehicle tracking (for the model level: *vehicle*), whereas the 2–D measurement is needed for tracking more specialized objects (*vehicle_shape* and *vehicle_fine*). During the update of the object state, the predicted model features (3–D bounding box or model edges) are projected onto the image and they are matched with the new image features (groups or edges). A successful match implies that a new measurement is available and the 3–D object state can be updated.

Let us concentrate on the 2–D measurement required for the *vehicle_shape* level. The matching of model features with the next image features can be processed in two ways: the measured points are derived from contour groups or from line segment groups (*Figure* 8.11). In both cases the modification is based on the differences between projected model points and significant points of the measured data group. The projected model points are matched against the measured points from the vector:

$$\boldsymbol{m}(k)^T = [K_{minx}, K_{maxx}, G_{Cx}, G_{Cy}, G_{minx}, G_{maxx}, G_{miny}, G_{maxy}, K_x, K_y]^T \tag{8.20}$$

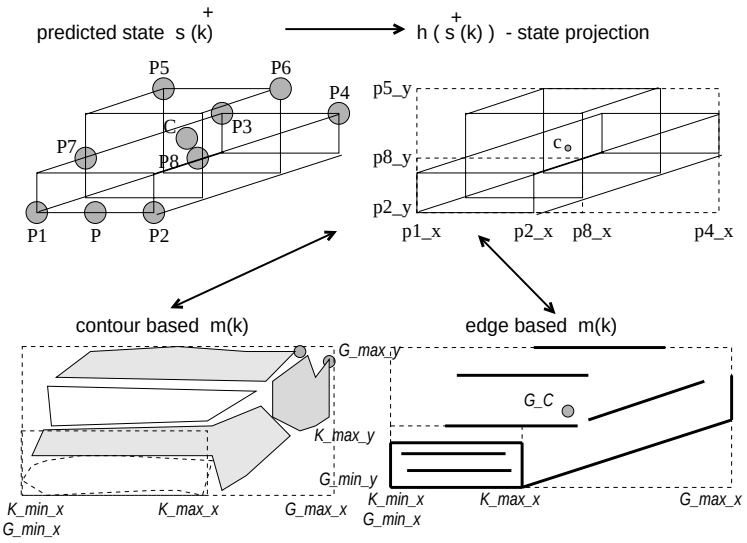


Figure 8.11 The principle of a 2–D measurement (point vector matching): grouping contours (left), grouping edges (right)

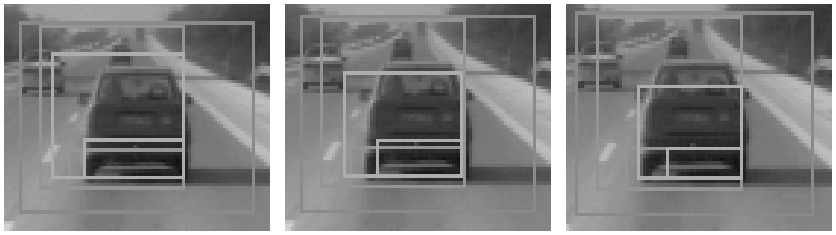


Figure 8.12 Adaptation of the measured segments and the object hypothesis onto the real shape in consecutive images (the outer double box denotes the image search area, the smaller one the object prediction and the bright one comes from the measured data).

Thus the 2–D measurement vector contains the x–components of the lower bounding box (K_{minx}, K_{maxx}), the gravity center $G = (G_x, G_y)$, the overall bounding box and the front location (K_x, K_y) of the smaller object box (*Figure* 8.11).

In the simplest case of edge based measurements, the objects are defined by 5 edge segments of a 3–D bounding box. These edges are projected onto the image plane and matched against the new image edges. The edge segments are usually defined by the location of their center points $C_i (i = 1, ..., 5)$, their direction angle versus the x–axis ϕ_i and their lengths $l_i (i = 1, ..., 5)$. The searched edge group should contain in a perfect situation five edge segments: $\{[C_{ix}(k), C_{iy}(k), \phi_i(k), l_i(k)] | (i = 1, ..., 5)\}$. In the case of more detailed vehicle models, more than five edges are searched for, but the projection principle is identical with the above scheme. The projection function is specified in detail in *Table* 8.4.

1. *Selection of model points.*
   In accordance with the direction angles of the object's length axis Z relative to the Z^c axis (of the ego-coordinates), i.e. $\Theta^c_B(k)$ (for moving objects) or $\Theta^c_S(k)$ (for stationary objects), and depending on the object type, the currently visible model points from the set $\{P1, P2, P3, P4, P5, P6, P7, P8G\}$ are determined.
2. *Computation* of model points in ego-coordinates.
   The current position of (in step 1) selected model points is determined as a function of the current state $\boldsymbol{s}(k)$, for example:

$$P1^c_X = p^c_X - \frac{Br}{2}\cos(\Theta)\ ; P1^c_Z = p^c_Z + \frac{Br}{2}\sin(\Theta)\ ; P1^c_Y = 0; \tag{8.21}$$

   where Θ is equal to $\Theta^c_B(k)$ or $\Theta^c_S(k)$ depending on the object type.
3. *Projecting* the ego-coordinates onto the image plane.
   The coordinates from step 2 are transformed into camera coordinates first and next they are projected onto the image plane. The complete transformation from the ego-coordinates (P^c_X, P^c_Y, P^c_Z) of a point P to the image coordinates (p_x, p_y) is as follows:

$$p_x = -F_x \frac{P^c_X}{-\sin(\alpha)P^c_Y + \cos(\alpha)P^c_Z + H\sin(\alpha)} \tag{8.22}$$

$$p_y = -F_y \frac{\cos(\alpha)P^c_Y + \sin(\alpha)P^c_Z - H\cos(\alpha)}{-\sin(\alpha)P^c_Y + \cos(\alpha)P^c_Z + H\sin(\alpha)} \tag{8.23}$$

The measured point coordinates from $\boldsymbol{m}(k)$ should correspond to the projected values. In the case of "top left" view there are the following correspondences:

$$p1_x \leftrightarrow (K_{minx} = G_{minx}),\ (p2_x, p2_y) \leftrightarrow (K_{maxx}, G_{miny}), \tag{8.24}$$

$$MAX(p3_y, p5_y) \leftrightarrow G_{maxy},\ p4_x \leftrightarrow G_{maxx},\ p7_y \leftrightarrow K_{maxy}. \tag{8.25}$$

If the "top right" view is selected the correspondences are as follows:

$$p2_x \leftrightarrow (K_{maxx} = G_{maxx}),\ (p1_x, p1_y) \leftrightarrow (K_{minx}, G_{miny}), \tag{8.26}$$

$$MAX(p4_y, p6_y) \leftrightarrow G_{maxy},\ p3_x \leftrightarrow G_{minx},\ p8_y \leftrightarrow K_{maxy}. \tag{8.27}$$

Table 8.4 The vehicle object projection function $\boldsymbol{H}(\boldsymbol{s})$ in case of 2-D measurements.

In *Figure* 8.12 an example of vehicle object hypothesis adaptation onto the real vehicle shape in a 3-image sequence is given. Let us also note, that due to object state change the search area for the next image measurement is changed accordingly.

8.4 TEST RESULTS

The vision system was tested on several image sequences with 125 – 500 non–interlaced images of resolution 351x283x8 bit each. There were images of motor-way scenes and federal road scenes provided, with both linear and curved road elements. A varying number of cars and trucks (usually 3–5) were visible in the images. The projection conditions were determined by the use of a camera with the relation of the focal

	H_2: two lane hypothesis				H_3: three lane hypothesis				Real values		
	B_2 [m]	W_2 [m]	E_2 [m^2]	Select H_2	B_3 [m]	W_3 [m]	E_3 [m^2]	Select H_3	Type	B [m]	W [m]
1	-0.783	6.97	0.0591	0	-0.858	10.34	0.0844	123	3	-0.83	11.00
2	1.107	7.07	0.0371	22	2.891	10.34	0.0191	101	3	3.0	10.4
3	1.389	6.86	0.0105	112	0.623	9.47	0.1014	11	2	1.53	6.58
4	1.105	7.00	0.0051	102	0.887	10.07	0.0832	21	2	1.36	6.9
5	1.375	6.86	0.0419	23	1.076	9.70	0.0640	100	2	1.20	7.7

Table 8.5 Parallel tracking of two road hypotheses and selection of the best one: road widths W_i, positions B_i and their estimation variances P_i (normalized over single lane width) in 5 image sequences.

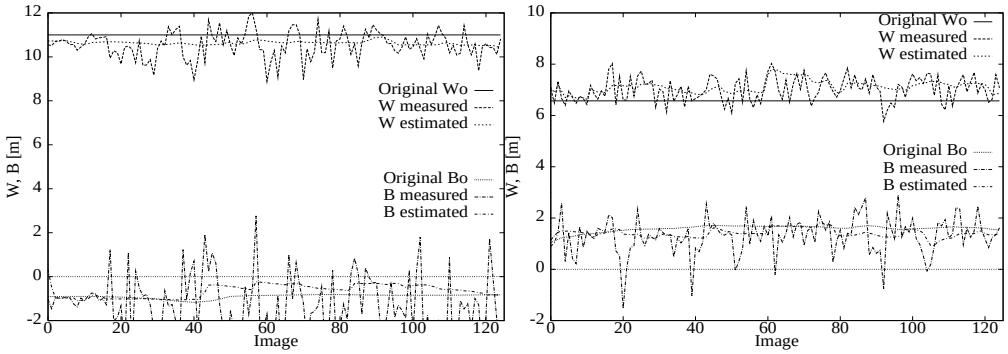


Figure 8.13 Individual measurements and estimated values of the recognized road parameters in two image sequences: linear road (left), curved road (right).

length to pixel width of 708 (after sub–sampling) and with the height position over the road of approx. 1.70[m]. We claim, that the quality of image analysis results is sufficient if they (1) reach the same stability as the external measured data or (2) they follow exactly some reference data, measured manually in the image or scene.

8.4.1 Road recognition

Test results of the parallel tracking of two competitive road hypotheses in 5 image sequences, containing 125 images each, are summarized in *Table* 8.5. The proper hypothesis has been selected nearly all the time. The wrong decisions in sequences No. 3, 4 and 5 occurred in situations, where several moving vehicles occluded the border for a relatively long time.

Now we compare the measured and estimated values of B and W given in the selected road hypothesis with the real reference data. The original road type Ti (2–lanes or 3–lanes) in each image sequence was obviously known. The reference values of road width $W_o(k)$ and the original camera position $B_o(k)$, relative to the road center line, were measured manually in the back projected image that was obtained during the analysis. On the basis of two examples in *Figure* 8.13 it can be observed that the error of road width estimation was below ± 0.9 m, whereas the error of ego–position estimation was below ± 0.4 m. The estimated values were of good quality, although the measured values sometimes contain large errors.

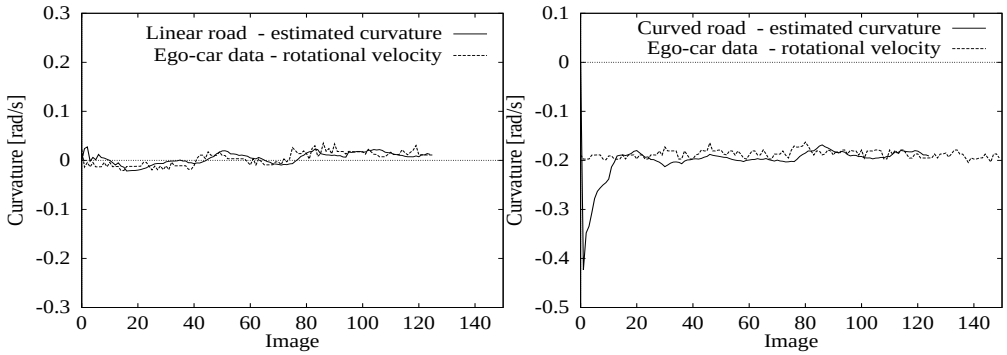


Figure 8.14 Road curvature estimation in linear (left drawing) and curved (right drawing) road sequences, obtained by the VP–based method.

The results of road curvature estimation in two image sequences are provided in *Figure* 8.14. The first sequence maps a straight road scene (the ego–vehicle is driving nearly straight ahead),whereas the second one maps a curved road scene (the ego–vehicle is driving a right curve). Exactly speaking we have estimated the road's rotational velocities, if the origin of road coordinates is moving in accordance with the ego–vehicle position. Both methods of curvature estimation, the VP-based method and the B-based one, gave similar results. Let us compare these test results with the quality of external data streams. From the translational ego–velocity and the steering angle streams the required parameters of the ego–state are directly computed, i.e. the rotational ego-velocity (*Figure* 8.15) and the ego-car direction in world coordinates (*Figure* 8.16). Small variations of the measured data can be observed, but nevertheless this data is of very good stability. Now it is clearly seen that the estimated road velocities follow the rotational velocities of the ego–vehicle very well. This means that in both image sequences – linear or right curve – a stable position of the ego–car relative to the road's center was computed. This was really true in considered image sequences as the ego–car was driven inside one road lane.

In the presented experiment it was started with the curvature estimation immediately in the first image in order to demonstrate how fast after system initialization the proper and stable estimations of curvature are achieved. Of course in practice one should start the road curvature estimation only after the basic estimations of VP or B are sufficiently stable.

8.4.2 3–D object recognition

The quality of vehicle recognition is verified by visual samples given in *Figure* 8.17, and by the tracking and recognition rate data summarized in *Table* 8.6. In the first scene six moving vehicles are visible, although one of them is in the half number of images only. Four vehicles were very well detected and they were tracked in 95–100 % of the images. The image projections of these vehicles are from the interval of $20 \times 20[pixel^2]$ – $50 \times 70[pixel^2]$. The only partly visible vehicle, whose image size is about $10 \times 12[pixel^2]$, was detected on average in every third image only. Similar recognition quality was ob-

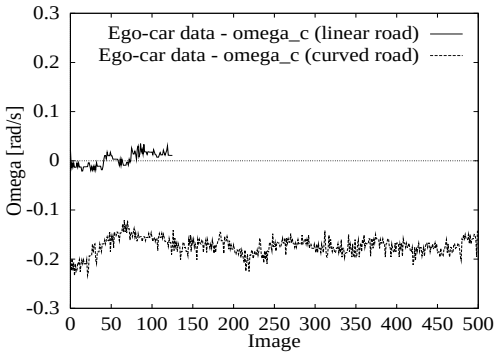


Figure 8.15 Ego-car's rotational velocities in two image sequences.

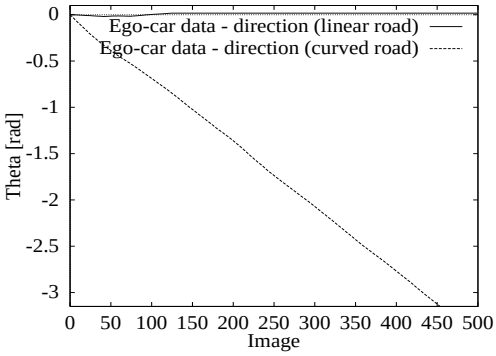


Figure 8.16 Heading directions of the ego–car in world coordinates in two image sequences.

Object	Vis	Gen	Gen/ V(%)	Tr- ack	Tr./ V(%)	Rec og.	Rec/ V(%)
l. car	125	125	100.0	125	100.0	122	97.6
bus	125	122	97.6	125	100.0	120	96.0
tanker	125	105	84.0	88	70.4	35	28.0
truck	125	123	98.4	121	96.8	65	52.0
r. car	62	51	82.2	50	80.6	40	64.5
far car	125	37	29.6	28	22.4	17	13.6

Table 8.6 Qualitative evaluation of vehicle recognition results in one image sequence. *Vis (V)* means the number of images in which an object was visible, *Gen* – image number when at least one hypothesis for given object was generated, *Track* – the time a hypothesis was in the tracking phase, *Rec* – the time a hypothesis was in the recognized phase.

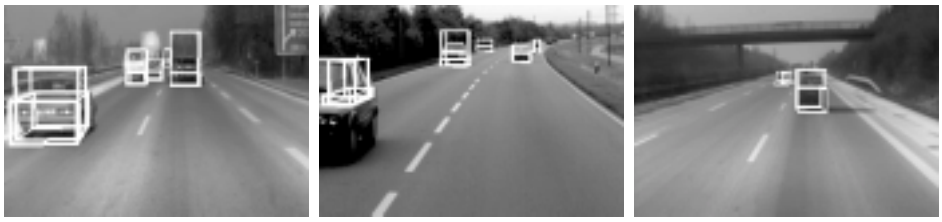


Figure 8.17 Examples of selected vehicle hypotheses in three image sequences [KAS97].

served for the other four sequences. After the hypotheses are tracked successfully for some time (in 15-30 images) they change to the recognition status.

The estimated position parameters for the left car and truck in the center in first image sequence are shown in *Figure* 8.18. In our tests we achieved an acceptable depth estimation quality for such vehicles, which are projected to image regions of size $30 \times 30[pixel^2]$ and larger. This means for given projection conditions a maximum depth of $60[m]$ only, but by increasing the image resolution only slightly, vehicles located at much larger distance from the observer should be recognized properly.

In *Figure* 8.19 the dynamics of the estimation variances and gain parameters is shown for the depth and side positions of the two best tracked vehicles in the first image se-

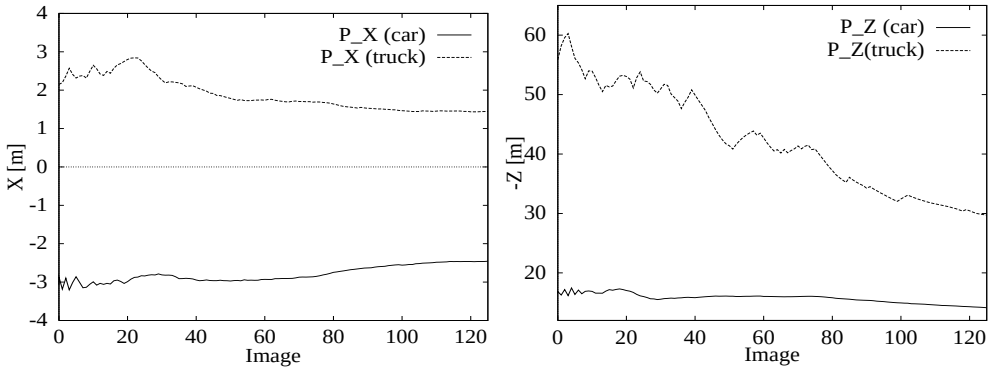


Figure 8.18 The estimated side positions (left) and depths (right) of the car and truck.

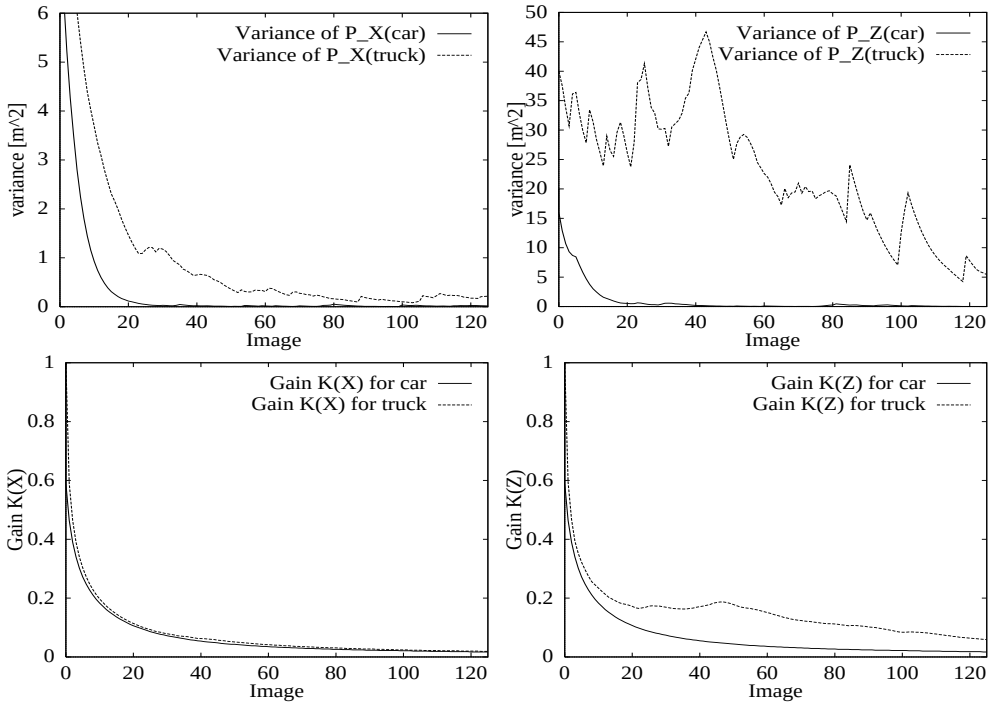


Figure 8.19 Estimation variances (top row) and gains (bottom rom) corresponding to two tracked vehicles.

quence. It can be observed that for the truck hypothesis the depth tracking error was relatively high several times, but it was nearly immediately compensated due to the self–adaptive increase of the corresponding gain parameter.

8.5 CONCLUSIONS

The adaptive recognition scheme was applied to model-based object recognition in image sequences of traffic scenes under a moving camera. A particular application system was developed which is able to recognize the ego-position, the road and moving vehicles. The scene and object domain restrictions given by the model lead to the creation of a robust image analysis system, which has to deal with eminent problems of unknown camera movement, of highly projective mapping of near objects and of nearly parallel mapping of distant objects.

A solution to the problem of how to detect the road elements robustly, and how to associate certainty judgments to measurements, is provided. The road parameters to be recognized are the road width, road lane number and road curvature. The robustness of measurements is achieved by using weighted measurements from a short time tracking of individual measurements. Two alternative methods for ego–motion based road curvature estimation are proposed. Both methods require knowledge of the ego–motion data and of the dynamics of vanishing point or ego–position estimations (alternatively).

The presented 3–D object module constitutes the, so far, most complete implementation of the adaptive recognition approach. The generation and estimation of vehicle hypotheses is as follows: after tracking the object features in a short image sequence the initialization of vehicle's state parameters is performed; then the current 3–D object parameters are projected onto the image plane and they are matched with new image features; finally the individual matching results lead to the modification of the object parameters.

9 DYNAMIC TRAFFIC SCENE DESCRIPTION

The system–shell *ERNEST* [NIS90] was chosen for the development of our application system for traffic scene recognition [KAS95, KAS97]. ERNEST uses the *procedural semantic networks* as the knowledge representation type and the optimal graph search as the basic control mechanism (*Figure* 9.1). A short introduction to this framework (i.e. an explanation of knowledge representation, inference rules, basic control) is given in the first subsection. Our application system is specified after supplying the model network with application-dependent methods to the knowledge representation module. ERNEST was designed for a static type of analysis, i.e. single images, single spoken sentences. For image sequence analysis the author has developed a consecutive control algorithm in the ERNEST framework.

9.1 A KNOWLEDGE-BASED REALIZATION

From a computational point of view an image analysis system can take the following alternative design types:

- a *procedural* system - usually suitable for the signal, iconic and segmentation levels;
- a *model-based* system - usually suitable for the object recognition level;
- a *knowledge-based* analysis.

At the scene interpretation level the analysis system usually takes a knowledge-based system form. The scene contains objects of complex structure, and abstract processes appear also. During the analysis the knowledge stored at all data abstraction levels is explored, i.e. from the signal level to the cognitive level. The framework of a knowledge-based system allows the integration of particular system components. The main system modules are [NIE90]: the *Model* (knowledge about the application field stored in the given representation form) and the *Control* (implementation of an application-independent control strategy).

Among the *knowledge representation* types the most popular so far are: *grammars* [FU 82], *production rules* [HWA86], *relation structures* [HAR79], *graph– and structure–grammars* [KAS87], *frames* [BRO81] and *semantic networks* [NIS90]. Appropriate control strategy implementations take the form of: *search* (i.e. state space search, AND/OR graph search) [NIL82], *linguistic analysis* (i.e. syntactic–semantic parser) [FU 82], *logical reasoning*, *structure matching* and *relaxation*.

9.1.1 The ERNEST system structure

In the knowledge base one distinguishes the *model* and the *data* structures, as well as several *inference rules* for model activation (the model expansion and instantiation rules). The procedural semantic network may contain both domain-specific and

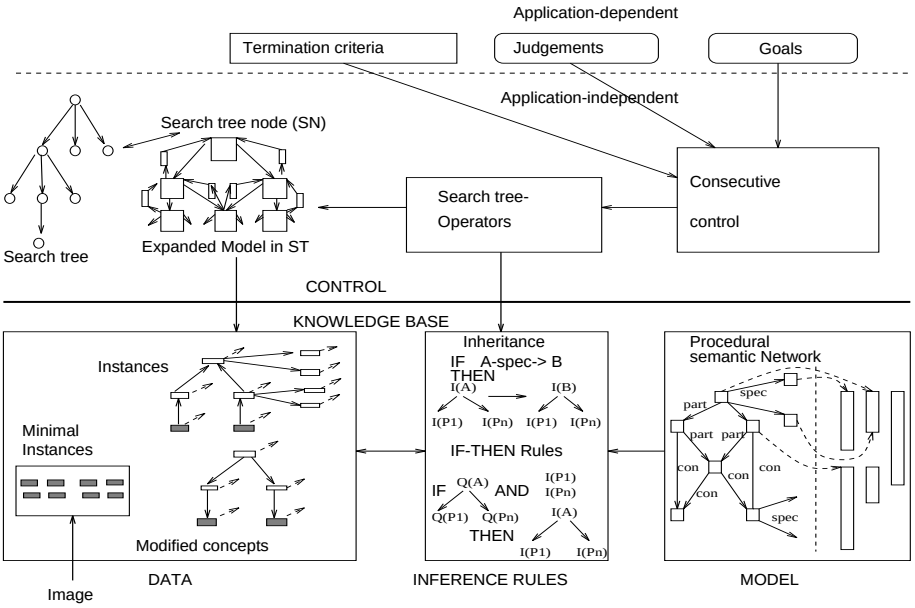


Figure 9.1 The ERNEST system structure.

application-independent knowledge at different data abstraction levels, i.e. iconic processing, segmentation, object recognition, scene interpretation.

The semantic network provides three node types: the *concept*, the *modified concept* and the *instance*, as well as three link types: *part*, *concrete*, *specialization*. A part is *context dependent* or *not*. Part– and concrete–links of a concept are aggregated into *modality* sets, and each link is marked inside a set by one of the labels: *obligatory, optional, inherent* or *reference*.

Declarative knowledge takes the form of *concepts*, whereas procedural knowledge is given by *procedures* referred to by concepts. A concept with its procedures can describe a material object (e.g. car, road, image edge, etc.), or a non-material object - situation (e.g. traffic jam, "red traffic lights are on", "people are walking across the road "). The syntax of concepts allows the representation of attribute and judgement values, it provides links to other concepts along two hierarchies (part- and specialization-hierarchies) and it refers to relationships that should hold between referred concepts. The procedural model consists of computation procedures for attributes, relationships and quality judgement.

In *data* the initial, intermediate and final results of analysis are stored. They take the form of *instances* of concepts of *modified concepts*), and they are generated by applying inference rules to given data entities and selected concepts.

There exist five "IF–THEN" rules for inferences along the part-hierarchy (three bottom-up instantiation rules and two modification rules), and one derivation rule for inferences along the specialization hierarchy [SAG90]. At first, a *partial* instance $I_{partial}(A_i)$ of a modality set $md_i(A)$ of concept A (or its modified concept $Q_{partial}(A_i)$)

is computed by requiring instances of the *context independent* parts and concretes only (RULE 1). Having a partial instance of A_i, instances of *context dependent* parts $M = \{(I(K_l + 1), ..., I(K_n))\}$ of given modality set $md_i(A)$ can be generated. With the instances of $\{M\}$ and due to RULE 2 the partial instance of A_i can be *completed* ($I_{complete}(A_i)$). RULE 3 checks whether there are instances of *optional* parts or concretes and it generates *extended* instances from a complete instance of A_i.

Constraints can be propagated upwards (RULE 4) or downwards (RULE 5) in the knowledge hierarchy. A repeated application of the top-down concept modification rule to some goal concept leads to model expansion into an attributed AND / OR graph. The purpose of the instantiation rules is to make an expanded model-to-image matching. The uncertainty of analysis is modeled by the generation of competitive instances or modified concepts, which are returned from the user-defined procedures, called from the inference rules.

For the judgement of search space nodes an estimation of the goal object judgement with respect to the set DATA(N) is performed. This measure should satisfy the admissibility requirements for the A^*–tree search algorithm.

9.1.2 Basic control

In the case of a complex scene model a hierarchy of object concepts is usually given. Theoretically multiple hypotheses could be generated for all object concepts (called *instances* or *modified concepts*) and/or for all data items and previous inference results. The basic control performs a model-to-image matching process in a step-by-step manner, by combining proper data subsets with model concepts at each step, i.e. narrowing the computational complexity according to optimization criteria provided by the user.

One of the advantages of a knowledge based system is the availability of a mainly application-independent control. This general part stores the competitive analysis results in a search tree, it controls the selection of the current best search node, due to given node judgements, and it activates proper inference rules for the selected node. Additionally, the user should supply some specific control procedures (in order to determine the start and end conditions) and he should supply judgement procedures for data items and decision tree nodes.

The main features of this basic control in ERNEST are [SAG90]:

1. *non-deterministic analysis* – the analysis flow depends on the current data and model, the ordering of analysis steps is not specified in advance.
2. *varying direction of analysis* – the analysis flow varies between a "top–down" model expansion (modification) and a "bottom–up" instantiation of expanded concepts, and a "bottom–up" modification of more abstract concepts or the derivation of more specialized goals.

This basic control has already been explained by the example in *Figure* 6.3.

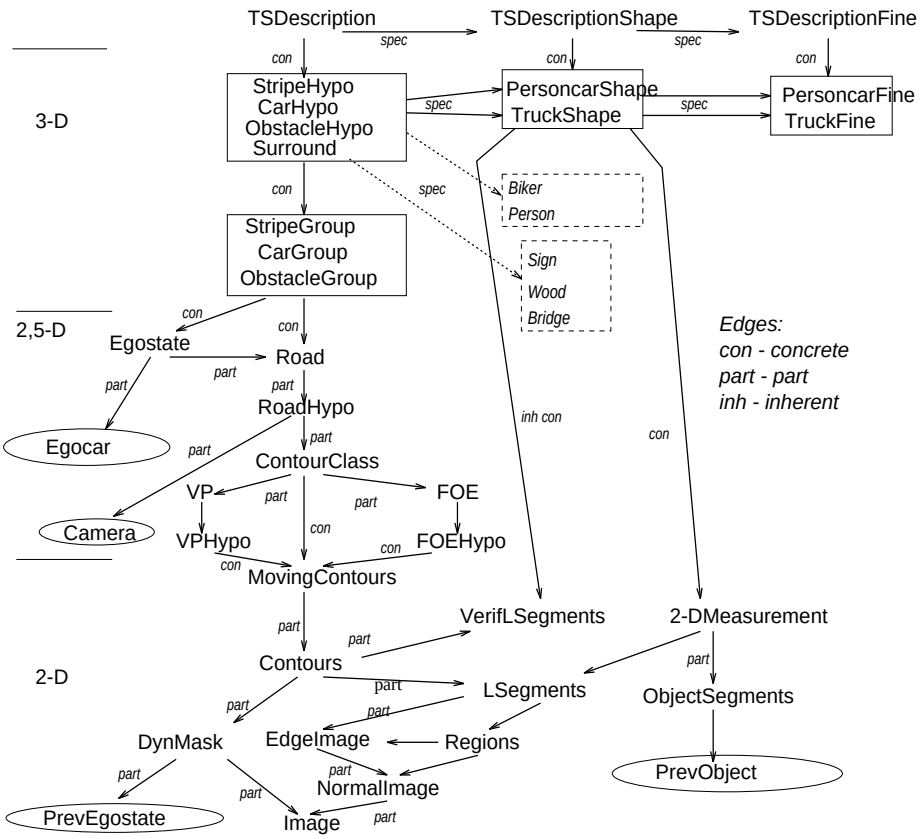


Figure 9.2 The model structure for traffic scene interpretation.

9.2 THE TRAFFIC SCENE MODEL

The author has developed an application system for traffic scene recognition, by defining an appropriate model in the ERNEST framework and by supplying a newly designed consecutive control to ERNEST.

9.2.1 The model network

The model scheme is given in *Figure* 9.2, where at first the concept groups corresponding to segmentation, road recognition and object recognition tasks can be distinguished. The model data is mapped onto two hierarchies of the semantic network representation: the concrete– (with its part sub-hierarchy) and specialization–hierarchy. Four concrete-levels are defined, denoted as: scene description, *3–D* (object recognition), *2.5–D* (road recognition) and *2–D* (segmentation and motion detection). Three specialization levels correspond to the three concepts at the highest abstraction level: *Description*, *DescriptionShape* and *DescriptionFine*. These most abstract concepts represent the general analysis goal, decomposed into a three-goal analysis sequence.

The *3–D* level contains in the most general "column" the concept group *Objects* and the concept *Egostate*. The objects related to the road are divided into three classes: *StripeHypo* (road stripes), *CarHypo* and *ObstacleHypo*. An extension by other type of objects, i.e. *surrounding* objects (traffic signs and information tables) or *horizon–* (objects positioned over the road), is possible. The car objects are specialized into *PersoncarShape* and *TruckShape*, and further into detailed car models *PersoncarFine* and *TruckFine*. The *3–D* model contains further the concept family *Groups*, the concept *Camera* (projection conditions) and *Egostate* (ego-car parameters).

The *2,5–D* model contains the road-related concepts: *Road*, *RoadHypo*, *ContourClass*, vanishing point and focus of expansion point concepts *VP, FOE*.

The *2–D* represents at first iconic processing by the concepts: (*Image, NormalImage, EdgeImage* and *DynMask*). Next concepts for segmentation and visual motion estimation follow: (*Contours, LSegments, Regions* and *MovingContours*). At this level some concepts represent the projection of higher level concepts, like *ObjectSegments* (the projection of a segment set of a 3-D object instances onto the image plane). The adaptive recognition scheme requires additional concepts of *2-DMeasurement* (for limited area grouping of segments) and *VerifLSegments* (representing the segments already corresponding to current object hypotheses).

A cycle-free model

A necessary condition for the termination of a single-image analysis in ERNEST is a cycle-free model network. This requirement would not be satisfied by our model, as we have introduced cycles for concepts *DescriptionShape* and *DescriptionFine* ranging between the object and segment levels. This design is necessary if we want to represent some hypothesis by the same instance over the time. In order to open the model cycle, explicit minimal concepts are introduced of the type *PreviousXXX*, for example *PreviousObject*. These are copies of previous image instances, which are added to the minimal instances corresponding to current image data.

Supporting the control

The syntax and semantics of this model network support the control in limiting the computation effort and increasing the stability of results. This is emphasized here, as ERNEST was primarily designed for static signal analysis, meeting the main requirement of providing competitive search paths with consistent local interpretations at each path.

Due to the small syntactic model variability a repeated generation of same inferences along different search paths is kept minimal. The parallel tracking of competitive hypotheses of vanishing point, road and 3–D objects is made possible due to the existence of concepts of the type *XxxHypo*. The selection of the best hypothesis corresponds to the instantiation of following concepts: *VP* or *FOE*, *Road* and *Description*.

The reduction of competitive hypotheses is performed step-by-step during the derivation of a more specialized goal concept.

Due to such model design our consecutive control can be a relatively simple extension of the basic control. Although it is search-tree oriented, with local consistent instance

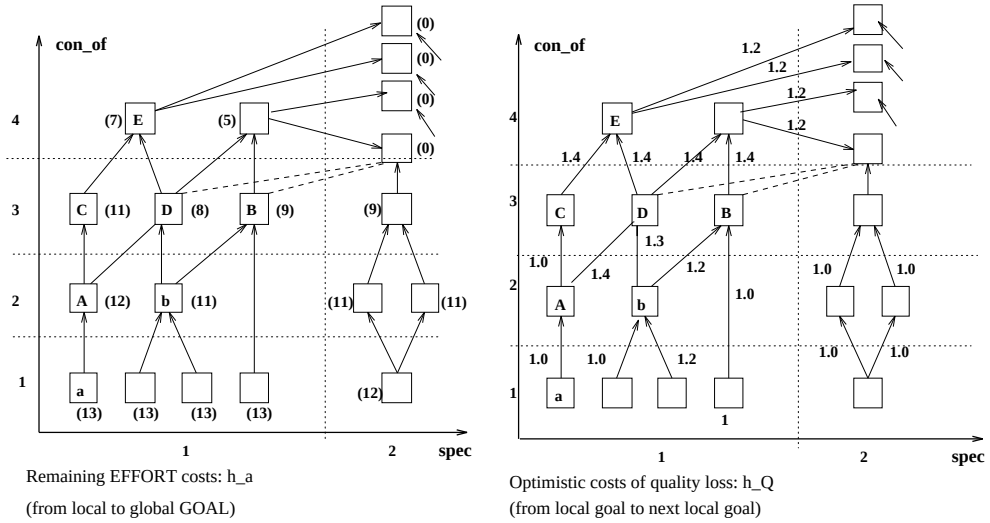


Figure 9.3 Example of remaining cost estimation during model-based analysis under a semantic network model: remaining effort costs are associated with concepts (left drawing), while quality (risk) costs are associated with links between concepts (right drawing.)

sets provided at each search node, it can avoid the problems of high computation complexity and low stability of results.

9.2.2 The node judgement scheme

As we know from section 6. a *monotonic* judgement scheme is required for the cycle-free search graph of consecutive control. The basic A^*–search for the selection of search nodes, assures finding the optimal path in such space (if one exists).

In single image analysis usually the *compatibility* and *certainty* factors of an object hypothesis, obtained during an image-to-model matching, are estimated [NIB90]. The compatibility value expresses the quality of correspondence between model constraints and image data, whereas the certainty depends on the quality of detected image segments (for example, a good-quality linear segment is sufficiently long and thick). Thus the judgment vector of an object hypothesis starts with the compatibility costs g_C, followed by the certainty costs g_S. Both measures take into account the attribute values of an image object and their relation to model object attributes. For model concepts with complex structure, a third component g_R is also included into the judgement vector, and it represents the result of the structure relation test (verification) for a given model concept. This test is computed over the set of part instances of a given concept. The final judgement of object hypothesis is some function (for example the minimum function) of all three components $< g_C, g_S, g_R >$.

In the context of the A^*-search, in order to assure a monotonic judgement scheme, the optimistic estimation (heuristic) of quality costs h_c, h_S and h_R of reaching a global analysis goal should also be included in the application model.

In dynamic image analysis one additional criterion appears. One has to consider the computation effort (time) required for the generation of a particular hypothesis. The goal of a judgement scheme is to find a proper balance between the high quality of a hypothesis (via certainty, compatibility, relation test) and the low computational effort for its computation. Such a scheme will satisfy the general requirement of dynamic analysis - a limited time answer of the analysis system to every image data, while always providing sufficiently robust results.

In accordance with our control algorithm, with each search node $n \in \mathcal{G}$ one (local, temporary) goal concept object $LG(n)$ is associated, for which the local inference net (expanded model) is created during the analysis process. Obviously in the model network some final analysis goals are expressed by the existence of most abstract and most specialized goal concepts $\{GG\}$(called: global goal concepts). The cost function $f(n)$ of search node n will be expressed by the judgement of the local goal hypothesis and the estimation of remaining costs of reaching an instance of some global goal concept.

Summarizing the above discussion, for each search node n, a judgement vector $\psi(n)$ is proposed, that contains the main judgement value $f(n)$ and some further specialized costs. The judgement vectors of two search nodes are compared in lexicographical order, i.e. if first elements $f(n_1)$, $f(n_2)$ are equal then the lower costs of the second elements are taken into account, and so on. The judgement vector consists of:

$$\psi = [f, g, f_Q, g_Q, f_S], \tag{9.1}$$

where the components have the following meaning -

– f – are additive *total costs* of a given search node:

$$f = \max\{g_Q, h_Q\} + (g_a + h_a), \tag{9.2}$$

– g – are the *effort costs* of reaching a given search node from the start node:

$$g = g_Q + g_a, \tag{9.3}$$

– f_Q – are the lowest *quality costs*, estimated for some global goal, reachable from a given local goal:

$$f_Q = \max(g_Q, h_Q), \tag{9.4}$$

– g_Q – the *quality costs* of a local goal object,

– f_S – the total *certainty* of an achievable global goal instance.

By f_a we denote the time (computation) *effort costs* of the analysis process, which also consists of the current effort and the expected remaining effort costs:

$$f_a = g_a + h_a. \tag{9.5}$$

The current *quality* $g_Q(O)$ of some object hypothesis O (instance, modified concept) depends on following components:

$$g_Q(O) = \max_{i \in \{Parts(O)\}} \{g_Q(i)\} \times g_C[Attr(O)] \times g_R[Rel(O)], \text{ with } q_C \geq 1, q_R \geq 1, \tag{9.6}$$

i.e. the quality costs (*risk*) of hypothesis O are not lower than the quality costs of its parts. Usually they are higher as the judgements of compatibility- and relation tests of O are not optimal.

The total *quality costs* $f_Q(n)$ of search node are a function of the quality of the current local goal hypothesis $g_Q[O_L(n)]$ and of an optimistic estimation of an achievable global goal $h_O[O_G(n)]$. This optimistic estimation can be expressed by the certainty judgement of image segments, that are not included in the local inference net of node n (are not explained so far):

$$h_Q(n) \quad = \quad \min_{i \in Segm - Data(n)} [g_S(i)], \tag{9.7}$$

where $Segm$ means the set of current image segments (minimal instances), $Data(n)$ - is such a part of set $Segm$, which is already included in node n and $g_S(i)$ are the certainty costs of some segment i.

Example 9.1 - In *Figure* 9.3 an example is provided, illustrating how remaining matching costs are computed, after reaching an instance of temporary goal concept. In both drawings a schematic model network is given (this is not a decision tree). The nodes of this model represent concepts in the semantic network, whereas the links between nodes express the two model hierarchies: 1) *spec* - the horizontal specialization hierarchy, 2) *con_of* - the perpendicular *concrete_of* hierarchy. In this example two specialization levels and four concrete levels are given. The concepts represent possible subgoals of the incrementally performed model-to-image matching analysis, the links represent bottom-up transitions between subgoals, i.e. from lower concrete-level subgoal to a more abstract or more specialized subgoal. Remaining effort costs are associated with concepts (left drawing), whereas usual (expected) quality risk costs are associated with links between concepts (right drawing).

9.3 CONSECUTIVE CONTROL

A consecutive tree search algorithm is proposed for the control mechanism of dynamic scene interpretation by an ERNEST-like image sequence analysis system. It combines the focusing strength of a model-based tree search with locally parallel tracking of competitive object hypotheses, i.e. instances of model concepts and modified concepts.

9.3.1 Requirements

A robust control for dynamic analysis means usually a tradeoff between computational complexity and the quality of results. This requires *incremental* and *adaptive* analysis features. The incremental feature means to focus on important model and data parts first and subsequently to extend the temporary interpretation to a full one. The adaptive feature induces two recognition phases – hypothesis initialization and tracking. The key problem is a proper selection of the initial interpretation among the set of competitive interpretations. This requires the existence of a robust *judgement* scheme

APPL: *Parameter_Init*
Search_tree_Init;
WHILE APPL: *End* is not satisfied
  WHILE search nodes exist in OPEN
    v =*Select_and_eliminate_node_from*_OPEN;
    IF APPL: *Single_End* is satisfied
    THEN GOTO NEXT
    APPL: S = *Goals(v)*; bottom–up modification
    IF S is not empty
    THEN *Init_Subspaces(S)*;
    ELSE IF an Entity $o_l \in$ DATA(v) can be refreshed
      THEN *Refresh(* o_l, v*)*;
      ELSE IF an Entity $o_l \in$ DATA(v) can be instantiated
        THEN *Instantiate(* o_l, v*)*;
        ELSE IF an Entity $o_l \in$ DATA(v) can be modified
          THEN *Modify(* o_l, v*)*;
  NEXT: *New_Init()*

Figure 9.4 The consecutive tree search algorithm

of partial results. Contrary to single image analysis, in the case of dynamic analysis the *stability* of a hypothesis over time allows the creation of reliable judgement schemes. Obviously this requires a parallel tracking of competitive hypotheses in a sufficient amount of time.

9.3.2 The algorithm

The consecutive control module extends the basic control by the following principles:

1. *consecutive* processing mode – the images are processed one after the other in order, according to their time-order in the image sequence.
2. *incremental* analysis principle – the result of analysis for image k need not to be complete nor consistent, i.e. many search tree nodes with competitive scene descriptions for image data k can still exist, and some of them constitute the start set for the analysis of next image $k + 1$.
3. *recursive* estimation of instances – due to a new inference rule of *instance refreshing*, the previous instance values are taken into consideration during repeated evaluation of such instances in the next instantiation steps (the prediction-modification principle of recursive estimation).

The general *consecutive* tree search algorithm (image sequence analysis with explicit search space representation) is described in *Figure* 9.4. The control always activates search tree operators that call the inference rules. They select and expand the current best node of the search tree, according to the local inference set of the node. The WHILE–cycle corresponds to a single image analysis. There are three types of inference operators that generate, modify or *refresh* the instances and perform appropriate search space expansion. The function *New_Init* selects some search space nodes and propagates them to the next image search tree.

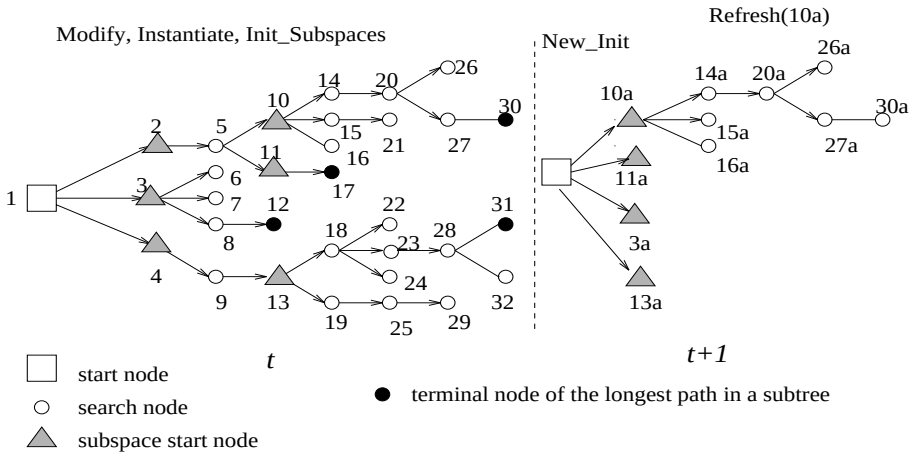


Figure 9.5 Example of search tree propagation by the use of "New_Init" and "Refresh".

The *refreshing* operator is equivalent to a prediction-modification of instances, that were already generated previously and which are referred by this particular search tree node. In this way an update of previous instances on the basis of new data items occurs. In order to refresh an instance all of its part instances should be already refreshed.

The user-defined application functions (with prefix APPL) determine the processing parameters, the termination conditions and the selection of temporary goal concepts in the model.

Search Tree Propagation and Refreshing

Let us first assume the following realization of the *New_Init* procedure. In this possibly wrong solution, the best nodes from the final, previous-time terminal set would be selected and they would be made the immediate successors of the next-time starting node (in the new search tree). This solution would correspond to an immediate refreshing of all instances of the selected search node in *Refresh*. Only one successor node of the "refreshed" node in the search tree would be generated. This solution would work well only if the interpretation we search for is really among the selected set and if the same number of hypotheses is generated in subsequent images. As this requirement is seldom satisfied another solution to *New_Init* was implemented.

In our solution local subtrees that correspond to different sub-goal concepts, are recognized first (*Figure* 9.5). The root node of such a subtree was generated by a bottom–up modification of its goal concept, which is a sub-goal of overall analysis. Now, instead of propagating only terminal nodes, all the root nodes of subtrees, that contain a node in the OPEN set, are propagated. On the basis of the instances from selected subtree an *updated* subtree root node is propagated to the next search tree. While the original node contains modified concepts the updated local root node refers all the instances, that have been generated in its subtree (called a *local instance* net).

If the *Refresh* procedure is applied to an updated node the longest path from the previous subtree is repeatedly generated. As the previously performed instantiations are

now repeated, the generated hypothesis numbers need not to be equal in both images. All competitive instances along the refreshed search path are modified (i.e. recursive estimation of instance attributes), not only the currently best instances. In this solution it is no longer required that the same number of competitive instances (structure) is generated in subsequent images. Now after several images the refreshed longest "search tree path" should, most of the time, be the best solution in this subtree. If this path is proved to be wrong the analysis for current image can still be continued from other search paths with the information contained in the local instance net of the appended start node.

Local Instance Net

Let us explain the required short-time tracking of competitive instances by a three-image example with three objects. For each object there are always three competitive instances generated. During the analysis of first image t obviously the tracking process of three objects would be initialized, i.e. this corresponds to the expansion of a search tree path, which contains three instances of the object concepts A, B, C. A strategy of following the single best instance only would concentrate on a single triple of correspondences in three consecutive images from the total possible set of 27^3 correspondences. If at the time $(t + 2)$ it is verified that the instances tracked so far are wrong, there is no possibility of going back to competitive solutions as long as the other search tree nodes have not also been continuously refreshed. Hence, due to the tracking of one tree node with three instances alone the image description will probably not be stable.

In the proposed strategy of local inference net propagation, the tracking of competitive instances is possible. In this example one search path of length 3 with 3×3 competitive instances (9 search tree nodes) only would be generated. After several images a better stability of some instances should be achieved and the selection of the consistent instance set will be much easier than the single image based selection. The number of competitive instances in the net is reduced step by step to one instance for each object. Thus in the tracking phase the local instance net will be reduced to 3 instances only.

9.4 TEST EXAMPLE

In the general traffic scene model we have selected a subset of concepts, which are responsible for the general recognition mode of road objects under ego-motion, as presented in *Figure* 9.6. Competitive instances of the following concepts are usually generated during the analysis: VanishingPoint (VP), Road, StripeGroup, StripeHypo, CarGroup and CarHypo. Competitive instances of the global goal concept TSDescription represent different subsets of object hypotheses - consistent scene descriptions.

Restricting the highest number of competitive inference results to six modified concepts in one modification step and to three instances in one instantiation step, following the complexity of the consecutive tree search algorithm has been experimentally verified for a given model and image sequence (*Figure* 9.7): during the initialization phase

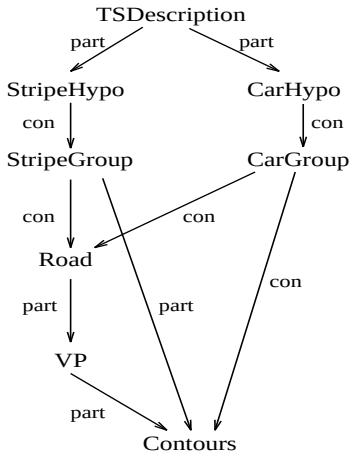


Figure 9.6 The general model structure for traffic scene interpretation.

	Time			
Instances/nodes	0	5	20	50
I(VP)	3	3	3	2
I(Road)	4	4	2	2
I(StripeGroup)	2	2	1	1
I(CarGroup)	2	2	1	1
I(CarHypo)	35	21	3	3
I(StripeHypo)	12	6	6	6
I(TSDescription)	2	2	1	1
Total I	60	45	17	16
Nodes	138	107	58	54
Inconsistency	of all	small	no	no

Figure 9.7 Example of consecutive search complexity.

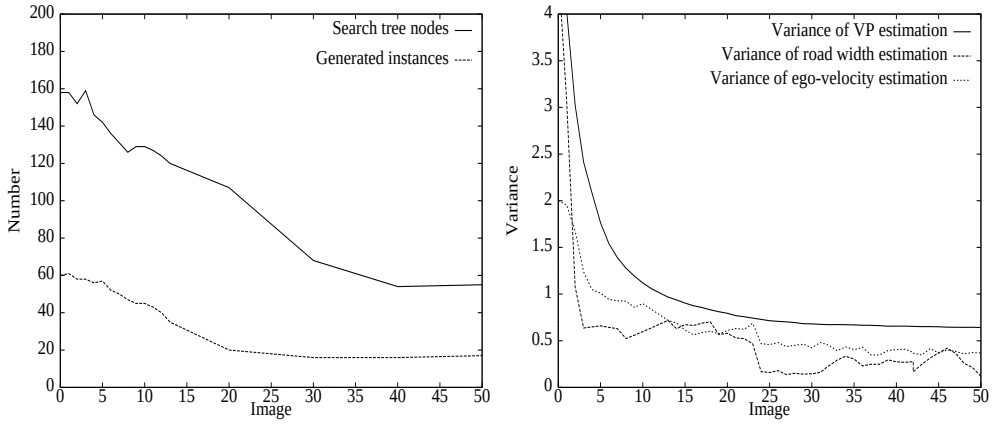


Figure 9.8 Results: (left) complexity of search, (right) stability of three instances

the average number of competitive instances for one concept was reduced from three to two; after 50 images 1.2 competitive instances were stored on average (*Figure* 9.8(left)).

The stability of tracked instances was achieved after 20 images. In *Figure* 9.8 (right) estimation variances of three refreshed instances in 50 images are presented. The variance of the vanishing point position is expressed in $100 pixel^2$ units, the road width variance is given in m^2 and the translational ego-velocity variance is given in $(m/0.04s)^2$.

10 CONCLUSIONS

This work gives a broad view of computer-based image sequence analysis. The main goal of this work was to develop general adaptive computation schemas, which could be applied for analysis tasks at different data abstraction levels. This goal was achieved by the development of two adaptive computation schemas, which are based on the *connectionist* and *dynamic systems* theories. In fact, two main analysis approaches were modeled: (1) the *batch* approach, applied to a limited-length image sequence and (2) the *recursive* approach, applied to an infinite sequence. Providing different implementations of the above schemas for the solution of particular image sequence analysis tasks it was shown that adaptive image analysis may be successfully performed at five hierarchically ordered data abstraction levels.

In the first scheme the use of artificial neural networks for low-level and segmentation-level image analysis was proposed. This kind of method is biologically justified, contrary to most technically- and numerically-oriented methods, established in the pattern recognition theory.

In the first implementation typical feed-forward neural networks for the many-image restoration problem at the lowest data abstraction level (so called *signal* level) were applied. We studied experimentally the blind source separation (BSS) problem and its solutions by *independent component* analysis (ICA), focusing on the problem where the number of sources is different from the number of sensors or outputs of the networks. This is an important practical issue, which is usually omitted from consideration. Moreover, in the basic ICA/BSS model it is assumed that the source signals are mutually independent. For images this assumption is often violated, for example face images are usually clearly correlated. Nevertheless, the output images given by our adaptive techniques are in fact more independent than the original correlated source images.

Next we have demonstrated the usability of the ANN-learning based scheme for image frame classification, performed at the *iconic* level. In many restricted world applications such a classification process is already sufficient to meet user requirements. The proposed adaptive techniques for image compression and discriminant analysis are superior to similar conventional numeric techniques, when complexity and computation time is considered. Especially images, due to their large sizes, lead to large matrix inversion problems required by conventional techniques, but they are changed into relatively simple neural learning rules in the adaptive techniques.

At the *segmentation* level, the problem of visual motion detection/estimation and a motion-based image segmentation was discussed. We have proposed a relaxation-like feedback ANN for the solution of visual motion detection. The behavior of this approach was experimentally compared with some other, conventional approaches and it was found to perform similarly well for the stationary camera. It was also shown, that visual motion estimation is very sensitive to camera motion and that geometry based image segmentation should be preferred in the moving camera case.

For the analysis of an infinite image sequence the *adaptive object recognition* scheme was developed. Contrary to simple object tracking approaches, this scheme is based on parallel tracking of competitive hypotheses and it is extended to consistency and recognition search. During each tracking step a recursive estimation of object state parameters is performed and a judgment of each object hypothesis is also provided. The most frequently applied method of estimation is the *Kalman* filter, either a linear KF or non–linear EKF one, depending on the type of state transition/projection function. In order to be an optimum estimator this tool needs a proper judgment scheme for individual measurements. We have observed that this requirement is usually violated in object tracking approaches, i.e. the measurement judgment is not separated from the expectations of the current object hypothesis. Hence, a Kalman filter can be used as the recursive estimator if properly designed measurement judgement schemas, independent from the object hypothesis, are available in the given implementation of the analysis task. Otherwise a *tracking-error* based estimator should be used, where the estimator's gain is made independent from the measurement variance. This solution explores the self-adaptation of learning rates in ANN and the estimations follow non-stationary objects very well.

Notwithstanding, which of the above two recursive estimators we choose in specific implementation, the summarized variance of an object hypothesis will express its quality and together with the time of tracking it will allow a reliable selection among competitive hypotheses. For the problem of consistent hypothesis selection we have proposed the optimal A^*-graph search algorithm. It is known that A^* is an optimal search method under consistent (monotonic) heuristic part of the judgement function. In general, under an admissible heuristics another algorithm A^{**} dominates over A^*. We discussed modifications of A^* (called $B-, C-$ and $D-$graph search), specially designed for reducing the complexity of graph search, by avoiding a repeated expansion of graph nodes.

Among the possible implementations of above object recognition scheme first we considered two analysis tasks at the *segmentation* level. In the contour tracking- and vanishing point recognition- methods a weighted averaging of corresponding measurements in a short image sequence is performed. This allows a more reliable image structure detection than in single image or in two images. Moreover, it also provides a robust scheme for the hypothesis–independent judgment of new measurements. Vanishing point line segments are detected directly in front of the camera vehicle, thus avoiding the problems which have approaches requiring good visibility of distant road boundaries (projected to potentially occluded or small image parts).

Next the adaptive object recognition scheme was applied for *model–based* solution of the object recognition problem in image sequences under true camera motion. In two particular applications the goals were to recognize the road and on-road objects, i.e. moving vehicles. The approach to road recognition has been demonstrated to be robust, even in the case of obstacles and ego-state estimation problems caused by the partially unknown movement of the camera. Contrary to single object tracking, which is mainly a stabilization task performed as a hypothesis–driven process, in the presented applications we provide a 3-D model–based explanation of image segments and we assure a real recognition of moving objects. Due to the large depth interval of the observed objects in

relation to the camera position, causing very different image projections of even similar objects, several object shape specialization levels (model restrictions) are provided.

Finally, a knowledge-based based system for dynamic scene description was designed, i.e. a consecutive search algorithm for general analysis control and a semantic network structure for scene model representation. The control algorithm and the concepts of the model are the most complete implementation of the adaptive object recognition scheme in this work. The most general objects correspond to the instances of model concepts. After completing the analysis for a given image some search nodes remain for the next image analysis, i.e. there is a mechanism for search path storage included, which allows the prediction of some initial competitive search sub-spaces in the search tree for the next consecutive image. Due to bi-directional (model- or data-driven) inference rules a flexible tradeoff between low computational complexity and high quality of the image sequence analysis is possible.

Recently the connectionist research on adaptive system design focuses on coordination of heterogeneous networks and on the integration of signal level-, sub-symbolic and symbolic processing functions in one system. This requires the development of network structures for the representation of symbolic information and of complex data structures. This line of research will lead to system configurations consisting of multiple nets, organized in competitive or hierarchical architectures. The work described in this monograph presents a step towards unified models of low-, intermediate and high-level image analysis, as both ANN-based and structural model-based analysis schemas have been proposed. The goals of this monograph have been achieved principally through the description of the author's research on both adaptive analysis tools and related non-adaptive approaches. Both theoretical and implementation questions have been addressed, and test results have been provided for all proposed solutions of considered analysis tasks.

In the presented complete form, the proposed two image analysis schemas were not published before. The particular methods were developed and tested by the author himself. Appropriate references to previous publications of the author are provided. In most of the references to his own paperwork there is a single author only [KAS85, KAS86], [KAS87, KAS89a], [KAS89, KAS90, KAS91], [KAS93a, KAS93b, KAS94], [KAS95, KAS95a], [KAS97, KAS00]. Some of the other publications are co-authorized papers, as in the past the author of this monograph was a member of large international teams, where he collaborated with his team leaders and other team members. In most of the referred to co-authorized publications, he was the first co-author [KAS89a, KAS93c, KAS94a, KAS94d, KAS94b, KAS96], [KAS96a, KAS96b, KAS97a, KAS98]. This means, most of the own references present work, done either solely by the author, or they present the author's work done in collaboration with his team leaders. In the latter case, the author has made a dominating contribution to such a paper. Also a few publications are referred to, where the author was a second or further co-author only ([CIK96a] - section 4, [CIK96] - sections 3-4, [KAK97] - section 5, [CIK99] - section 4). In such cases only those sections are explicitly referred to, for which the author made an unique or dominating contribution.

Acknowledgments

The author is very grateful to Professors: Heinrich Niemann (University of Erlangen-Nuremberg, Germany), Andrzej Cichocki (WUT Warsaw and BSI Riken, Japan), Shun-ichi Amari (BSI RIKEN, Japan), Władysław Skarbek (WUT Warsaw) and Juha Karhunen (HUT Helsinki, Finland) for the honor and opportunity to work closely with them in the past.

The author is also thankful to Professors: Krzysztof Malinowski, Piotr Tatjewski and Cezary Zieliński for the opportunity to prepare this work at the Institute of Control and Computation Engineering of WUT Warsaw.

Special thanks are due to my former and current colleagues from different countries for useful comments and discussions regarding my work in this area.

References

[ABB93] Abbas H.M., Fahmy M.M.: Neural model for Karhunen–Loéve transform with application to adaptive image compression. *IEE Proceedings–I*, 140(2):135–143, 1993.

[ACK85] Ackley D.H., Hinton G.E., Sejnowski T.J.: A learning algorithm for Boltzmann machines. *Cognitive Science*, vol. 9(1985), 147–169.

[AGG88] Aggarwal J.K., Nandhakumar N.: On the computation of motion from sequences of images - a review. *Proceedings of the IEEE*, vol. 76(1988), No. 8, 917–935.

[AMA67] Amari S.: Theory of adaptive pattern classifiers. *IEEE Transactions on Electric Computing*, EC–16(1967), 299–307.

[AMA77] Amari S.: Neural theory of association and concept formation. *Biological Cybernetics*, vol. 26(1977), 175–185.

[AMA96] Amari S., Cichocki A., Yang H.: A new learning algorithm for blind signal separation, *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge, MA, 1996, 757–763.

[BAG83] Bagchi A., Mahanti A.: Search Algorithms under Different Kinds of Heuristics – a Comparative Study. *Communications of the ACM*, vol. 30(1983), No. 1, 1–21.

[BAG88] Bagchi A., Sen A.K.: Average–Case Analysis of Heuristic Search in Tree-Like Networks. In: Kanal L., Kumar V. (Eds.): *Search in Artificial Intelligence*, Springer Vg., New York-Berlin-Heidelberg, 1988, 131–165.

[BAL89] Baldi P., Hornik K.: Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2:53–58, 1989.

[BAN95] Bannour S., Azimi-Sadjadi R.: Principal component extraction using recursive least squares learning. *IEEE Transactions on Neural Networks*, 6(2):457–469, 1995.

[BAR94] Barron J.L., Fleet D.J., Beauchemin S.S.: Performance of optical flow techniques. systems and experiment. *International Journal of Computer Vision*, vol. 12(1994), No.1, 43–77.

[BEL95] Bell A.J., Sejnowski T.J.: An information maximization approach to blind separation and blind deconvolution, *Neural Computation,* vol. 7, 1995, 1129–1159.

[BER93] Bernasch J., Koutny R.: Stabile objektverfolgung und detektion von nicht–vorhersagbarem *Mustererkennung 1993, Series: Informatik aktuell*. Springer–Verlag, Berlin Heidelberg New York, 1993, 19–26.

[BES85] Besl P., Jain R.: *Three–Dimensional Object Recognition*, *Computing Surveys*, vol. 17(1985), No. 1, 75–145.

[BIE89] Bielik A., Abramczuk T.: Real-time wide-traffic monitoring: information reduction and model-based approach. *Proceedings 6th Scandinavian Conference on Image Analysis*, 1223–1230, Oulu, Finland, 1989, Pattern Recognition Society of Finland.

[BLA93] Blake A., Curven R., Zisserman A.: A Framework for Spatiotemporal Control in the Tracking of Visual Contours, *International Journal of Computer Vision*, vol. 11(1993), No. 2, 127–145.

[BOB94] Bober M., Kittler J.: Estimation of complex multimodal motion: an approach based on robust statistics and hough transform. *Image and Vision Computing*, vol. 12(1994), No. 10, 661–668.

[BOU93] Bothemy P., Francois E.: Motion segmentation and qualitative dynamic scene analysis from an image sequence. *International Journal of Computer Vision*, vol. 10(1993), No. 2, 157–182.

[BRA75] Brammer K., Siffling G.: *Kalman-Bucy-Filter - Deterministische Beobachtung und stochastische Filterung*. R. Oldenburg-Verlag, München–Wien, 1975.

[BRE84] Breimann L., Friedman J.H., Olshen R.A., Stone C.J.: *Classification and regression trees*. Wadsworth, Belmont, 1984.

[BRO81] Brooks R.: Symbolic Reasoning among 3–D Models and 2–D Images, *Artificial Intelligence*, vol. 17(1981), 185–348.

[BRO91] Brockett R.W.: Dynamical systems that sort lists, diagonalize matrices, and solve linear programming problems. *Linear Algebra Applications*, 146:79–91, 1991.

[BRO86] Broida T., Chellappa R.: Estimation of Object Motion Parameters from Noisy Images, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8(1986), 90–99.

[BRO88] Brown C. (ed.): *Advances in Computer Vision*. Lawrence Erlbaum Ass. Pub., Hillsdale, N.J., 1988.

[CAN86] Canny J.: A Computational Approach to Edge Detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8(1986), 679–698.

[CAO96] Cao X.-R., Liu R.-W.: A general approach to blind source separation, *IEEE Trans. on Signal Processing*, vol. 44(1996), March 1996, 562–571.

[CAR86] Carbonell J.G., Michalski R.S., Mitchel T.M.(Eds.): *Machine–Learning: An Artificial Intelligence Approach, Vol. 2*, Morgan Kaufmann Publishers Inc., 1986.

[CAR89] Carbonell J.G.: Introduction: Paradigms for Machine Learning. *Artificial Intelligence*, vol.40(1989), 1-9.

[CAR96] Cardoso J.F., Laheld B.: Equivariant adaptive source separation, *IEEE Trans. on Signal Processing*, vol. 44(1996), December 1996, 3017–3030.

[CAR98] Cardoso J.F.: *Entropic contrasts for source separation*. In: S. Haykin (ed.), *Adaptive Unsupervised Learning*, 1998.

[CAR87] Carpenter G.A., Grossberg S.: A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, vol. 37(1987), 54–115.

[CHA89] Chakrabarti P.P., et al.: Heuristic Search in Restricted Memory. *Artificial Intelligence, 41*(1989), 197–221.

[CHD98] Chen S.S., Donoho D.L.: Application of Basis Pursuit in Spectrum Estimation. *ICASSP'98, Proceedings*, vol. 3(1998), 1865–1868.

[CHE87] Chen J.: Fast Convolution with Laplacian-of-Gaussian Masks, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9(1987), 584–590.

[CHE93] Chen X., Dagless E., Zhang S., Thomas B.: A real–time plan view method for following bending roads. *1993 IEEE Symposium on Intelligent Vehicles*, Tokyo, Japan, 1993, 219–224.

[CHE98] Chen S.: *Learning Based Vision and Its Application to Autonomous Indoor Navigation*, PhD Dissertation, Michigan State University, Dept. of Computer Science and Eng., 1998.

[CIC94] Cichocki A., Unbehauen R.: *Neural Networks for Optimization and Signal Processing*. New York, John Wiley, 1994, 461–471.

[CIC94a] Cichocki A., Unbehauen R., Rummert E.: Robust learning algorithm for blind separation of signals, *Electronics Letters* vol. 30(1994), August 1994, 1386–1387.

[CIK96a] Cichocki A., Kasprzak W.: Local adaptive learning algorithms for blind separation of natural images, *Neural Network World*, vol. 6(1996), No. 4, 515–523.

[CIK96b] Cichocki A., Amari S., Adachi M., Kasprzak W.: Self–adaptive neural networks for blind separation of sources. *1996 IEEE International Symposium on Circuits and Systems, ISCAS'96*, IEEE Publ., Piscataway, NJ, USA, 1996, vol. 2, 157–160.

[CIK96] Cichocki A., Kasprzak W., Skarbek W.: Adaptive Learning Algorithm For Principal Component Analysis With Partial Data, *Cybernetics and Systems '96*. Austrian Soc. for Cybernetic Studies, Vienna, Austria, 1996, 1014–1019.

[CIK99] Cichocki A., Karhunen J., Kasprzak W., Vigario R.: Neural Networks for Blind Separation with Unknown Number of Sources, *Neurocomputing*, Elsevier, NL, vol. 24 (1999), 55–93.

[COM91] Comon P., Jutten C., Herault J.: Blind separation of sources, Part II: Problem statement, *Signal Processing,* vol. 24, 1991, 11–20.

[COM94] Comon P.: Independent component analysis - a new concept?, *Signal Processing*, vol. 36, 1994, 287–314.

[COT91] Cottrell G.W., Metcalfe J.: Face, gender and emotion recognition using holons. *Advances in neural information processing systems, 3*. Kaufmann, San Mateo, 1991, 564–571.

[COX93] Cox I.J.: A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, vol. 10(1993), No. 1, 53–66.

[COX96] Cox I.J, Hingorani S.L.: An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 18(1996), No. 2, 138–150.

[DAN95] Daniilidis K., Krüger V.: Optical Flow Computation in the Log–Polar Plane, *International Archives of Photogrammetry and Remote Sensing*, vol. 30(1995), Part 5W1, 214–219.

[DEC88] Dechter R., Pearl J.: The Optimality of A*. In: L. Kanal, V. Kumar (eds.), *Search in Artificial Intelligence*, Symbolic Computation – Artificial Intelligence, Springer Vg., New York-Berlin-Heidelberg, 1988, 166–199.

[DEK86] De Kleer J.: An Assumption-Based TMS. *Artificial Intelligence*, vol. 28(1986), 127–162.

[DER87] Deriche R.: Using Canny's criteria to derive a recursively implemented optimal edge detector, *International Journal of Computer Vision*, vol. 1(1987), No. 2, 167–187.

[DER90] Deriche R., Faugeras O.: Tracking line segments. *Image and Vision Computing*, vol. 8(1990), No. 4, 261–270.

[DIC88] Dickmanns E.D.: Object recognition and real–time relative state estimation under egomotion. In: A.K. Jain (ed.), *Real–Time Object Measurement and Classification*, Springer, Berlin–Heidelberg etc., 1988, 41–56.

[DIG88] Dickmanns E.D., Graefe V.: Applications of dynamic monocular machine vision. *Machine Vision and Applications*, vol. 1(1988), 241–261.

[DIM92] Dickmanns E.D., Mysliwetz B.: Recursive 3d road and relative ego–state recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14(1992), No.2, 199–214.

[DOM98] Domański M.: *Zaawansowane techniki kompresji obrazów i sekwencji wizyjnych*. Poznań, Wydawn. Politechn. Poznańskiej, 1998.

[DRE82] Dreschler L.S., Nagel H.-H.: Volumetric model and 3–d trajectory of a moving car from monocular tv-frame sequences of a street scene. *Computer Graphics and Image Processing*, vol. 20(1982), 199–228.

[EKL94] Eklund M.W., Ravichandran G., Trivedi M.M., Marapane S.B.: Real–time visual tracking using correlation techniques. In: *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1994, 256–263.

[ENK88] Enkelmann W., Kories R., Nagel H.-H., Zimmermann G.: An Experimental Investigation of Estimation Approaches for Optical Flow Fields. In: *Motion Understanding. Robot and Human Vision*, Kluwer Academic Publ., Boston etc, 1988, 189–226.

[FER94] Ferrier N.J., Rowe S.M., Blake A.: Real–time traffic monitoring. *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, 81–88, Los Alamitos, CA, USA, 1994, IEEE Computer Society Press.

[FLE90] Fleet D., Jepson A.: *Computation of component image velocity from local phase information*, *International Journal of Computer Vision*, vol. 5(1990), 77–104.

[FU 82] Fu K.-S.: *Syntactic Pattern Recognition and Applications*, Prentice–Hall, Englewood Cliffs, N.J., 1982.

[FUJ90] Fujimori T., Kanade T.: An approach to knowledge–based interpretation of outdoor natural color road scenes. In: Thorpe, C. (ed.), *Vision and Navigation: The Carnegie Mellon Navlab.*, Chapter 4, Kluwer Academic Publishers.

[FUK88] Fukushima K.: Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, vol 1(1988), 119–130.

[GEN92] Gennery D.-B.: Visual tracking of known three–dimensional objects. *International Journal of Computer Vision*, 7:243–270, 1992.

[GER96] Van Gerven S. : *Adaptive Noise Cancellation And Signal Separation with Applications to Speech Enhancement*, Disertation, Katholieke Universiteit Leuven, Departement Elektrotechnik, Leuven, Belgium, 1996.

[GON87] Gonzalez R.C., Wintz P.: *Digital Image Processing*, Addison-Wesley Publ. Co., Reading MA., 1987.

[GOR97] Gorodnitsky I.F., Rao B.D.: Sparse Signal Reconstruction from Limited Data Using FOCUSS: A Re-weighted Minimum Norm Algorithm, *IEEE Transactions on Signal Processing*, vol. 45(1997), No.3, 600–616.

[GRA93] Graefe V.: Vision for Intelligent Road Vehicles. *Proceedings, IEEE Symposium on Intelligent Vehicles*, Tokyo, 135–140.

[GRI90] Grimson W.: *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, Mass., 1990.

[HAM89] Hammond K.J.: *Case–based Planning*. Academic Press Inc., London 1989.

[HAN78] Hanson A., Riseman E. (Eds.): *Computer Vision Systems*, Academic Press, New York, 1978.

[HAN78a] Hanson A., Riseman E.: VISIONS, A Computer System for Interpreting Scenes, In: A. Hanson, E. Riseman (Eds.), *Computer Vision Systems*, Academic Press, New York, 1978, 303–333.

[HAR79] Haralick R., Shapiro L.: The Consistent Labeling Problem, Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1(1979), No. 2, 173–184.

[HAS86] Hastie T., Tibshirari R.: Generalized additive models. *Statistical Science, 1*(1986), 297–318.

[HAS89] Hastie T., Stuetzle W.: Principal Curves. *Journal of The American Statistical Association*, vol. 84(1989), No.406, 502–516.

[HAY96] Haykin S.: *Adaptive filter theory*. Upper Saddle River, Prentice-Hall, 1996. Series: Prentice Hall Information and System Sciences Series.

[HEE88] Heeger D.: *Optical flow using spatiotemporal filters. International Journal of Computer Vision*, vol. 1(1988), 279–302.

[HOP82] Hopfield J.J.: Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(1982), 2554-2558.

[HUA94] Hu X., Ahuja N.: Mirror uncertainty and uniqueness conditions for determining shape and motion from orthographic projection. *International Journal of Computer Vision*, vol. 13(1994), No. 3, 295–309.

[HUY80] Huyn N., Dechter R., Pearl J.: Probabilistic Analysis of the Complexity of A*. *Artificial Intelligence, 15*(1980), 241–254.

[HWA86] Hwang L., Davis V.-S., Matsuyama T.: Hypothesis Integration in Image Understanding Systems, *Computer Vision Graphics and Image Processing*, vol. 36(1986), 321–371.

[IRA98] Irani M., Anandan P.: A Unified Approach to Moving Object Detection in 2D and 3D Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20(1998), No. 6, 577–589.

[JAI88] Jain R.: Dynamic Vision, *Proceedings of the Int. Conf. on Pattern Recognition (ICPR) 1988*, IEEE Publ., 1988, 226–235.

[JOC96] Jochem T.M.: *Vision Based Tactical Driving*, Carnegie Mellon University, Ph.D. dissertation, CMU-RI-TR-96-14, January 1996.

[JOL86] Jolliffe I.T.: *Principal Component Analysis*. Springer, New York, 1986.

[JUT91] Jutten C., Herault J.: Blind separation of sources, Part I: An adaptive algorithm based on neuromimetic architecture. *Signal Processing*, vol. 24, 1991, 1–20.

[KAN80] Kanade T.: Region Segmentation: Signal vs. Semantics, *Computer Graphics and Image Processing*, vol. 13(1980), 279–297.

[KAK97] Karhunen J., Cichocki A., Kasprzak W., Pajunen P.: On neural blind separation with noise suppression and redundancy reduction, *Int. J. of Neural Systems*, vol. 8 (1997), No. 2, 219–237, World Scientific Publ., London–Singapore.

[KAR94] Karhunen J., Joutsensalo J.: Representation and separation of signals using nonlinear PCA type learning, *Neural Networks,* vol. 7(1994), No. 1, 113–127.

[KAR97] Karhunen J., Oja E., Wang L., Vigario R., Joutsensalo J.: A class of neural networks for independent component analysis, *IEEE Trans. on Neural Networks*, vol. 8(1997), May 1997, 486–504.

[KAR90] Karmann K.-P., von Brandt A.: Moving object recognition using an adaptive background memory. In: V. Cappellini (ed.), *Time-Varying Image Processing and Moving Object Recognition*, Elsevier Science Publ., vol. 2, 1990, 289–296.

[KAS85] Kasprzak W.: Przykład komputera dostosowywalnego a problemy rozpoznawania obrazów *Prace Naukowe Elektronika*, 1985, No.64, 47–79, Warsaw University of Technology Publ., Warsaw.

[KAS86] Kasprzak W.: *Model systemu rozpoznawania obiektów 3–wymiarowych A Model of a 3–D Object Recognition System* (in Polish), Doctoral Dissertation, Warsaw University of Technology, Department of Electronic Engineering, (1986), 125 pages.

[KAS87] Kasprzak W.: A Linguistic Approach to 3–D Object Recognition, *Computers & Graphics*, vol. 11(1987), No.4, 427–443, Pergamon Journals, Londyn, UK.

[KAS89] Kasprzak W.: A Two–Step Modelling Algorithm for Tomographic Scenes. *Mustererkennung 1989*, *Informatik–Fachberichte*, vol. 219, Springer, Berlin, Germany, 1989, 119–123.

[KAS89a] Kasprzak W., Niemann H.: Semantic Networks for Scene Analysis, *Proceedings of the 6-th Scandinavian Conference on Image Analysis*, ISBN 952-90089-0-2, Pattern Recognition Society of Finland, Oulu, Finlandia, 1989, 424–431.

[KAS90] Kasprzak W.: A Meta Control for Knowledge Based Image Sequence Interpretation. *Mustererkennung 1990*, *Informatik–Fachberichte*, vol. 254 (1990), Springer, Berlin etc., Germany, 90–97.

[KAS91] Kasprzak W.: *Suboptimale Graphensuche für wissensbasierte Bildinterpretation.* Final report of AvH–funded research project, University of Erlangen-Nuremberg, Institute IMMD 5 (Pattern recognition), Germany, July 1991, 35 pages.

[KAS93a] Kasprzak W.: Translational Motion or Depth from Segmentation in Image–Time Space. *Research Reports of the Computer Science Institute (IMMD)*, vol. 26 (1993), No. 1, 73–97, University of Erlangen–Nuremberg, Germany.

[KAS93b] Kasprzak W.: Modellunabhängige Schätzung von 3–D Attributen während der Bildfolgensegmentierung, *Mustererkennung 1993*, *Informatik aktuell series*, Springer, Berlin etc., Germany, 1993, 51–58.

[KAS93c] Kasprzak W., Niemann H.: Visual Motion Estimation from Image Contour Tracking, *Computer Analysis of Images and Patterns*. Proceedings, Lecture Notes in Computer Science, vol. 719 (1993), Springer, Berlin etc., Germany, 363–370.

[KAS94] Kasprzak, W.: Road Object Tracking in Monocular Image Sequences Under Egomotion, *Machine Graphics and Vision*, vol. 3(1994), No. 1/2, 297–308, ICS PAS Publ., Warszawa, Poland.

[KAS94a] Kasprzak W., Niemann H.: Moving segment detection in monocular image sequences under egomotion. *Signal Processing VII: Theories and Applications*, EURASIP, Lausanne, 1994, 708–711.

[KAS94d] Kasprzak W., Niemann H., Wetzel D.: Adaptive Estimation Procedures for Dynamic Road Scene Analysis. *Proceedings ICIP–94*, IEEE Computer Society Press, Los Alamitos, CA, USA, 563–567.

[KAS94b] Kasprzak W., Wetzel D., Consecutive Tree Search for Dynamic Road Scene Analysis, *Proceedings of the ANZIIS–94*, IEEE Publ. 94TH8019, Piscataway, NJ, USA, 1994, 312–316.

[KAS95] Kasprzak W.: *Untersuchungen zur Erkennung bewegter Objekte auf der Basis semantischer Netzwerke*, Final Report of DFG–Project Ni-191/8, FORWISS Erlangen, Germany, August 1995, 109 pages.

[KAS95a] Kasprzak W.: Ground plane object tracking under egomotion. *International Archives of Photogrammetry and Remote Sensing*, vol. 30(1995), Part 5W1, 208–213, ISPRS Publ., Zurich, CH.

[KAS96] Kasprzak W., Niemann H.: Applying a Dynamic Recognition Scheme for Vehicle Recognition in Many Object Traffic Scenes, *IAPR Workshop on Machine Vision Applications, MVA'96*, Proceedings. University of Tokyo, 1996, Japan, 212–215.

[KAS96a] Kasprzak W., Cichocki C.: Hidden image separation from incomplete image mixtures by independent component analysis, *Proc. of 13th Int. Conf. on Pattern Recognition (ICPR'96)*, IEEE Computer Society Press, Los Alamitos CA, 1996, vol. II, 394–398.

[KAS96b] Kasprzak W., Cichocki A.: Recurrent least square learning for quasi–parallel principal component analysis, *ESANN'96, European Symposium on Artificial Neural Networks*, D'facto Publ., Brussels, Belgium, 1996, 223–228.

[KAS97] Kasprzak, W.: *Adaptive Erkennung von bewegten Objekten in monokularen Bildfolgen bei Eigenbewegung*. INFIX, St. Augustin, Germany, *DISKI* Series, vol. 172, 1997, 181 pages.

[KAS97a] Kasprzak W., Cichocki A., Amari S.: Blind Source Separation with Convolutive Noise Cancellation, *Neural Computing and Applications*, Springer-Verlag London Ltd., vol. 6(1997), 127–141.

[KAS98] Kasprzak W., Niemann H.: Adaptive Road Recognition and Egostate Tracking in the Presence of Obstacles. *International Journal of Computer Vision*, Kluwer Academic Pub l., Boston/DordrechtŁondon, vol 28(1998), No. 1, 6–27.

[KAS00] Kasprzak W.: Adaptacyjna metoda detekcji obiektów w obrazach cyfrowych. *Raport IAiIS Nr. 00 - 16*, Inst. of Control and Computation Eng., WUT, Warsaw, 2000, 40 pages.

[KAW88] Kass M., Witkin A., Terzopoulos D.: Snakes: Active contour models. *International Journal of Computer Vision*, vol. 1(1988), 321–331.

[KIR93] Kirchner H.: *Bewegungserkennung in Bildfolgen: Ein mehrstufiger Ansatzt*, Deutscher Universitätsverlag, Wiesbaden, 1993.

[KLU90] Kluge K., Thorpe C.: Explicit models for robot road following. In: Thorpe, C. (ed.), *Vision and Navigation: The Carnegie Mellon Navlab.*, Chapter 3. Kluwer Academic Publishers.

[KOH88] Kohonen T.: *Self-Organization and Associative Memory.* Springer-Verlag, New York-Berlin etc., 2nd ed., 1988.

[KOL93] Koller D., Daniilidis K., Nagel H.-H.: Model–based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, vol. 3 (1993), No. 10, 257–281.

[KOR85] Korf R.E.: Depth–First Iterative–Deepening: An Optimal Admissible Tree Search. *Artificial Intelligence*, vol. 27(1985), 97–109.

[KOR90] Korf R.E.: Real–Time Heuristic Search. *Artificial Intelligence*, vol. 42(1990), 189–211.

[KUL72] Kulikowski J.L.: *Cybernetyczne układy rozpoznające.* PWN, Warszawa, 1972.

[KUN91] Kung S., Diamantaras K., Taur J.: Neural networks for extracting pure / constrained / oriented principal components. In: *SVD and Signal Processing*, Elsevier Science, Amsterdam, 1991, 57–81.

[LAI86] Laird J.E., Rosenbloom P.S., Newell A.: Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, vol. 1(1986), 11–46.

[LIP89] Lippmann R.P.: Review of neural networks for speech recognition. *Neural Computation*, vol. 1(1989), 1–38.

[MAH88] Mahanti A., Ray K.: Network Search Algorithms with Modifiable Heuristics. In: Kanal L., Kumar V. (Eds.): *Search in Artificial Intelligence*, Springer Vg., New York-Berlin-Heidelberg, 1988, 200–222.

[MAM94] Mammone R.J.: *Artificial neural networks for speech and vision.* London, Chapman and Hall, 1994. Series: Chapman and Hall Neural Computing Series, 4).

[MAR80] Marr D., Hildreth E.: Theory of Edge Detection, *Proc. B of Royal Society of London*, vol. 207(1980), 187–217.

[MAR77] Martelli A.: On the Complexity of Admissible Search Algorithms, *Artificial Intelligence*, vol. 8(1977), 1–13.

[MAR91] Marshall J.A.: Challenges of vision theory. Self-organization of neural mechanisms for stable steering of object-grouping data in visual perception. *Stochastic and Neural Methods in Signal Processing, Image Processing and Computer Vision*, Proceedings of SPIE, 1569(1991), 200–215.

[MAS92] Masaki I.: *Vision-based Vehicle Guidance.* Springer, New York, Berlin–Heidelberg etc. 1992.

[MAU96] Maurer M., Behringer R., Thomanek F., Dickmanns E.D.: A compact vision system for road vehicle guidance. *13th International Conference on Pattern Recognition*, Vienna, August 1996, 313–317.

[MIC83] Michalski R.S., Carbonell J.G., Mitchell T.M.: *Machine Learning: An Artificial Intelligence Approach*, Tioga, Palo Alto, CA, 1983.

[MIT86] Mitchell T., Keller R., Kedar–Cabelli S.: Explanation–based generalization: A unifying view. *Machine Learning*, vol. 1(1986), 47–80.

[MOL89] Molina R., Ripley B.D.: Using spatial models as priors in astronomical image analysis. *Journal of Applied Statistics, 16*(1989), 193–206.

[MOR96] Moreau E., Macchi O.: High order contrasts for self-adaptive source separation. *Int. Journal of Adaptive Control and Signal Processing*, vol. 10(1996), 19–46.

[MOR90] Morgan A., Dagless E., Milford D., Thomas B.: Road edge tracking for robot road following: a real–time implementation. *Image and Vision Computing*, vol. 8(1990), No.3, 233–240.

[MUR97] Murata N., Mueller K-R., Ziehe A., Amari S-I.: Adaptive On–line Learning in Changing Environments. *Neural Information Processig Symposium, NIPS'96*, MIT Press, Cambridge MA., 1997.

[NAG79] Nagel H.-H.: Über die Repräsentation von Wissen zur Auswertung von Bildern. In: J. Foith (ed.), *DAGM-Symposium, Mustererkennung 1979*, Springer-Vg., Berlin–Heidelberg–New York, 1979, 3–21.

[NAG88] Nagel H.-H.: From Image Sequences Towards Conceptual Descriptions, *Image & Vision Computing*, vol. 6(1988) No. 2, 59–74.

[NAG89] Nagel H.-H.: On a constraint equation for the estimation of displacement rates in image sequences, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11(1989), 13–30.

[NIE81] Niemann H.: *Pattern Analysis*, Springer–Verlag, Berlin–Heidelberg–New York–Tokyo, 1981.

[NIE90] Niemann H.: *Pattern Analysis and Understanding*. Springer–Verlag, Berlin etc., 1990.

[NIS90] Niemann H., Sagerer G., Schröder S., Kummert F.: ERNEST: A semantic network system for pattern understanding, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI–12(1990), 883–905.

[NIB90] Niemann H., Brünig H., Salzbrunn R., Schröder S.: *A Knowledge–Based Vision System for Industrial Applications*, *Machine Vision and Applications*, vol. 3(1990), No. 4, 210–229.

[NIW93] Niemann H., Wu J.-K.: Neural network adaptive image coding. *IEEE Transactions on Neural Networks*, 4:615–627, 1993.

[NIL82] Nilsson N.: *Principles of Artificial Intelligence*. Springer, Berlin etc., 1982.

[OJA82] Oja E.: A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 16:267–273, 1982.

[OJA92] Oja E.: Principal components, minor components and linear neural networks. *Neural Networks*, 5:927–935, 1992.

[OJA97] Oja E.: The Nonlinear PCA learning rule and signal separation – mathematical analysis. *Neurocomputing*, vol. 17(1997), September 1997, 25–45.

[PAJ96] Pajunen P.: *An Algorithm for Binary Blind Source Separation*, Helsinki University of Technology, Lab. of Computer and Information Science, Report A36, July 1996.

[PAO89] Pao Y.-H.: *Adaptive pattern recognition and neural networks*. Reading, Addison-Wesley Publ., 1989.

[PAU93] Pauli J.: *Erklärungsbasiertes Computer–Sehen von Bildfolgen*, Infix, Sankt Augustin, 1993.

[PEA83] Pearl J.: Knowledge versus search: A quantitative analysis using A^*, *Artificial Intelligence*, vol. 20(1983), 1–13.

[PEA84] Pearl J.: *Heuristics. Intelligent Search Strategies for Computer Problem Solving*. Addison–Wesley, Reading, Mass., 1984.

[PLU93] Plumbley M.D.: Efficient information transfer and anti-Hebbian NeuralNetworks. *Neural Networks*, vol. 6(1993), 823-833.

[POL92] Polk A., Jain R.: A parallel architecture for curvature–based road scene classification. In: Masaki, I. (ed.), *Vision–Based Vehicle Guidance*. Springer, New York etc., 1992, 284–299.

[POM91] Pomerleau D.A.: Rapidly adapting artificial neural networks for autonomous navigation. *Advances in Neural Information Processing Systems, 3.* Morgan Kaufmann Publ., San Mateo, CA, 1991, 429–435.

[POM93] Pomerleau D.A.: Neural networks for intelligent vehicles. *Proceedings of IEEE Conf. on Intelligent Vehicles'93*, IEEE Publ., 1993, 19–24.

[QUI86] Quinlan J.R.: Induction of decision trees. *Machine Learning*, vol. 1(1986), 81–106.

[REG94] Regensburger U., Graefe V.: Visual recognition of obstacles on roads. In: *IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, 980–987, Munich, Germany, 1994.

[REI79] Reid D.B.: An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, vol. 12(1979), No.2, 110–120.

[RIS88] Riseman E., Hanson A.: The VISIONS Image Understanding System, In: C. Brown (ed.), *Advances in Computer Vision*, Lawrence Erlbaum Ass. Pub., Hillsdale, N.J., 1988, 6–103.

[ROW98] Rowley H.A., Baluja S., Kanade T.: Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20(1998), No. 1,

[RUM85] Rummelhart D.E., Zipser D.: Feature discovery by competitive learning. *Cognitive Science*, vol. 9(1985), 75–112.

[RUM86] Rummelhart D.E., McClelland J.L.: *Parallel Distributed Processing.* vol. 1 and 2, MIT Press, Cambridge MA, 1986.

[RUT94] Rutkowski L.: *Filtry adaptacyjne i adaptacyjne przetwarzanie sygnałów : teoria i zastosowania*. Warszawa, Wydawn. Nauk. -Techn., 1994.

[SAG90] Sagerer G.: *Automatisches Verstehen gesprochener Sprache*, Bibliographisches Institut, Mannheim, 1990.

[SAL90] Salari V., Sethi I.K.: Feature point correspondence in the presence of occlusion. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12(1990), No. 1, 87–91.

[SAN89] Sanger T.D.: Optimal unsupervised learning in a single–layer linear feed-forward neural network. *Neural Networks*, vol. 2(1989), 459–473.

[SCH92] Schaaser L., Thomas B.: Finding road lane boundaries for vision–guided vehicle navigation. In: Masaki, I. (ed.), *Vision–Based Vehicle Guidance*, Springer, New York etc., 1992, 239–254.

[SCH86] Schunck B.: *The image flow constraint equation*, *Computer Vision Graphics and Image Processing*, Bd. 35, 1986, S. 20–46.

[SCH92] Schwarzinger M., Noll D., von Seelen W.: Object recognition with deformable models using constrained elastic nets. In: S. Fuchs, R. Hoffmann (Eds.), *Mustererkennung 1992*, (Series: Informatik aktuell), Berlin etc., Springer, 1992, 96–104.

[SHA88] Shastri L.: *Semantic Networks. An Evidential Formalization and its Connectionist Realization*. Pitman, London, 1988.

[SIL91] Silva F.M., Almeida L.B.: A distributed decorrelation algorithm. In: E. Gelenbe (ed.), *Neural Networks, Advances and Applications*, North–Holland, Amsterdam, 1991, 145–163.

[SKA95] Skarbek W.: *Metody reprezentacji i kompresji obrazów cyfrowych*. Warszawa, WNT, 1995.

[SKA99] Skarbek W., Pietrowcew A., Sikora R.: *The modified Oja-RLS Algorithm, Stochastic Convergence Analysis and Application for Image Compression.* Fundamenta Informaticae, 36 (1999), 1-21.

[SUB88] Subbarao M.: *Interpretation of Visual Motion. A Computational Study.* Pitman, Morgan Kaufman Pub., London, San Mateo Ca., 1988.

[SWE96] Swets D.L., Weng J.: Using Discriminant Eigenfeatures for Image Retrieval, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18(1996), No. 8, 831–836.

[TAD97] Tadeusiewicz R., Korohoda P.: *Komputerowa analiza i przetwarzanie obrazów.* Kraków, Wydawn. Fundacji Postępu Telekomunikacji, 1997.

[TAN93] Tan T.N., Sullivan G.D., Baker K.D.: Recognizing objects on the ground plane. *Image and Computer Vision*, vol. 12(1993), No. 3, 164–172.

[TAN94] Tan T.N., Sullivan G.D., Baker K.D.: Linear algorithms for multi–frame structure from constrained motion. *BMVC94. Proceedings of the 5th British Machine Vision Conference*, Sheffield, U.K., BMVA Press, 1994, 589–598.

[TAY93] Taylor J.G., Coombes S.: Learning of higher order correlations. *Neural Networks*, vol. 6(1993), 423–427.

[THO93] Thorpe, C. (ed.), *Vision and Navigation: The Carnegie Mellon Navlab.* Kluwer Academic Publishers, 1993.

[TSA84] Tsai R.Y., Huang T.S.: Uniqueness and estimation of three–dimensional motion parameters of rigid objects with curved surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6(1984), 13–27.

[TUR91] Turk M., Pentland A.: Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3:71–86, 1991.

[ULL79] Ullman S.: *Interpretation of Visual Motion*, MIT Press, Cambridge, MA, 1979.

[VAL94] Valentin D., Abdi H., O'Toole A.J., Cottrell G.W.: Connectionist Models of Face Processing: A Survey. *Pattern Recognition*, vol. 27(1994), 1209–1230.

[VAL96] Valentin D., Abdi H.: Can a Linear Autoassociator Recognize Faces From New Orientations?. *Journal of the Optical Society of America A.* vol. 13(1996), .

[VOS88] Voss K.: *Theoretische Grundlagen der digitalen Bildverarbeitung*, Akademie-Verlag, Berlin, 1988.

[WAX88] Waxman A.M., Wohn K.: Image flow theory. A framework for 3–d inference from time-varying imagery. In: Ch. Brown (ed.), *Advances in Computer Vision*, Lawrence Erlbaum Ass. Pub., Hillsdale, N.J., 1988, 165–224.

[WEI91] Weighardt C., Niemann H.: Eine Inferenzkomponente für die Bildsequenzanalyse, In: B. Radig (ed.), *Proceedings. 13. DAGM-Symposium*, (Series: Informatik Fachberichte, vol. 290), Springer, Berlin, 1991, 111–120.

[WEN94] Weng J.: SHOSLIF: A framework for object recognition from images. *Proceedings IEEE ICANN-1994*, IEEE Publ., 1994, 4204–4209.

[WET94] Wetzel, D., Niemann, H., Richter, S.: A robust cognitive approach to traffic scene analysis. *Proceedings of the Second IEEE Workshop on Applications of Computer Vision*, Los Alamitos, CA, USA. IEEE Computer Society Press., 1994, 65–72.

[WID85] Widrow B., Stearns S.D. : *Adaptive Signal Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1985.

[WIN75] Winston P.: Learning structural descriptions from examples. In: Winston P. (Ed.): *The Psychology of Computer Vision.* McGraw-Hill, New York, 1975, .

[WUR88] Wu J., Rink R., Caelli T., Gourishankar V.: Recovery of the 3-d location and motion of a rigid object through camera image. *International Journal of Computer Vision*, vol. 3(1988), 373–394.

[WUE88] Wuensche H.-J.: *Bewegungssteuerung durch Rechnersehen*. Springer, Berlin, 1988.

[XU92] Xu L., Oja E., Suen C.Y.: Modified hebbian learning for curve and surface fitting. *Neural Networks*, vol. 5(1992), 393–407.

[YAN97] Yang H., Amari S.-I.: Adaptive online learning algorithms for blind separation: maximum entropy and minimum mutual information. *Neural Computation*, vol. 9, 1997, 1457–1482.

[ZHA94] Zhang Z.: Token tracking in a cluttered scene. *Image and Vision Computing*, vol. 12(1994), No. 2, 110–120.

Adaptacyjne metody komputerowej analizy sekwencji obrazów cyfrowych

Streszczenie

Celem pracy było sformułowanie ogólnych zasad stosowania *adaptacyjnych technik obliczeniowych* w systemach komputerowej *analizy sekwencji obrazów* cyfrowych oraz implementacja tych zasad dla wybranych problemów analizy, po jednym przykładowym problemie dla każdego poziomu abstrakcji danych. Praca stanowi również uogólnienie i podsumowanie dorobku autora w tej dziedzinie.

We wstępie pracy zaproponowano dwa sposoby klasyfikacji procesów w systemie analizy obrazów - według poziomu abstrakcji danych, na których te procesy operują i według rodzaju danych wejściowych - *skończona* lub *nieskończona* sekwencja obrazów. Wyróżniono hierarchię danych obejmującą pięć poziomów: *sygnałowy, ikoniczny, segmentacji obrazu, rozpoznawania obiektów* i *opisu sceny.* We wstępie zaproponowano również oparcie stosowanej metodologii badań o teorię systemów *konekcjonistycznych* i teorię *dynamicznych systemów.* Pierwsza z nich posiada biologiczną motywację i zajmuje się wypracowaniem mechanizmów łączących sztuczne sieci neuronowe (SzSN) i sieci semantyczne, badając struktury sieci i ich algorytmy uczące oraz mechanizmy aktywacji. Jak pokazują liczne zastosowania sieci neuronowych i sieci semantycznych w problemach analizy obrazów, w tym również szereg prac autora, jak dotąd oba narzędzia są niezależne od siebie. Sieci neuronowe stosowane są głównie do zadań analizy na niskich poziomach abstrakcji danych, podczas gdy sieci semantyczne modelują złożoną analizę symboliczną i interpretacje lingwistyczną obrazu. Teoria *dynamicznych systemów* daje zaś właściwe narzędzia techniczne dla opisu niestacjonarnego środowiska (śledzenie i rekursywna estymacja obiektów).

W ślad za przyjętą metodologią w tej pracy wyróżnia się dwa znaczenia pojęcia *"adaptacyjny"*. W przypadku sekwencji obrazów o ograniczonej długości oczekuje się opisu stacjonarnego środowiska. Dla tej klasy problemów proponuje się w pracy rozwiązanie zwane analizą w *"trybie wsadowym"* oparte na zastosowaniu odpowiednich SzSN i ich algorytmów uczących. W przypadku analizy sekwencji o nieskończonej długości oczekuje się stworzenia opisu niestacjonarnego środowiska (złożonego z ruchomych obiektów). W pracy proponuje się rozwiązanie takich zadań w oparciu o teorię dynamicznych systemów. Mówimy o tzw. *"rekursywnym trybie analizy"*, w którym wyniki dla poprzedniego obrazu (lub N obrazów) są adaptowane do opisu następnego obrazu w sekwencji. W zależności od konkretnego zastosowania traktujemy metody analizy w trybie wsadowym bądź to jako realizujące samodzielne zadanie bądź to jako realizujące specyficzne "pomiary danych" wymagane przez metody rekursywnej analizy.

Pierwsza część pracy obejmująca rozdziały 2.-5 dotyczy zagadnień stosowania algorytmów uczących do analizy sekwencji obrazów w trybie wsadowym. Odnosi się do najnowszych doświadczeń autora w dziedzinie przetwarzania obrazów na niskim poziomie abstrakcji danych: rekonstrukcji obrazu na poziomie sygnału, segmentacji i detekcji ruchu w obrazie oraz kompresji i klasyfikacji obrazów. Metody tego rodzaju posiadają biologiczną motywację i stanowią zwiastuny komputerowych odpowiedników mechanizmu percepcji wizyjnej człowieka.

W rozdziale 2. zaproponowano ogólny schemat analizy sekwencji obrazów w trybie wsadowym, oparty o uczenie w sztucznych sieciach neuronowych. Na wstępie wyróżniono trzy główne struktury sztucznych sieci neuronowych: jedno-warstwowe sieci typu "feed-forward" i liniowe rekurencyjne, wielowarstwowe sieci typu "feed-forward" oraz nieliniowe sieci rekurencyjne. Dla tych sieci podano zasadnicze techniki ich uczenia - uczenie z nadzorem i bez nadzoru. Następ-

nie podano ogólny schemat analizy N obrazów, w którym wyróżniono moduł sterujący i wiele modułów adaptacyjnej analizy (AAM). Pojedynczy AAM posiada cztery układy: sterowanie, blok główny oparty na sieci neuronowej odpowiedniego typu, przetwarzanie wstępne i przetwarzanie końcowe. Omówienie tego schematu analizy uzupełnione zostaje w rozdziałach 3.-5. opisem trzech zrealizowanych przez autora przykładowych implementacji schematu na niższych poziomach danych (na poziomie sygnałowym, ikonicznym i segmentacji).

W rozdziale 3. zaproponowano i przetestowano algorytmy uczące (bez nadzoru) dla SzSN rozwiązujących problem ślepej separacji obrazów z ich mieszanin (BSS), w warunkach gdy nie jest znana dokładna liczba oryginalnych obrazów.
(Problem BSS.) Załóżmy, że istnieje m sygnałów źródeł $s_1(t), \ldots, s_m(t)$ o zerowej składowej stałej i wzajemnie statycznie niezależnych w badanym przedziale czasu. Te oryginalne sygnały $s_i(t)$ nie są znane obserwatorowi, ale znanych mu jest n sygnałów $x_1(t), \ldots, x_n(t)$ będących zaszumionymi liniowo zmieszanymi źródłami (zwykle $n \geq m$). Zadaniem metody BSS jest odtworzenie sygnałów źródeł $\{s_i(t)\}$ (z dokładnością do skali amplitudy) na podstawie ich mieszanin $x_j(t)$ i (zwykle) podanej liczby m ilości źródeł. W postaci wektorowej model (nieznanego obserwatorowi) mieszania źródeł wynosi:

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{n}(t) = \sum_{i=1}^{m} s_i(t)\boldsymbol{a}_i + \boldsymbol{n}(t). \tag{10.1}$$

W pracy podano i przetestowano trzy adaptacyjne metody rozwiązania podstawowego zagadnienia BSS, wykorzystujące techniki uczenia bez nadzoru wag SzSN:
1. Jedno-warstwowa sieć typu "feed-forward" wykorzystująca algorytm uczący separacji globalnej: $\Delta\boldsymbol{W} = \eta\left[\boldsymbol{I} - \boldsymbol{f}(\boldsymbol{y})\boldsymbol{g}(\boldsymbol{y})^T\right]\boldsymbol{W}$.
2. Wielokrotna warstwa typu "feed-forward" dla prostego algorytmu uczącego separacji lokalnej: $\Delta\boldsymbol{W} = \pm\eta\left[\boldsymbol{I} - \boldsymbol{f}(\boldsymbol{y})\boldsymbol{g}(\boldsymbol{y})^T\right]$.
3. Dwu-warstwowa sieć typu "feed-forward" z pierwszą warstwą realizującą kompresję i ortogonalizację ($\boldsymbol{v} = \boldsymbol{V}\boldsymbol{x} = \boldsymbol{V}\boldsymbol{A}\boldsymbol{s}$) oraz drugą warstwą realizującą separację $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{v}$ z regułą uczącą tzw. nieliniowego PCA: $\Delta\boldsymbol{W} = \eta\boldsymbol{f}(\boldsymbol{y})\left[\boldsymbol{v}^T - \boldsymbol{f}(\boldsymbol{y})^T\boldsymbol{W}\right] \simeq \eta\left[\boldsymbol{f}(\boldsymbol{y})\boldsymbol{y}^T - \boldsymbol{y}\boldsymbol{f}(\boldsymbol{y})^T\right]\boldsymbol{W}$.

Obok podstawowego zagadnienia, dla którego addytywny szum jest mały w porównaniu z sygnałem użytecznym, zaproponowano modele separacji i usuwania dużego szumu o charakterze konwolucyjnym. W podstawowym modelu wymagane jest jednoczesne uczenie wag sieci neuronowej $\boldsymbol{W}$ i zespołu filtrów typu FIR $\boldsymbol{H}(z)$ służących do usuwania szumu. Sygnał wyjściowy dany jest wzorem:

$$\boldsymbol{y}(t) \quad = \quad \boldsymbol{W}\boldsymbol{x}(t) - \boldsymbol{H}(z)n_R(t), \tag{10.2}$$

gdzie $\boldsymbol{H}(z) = [h_1(z), ..., h_n(z)]^T$ zawiera wektory współczynników $h_i(z) = h_{i0} + h_{i1}z^{-1} + h_{i2}z^{-2} + ... + h_{iM}z^{-M}$, a z^{-1} jest jednostkowym opóźnieniem. W tym modelu zakłada się pomiar szumu $n_R(t)$ będącego obrazem konwolucyjnym (splotem) nieznanego szumu środowiska $\nu_R(t)$, zakłócającego także sygnały wejściowe:

$$n_R(t) = \sum_{j=0}^{N_R} b_{Rj}\nu_R(t - jT) = \boldsymbol{b}_R(z)\nu_R(t). \tag{10.3}$$

Podczas uczenia wag $\boldsymbol{W}$ stosuje się jedną z trzech reguł separacji, zaś dla nauczenia wag $\boldsymbol{H}$ zaproponowano wykorzystanie adaptacyjnej reguły "delta".
Na koniec rozdziału 3. podano przypadki szczególne i zaproponowano sposób rozwiązania problemu w sytuacji niedomiaru mieszanin (tzw. problem BSE).

W rozdziale 4. zaproponowano i przetestowano algorytmy uczące dla problemów kompresji obrazów i klasyfikacji obrazów (na poziomie ikonicznym analizy). Zdefiniowano przekształcenie *DKL* - wektorowego sygnału wejściowego $\boldsymbol{X}(t)$ w wyjściowy $\boldsymbol{Z}(t)$ jako:

$$\boldsymbol{Z} = \boldsymbol{DWX}, \tag{10.4}$$

gdzie przekształcenie *KL* jest wynikiem kompresji typu *PCA* lub *PSA*: $\boldsymbol{Y} = \boldsymbol{WX}$, a następne przekształcenie *D* jest klasyfikacją według analizy dyskryminacyjnej *DA*: $\boldsymbol{Z} = \boldsymbol{DY}$. W celu kompresji obrazów metodami PCA i PSA (tzw. komponentów i podprzestrzeni głównych) zaproponowano i zrealizowano efektywne i dokładne neuronowe algorytmy uczące (bez nadzoru). Dla analizy PCA powstała metoda CRLS PCA, która w odróżnieniu od znanej metody *Sangera* (wykorzystującej tzw. regułę *Oja*):
(1) wylicza składowe główne po kolei pobierając resztkowy sygnał wejściowy $\boldsymbol{e}_j$ zamiast oryginalnego wejścia $\boldsymbol{x}(k)$ (deflacja sygnału wejściowego): $\boldsymbol{e}_j(k) = \boldsymbol{e}_{j-1}(k) - y_j(k)\ \boldsymbol{w}_j$,
(2) parameter szybkości uczenia $\eta_j^{(-1)}$ jest każdorazowo inicjalizowany odpowiednio do spodziewanej wariancji sygnału resztkowego: $\eta_j(0) = \sigma^2[\boldsymbol{e}_{j-1}]/N = E[y_{j-1}^2]$,
(3) parametr η_j ma charakter adaptacyjny - zanika według metody RLS: $\eta_j(k) = y_j(k)^2 + \eta_j(k-1)$.
Następujący po redukcji przestrzeni reprezentacji etap klasyfikacji wektora realizowany jest dzięki neuronowej metodzie analizy dyskryminacyjnej - wykorzystującej algorytm uczenia kwantyzacji wektorowej z nadzorem. Pojedynczy krok uczący polega na wybraniu dwóch wyjść k, l posiadających wektory wag wejściowych $\boldsymbol{w}_l$, $\boldsymbol{w}_l$ najbliższe aktualnej próbce na wejściu $\boldsymbol{x}^p$ i porównaniu etykiet reprezentowanych przez nie klas C_k i C_l z etykietą d^p aktualnego wejścia oraz zastosowanie reguły: jeśli $C_k \neq d^p$ i $d^p = C_l$ to

$$\boldsymbol{w}_l(t+1) = \boldsymbol{w}_l(t) + \eta[\boldsymbol{x}^p(t) - \boldsymbol{w}_l(t)], \quad \boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t) - \eta[\boldsymbol{x}^p(t) - \boldsymbol{w}_k(t)]. \tag{10.5}$$

Oznacza to przesunięcie właściwego wektora wag $\boldsymbol{w}_l$ "w kierunku" wektora wejściowego, zaś odsunięcie od $\boldsymbol{x}^p$ wektora wag $\boldsymbol{w}_k$ związanego z niewłaściwą klasą.

Rozdział 5. dotyczy zagadnień detekcji ruchu w płaszczyźnie i klasyfikacji segmentów obrazu w oparciu o mapę ruchu (poziom segmentacji obrazu). Zaprezentowano implementacje kilku detektorów ruchu na poziomie pikseli obrazu. Zaproponowano także algorytm relaksacyjny dla nieliniowej sieci rekurencyjnej, w celu estymacji ruchu w obrazie. Metody detekcji i estymacji ruchu w obrazie mają zasadniczo zastosowanie w analizie sekwencji pochodzącej z nieruchomej kamery. Przetestowano również dokładność tych metod dla obrazów pochodzących z ruchomej kamery. W celu wykorzystania mapy ruchu dla segmentacji obrazu na ruchome obiekty i nieruchome podłoże zaproponowano dwie metody korekcji geometrii obrazów.

W drugiej części pracy, obejmującej rozdziały 6.-9., skoncentrowano się na analizie sekwencji obrazów o nieskończonej długości. Odzwierciedla ona dorobek autora w zakresie modelowania analizy sekwencji obrazów - sposoby reprezentacji wiedzy i strategie sterowania analizą oraz zagadnienie śledzenia i estymacji stanu systemów dynamicznych - a także doświadczenie w tworzeniu konkretnego systemu przeznaczonego do dynamicznej analizy obrazów ruchu drogowego.

W rozdziale 6. zdefiniowano schemat rekursywnego rozpoznawania obiektów dla analizy w trybie rekursywnym sekwencji obrazów o nieskończonej długości. Jest to najbardziej dojrzała wersja schematu, rozwijanego przez autora podczas swoich prac nad rozpoznawaniem sekwencji obrazów. Jedynie elementy tego schematu publikowane już były w pracach autora. Wyróżniono dwa stopnie złożoności modelu rozpoznawania obiektów: (1) śledzenie pojedynczego dynamicznego obiektu i (2) poszukiwanie zgodnego opisu obrazu przy jednoczesnym występowaniu wielu obiektów dynamicznych.

Zagadnienie śledzenia pojedynczego obiektu sprowadza się do dokonania właściwego *pomiaru* występowania obiektu w pojedynczym obrazie i do odświeżenia estymowanego wektora stanu obiektu. Pokazano, że może tu znaleźć zastosowanie jeden z dwóch rekursywnych estymatorów: (a) zgodny z rozszerzonym filtrem Kalmana dla nieliniowych systemów, gdy można jednoznacznie określić wiarygodność każdego pomiaru, lub (w przeciwnym razie) (b) filtr minimalizujący różnicę (błąd) między pomiarem a estymacją stanu zgodnie z zasadą minimalizacji błędu kwadratowego RLS.
Z uwagi na nieuchronną niedokładność przypisania pomiaru w obrazie do modelu obiektu konieczne jest tworzenie wielu hipotez danego obiektu. W sytuacji, gdy z założenia w obrazie może wystąpić tylko pojedynczy obiekt, wybór najlepszej hipotezy nie nastręcza trudności. Sytuacja zmienia się, gdy w obrazie może pojawić się z góry nieznana liczba obiektów. Potrzebny jest dodatkowy krok analizy - poszukiwanie największego zgodnego podzbioru hipotez obiektów. Sytuację dodatkowo może skomplikować fakt istnienia modeli obiektów o złożonej strukturze, tzn. obiektów dla których wyróżniono pod-obiekty (np. obiekt "koło" jest podobiektem "pojazdu kołowego") lub obiekty bardziej specjalizowane (np. "samochód osobowy" jest wersją specjalizacją obiektu "samochód"). Dlatego też podano w pracy ogólne rozwiązanie wspomnianego kroku analizy w postaci algorytmu optymalnego przeszukiwania przestrzeni (grafu) możliwych rozwiązań częściowych, reprezentujących możliwe dopasowania segmentów obrazu do obiektów zawartych w bazie modeli systemu. W pracy zaproponowano i opisano kilka algorytmów optymalnego przeszukiwania grafów, zwanych A^*, B, C, D.

W końcowej części pracy, w rozdziałach 7. - 9., opisane są projekty trzech zrealizowanych implementacji powyższego schematu rozpoznawania obiektów w ramach systemu analizy obrazów ruchu drogowego w warunkach ruchu własnego. Omówiono w pracy metody zrealizowane przez autora, zwracając szczególną uwagę na ich implementacje ogólnego schematu rozpoznawania obiektów. Te trzy aplikacje dotyczą trzech wyższych poziomów abstrakcji danych systemu analizy (segmentacja, rozpoznawanie obiektu i opis sceny).

W rozdziale 7. zaimplementowano generalny schemat rozpoznawania dla śledzenia 2-wymiarowych segmentów w obrazie i rozpoznawania punktu zbieżności drogi w płaszczyźnie obrazu (poziom segmentacji obrazu).

W rozdziale 8. zaimplementowano schemat rozpoznawania w dwóch rozwiązaniach opartych na modelach rozpoznawanego zakresu świata - rozpoznawanie drogi i ruchomych 3-wymiarowych obiektów (poziom rozpoznawania obiektów)

Trzecia implementacja schematu rozpoznawania obiektów ma miejsce na poziomie interpretacji scen (rozdział 9). Dla generacji opisów scen ruchu drogowego zaproponowano aplikacyjny system z bazą wiedzy oparty o reprezentacje proceduralnych sieci semantycznych. Zdefiniowano system użytkowy w specyficznym systemie-matce ERNEST, po uzupełnieniu go o nadążny algorytm sterowania analizą sekwencji obrazów. W tym celu bazowy algorytm sterowania rozszerzono o mechanizm przenoszenia wyników analizy i uwzględniono nową regułę inferencji zwaną *odświeżaniem*, realizującą modyfikację i predykcję śledzonych obiektów. Następnie zaproponowano sieć semantyczną reprezentującą model opisywanego świata. Pojęcie obiektu w takim systemie w sposób naturalny odnosi się do egzemplarzy jednostek opisu, generowanych podczas analizy.

Contents