**Forschungsberichte der
Technischen Fakultät
Abteilung Informationstechnik**

A System for the Content-Based Retrieval of Textual and Non-Textual Documents Based on Natural Language Queries

A. Knoll, I. Glöckner, H. Helbig, S. Hartrumpf

Report 97-03

Universität Bielefeld • Postfach 10 01 31 • 33501 Bielefeld • FRG

Impressum:

Herausgeber:
Robert Giegerich, Alois Knoll, Helge Ritter, Gerhard Sagerer,
Ipke Wachsmuth

Anschrift:
Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31, 33501 Bielefeld

ISSN 0946-7831

Contents

Abstract

A system for the content-based querying of large databases containing documents of different classes (text, images, image sequences, audio signals) is introduced.[1] *Queries are formulated in natural language and are evaluated for their semantic contents. For the document evaluation, a knowledge model consisting of a set of domain specific concept interpretation methods is constructed. Thus, the semantics of both the query and the documents can be interconnected, i.e. the retrieval process searches for a match on the semantic level (not merely on the level of keywords or global image properties) between the query and the document. Methods from fuzzy set theory are used to find the matches. Furthermore, the retrieval methods associate information from different document classes (texts, images, . . .). To avoid the loss of information inherent to pre-indexing, documents need not be indexed; in principle, every search may be performed on the raw data under a given query. The system can therefore answer every query that can be expressed in the semantic model. To achieve the high data rates necessary for on-line analysis, dedicated VLSI search processors are being developed along with a parallel high-throughput media-server. A further speed-up is achieved by a mediator module which maintains an intelligent result cache. This caching mechanism plays a similar role in HPQS as that of pre-computed indexes in more traditional retrieval systems. In the report we outline the system architecture and detail specific aspects of the individual modules. Preliminary results, a brief overview of the current project status and a short discussion of further issues conclude the report.*

[1]Parts of the work reported here were funded by the ministry of science and research of the state of Northrhine-Westphalia within the collaborative research initiative “Virtual Knowledge Factory”. The group developing the HPQS system includes working groups at the universities of Aachen (T.G. Noll), Bielefeld (A. Knoll), Dortmund (J. Biskup), Hagen (H. Helbig) and Paderborn (B. Monien). HPSQ is an acronym for “High Performance Query Server”.

1 Introduction

The recent advances in the fields of the electronic publishing and archive technology have fostered an enormous growth in the number of documents available in electronic form. Their sheer number makes it difficult, if not impossible, to keep up with the abundance of information, i.e. to enable an effective, efficient and accurate retrieval. Up to now, supplementing a document with textual indexing or “meta”-information, e.g. according to the “Dublin core” or “Warwick frame” standards is the only way of characterising documents for retrieval. Frequently, however, this meta-information is incomplete and does not lend itself to correctly identifying the stored documents the user wants to be provided with as the result of his query. Furthermore, successful retrieval based on the matching of keywords requires the user to formulate his query in a combination of words and operators, which more or less precisely circumscribe what otherwise he would have no difficulties to express in natural language. The problem of queries which use semantically adequate keywords with regard to the information wanted but not with regard to the descriptions really used with the target document, may be alleviated by expanding the single user query into many by looking for synonyms in a thesaurus. A search based on index systems or on a thesaurus results in relatively unspecific queries and hence in a very high number of matches – as painfully witnessed by users of the popular WWW search engines. Even when the user reformulates his query by excluding some of the thesaurus' entries, the number of irrelevant matches is usually high.

With non-textual documents, e.g. still images, graphics, or technical drawings, the situation is even worse: while a text normally centers around a few concepts that can be distilled into keywords, a picture normally contains a multitude of different objects. It is not the presence of the objects but very often their relations that constitute the interesting part of the image content (“A car *in front* of the house”). Moreover, depending on the context, the objects bear different names (*crowd, humans, women,...*) that are not synonymous. In other words: For domains other than texts it is generally very difficult to find appropriate textual descriptors (indexes) and the likelihood for retrieval misses under a given query (e.g. if objects were not registered in the index) is very high.

A small number of prototypical systems have been reported in recent literature that aim at retrieving images from a database. QBIC [1] retrieves colour images. It looks for regions whose characteristics can be specified by the user in terms of colour, texture and shape. It uses numerical index keys computed from colour distribution and colour histograms. The VIR system [2] determines the similarity between images by colour, composition, texture and structure and retrieves pictures that are similar with respect to these parameters. The user can steer the retrieval process by giving individual weights to each of them. Furthermore, there have been attempts to make image contents tractable by special query languages, or to formulate the query in pictorial form, e.g. as a rough sketch. Examples of these kinds of systems are described in [3] and in [4]; the latter gives a comprehensive list of references. Earlier work has been reported in [5].

All of these systems perform a pattern match at the level of pixels or regions. They do not aim at gaining some understanding of the image, which is necessary to handle queries relating to image content on the semantic level. In other words: they can find small red dots in images but they do not understand the concept of a “flower”. To the best of our knowledge IRIS [6], now commercially available as MediaMiner [7], is the only system that accepts keywords which are interpreted as concepts describing the queries for image contents. It transforms pictures into descriptive text indexes and performs the searches on these texts. Specific application domains other than searches for images of landscape type and CAD images have not been reported in the literature.

The systems that comes closest to ours in terms of knowledge representation and content evaluation (on a different domain) is Multos [8]. It was designed to retrieve documents in office environments with special emphasis on graphical images; a coupling with a real-world image interpretation module and a fusion of document contents for controlling the retrieval process was not implemented.

To allow queries dealing with the semantics of different document classes, to enable queries to be formulated in natural language, to exploit references between documents and fusing their information content as well as to support on-line queries obviating the need for complete pre-indexing, we have developed an architecture for a High Performace Query Server (HPQS) which is outlined in the following section.

1.1 HPQS approach

To cover the complete search cycle starting from the query input in natural language by means of recognising the semantic content of queries and documents to postproceessing and delivery of the results, our approach combines the following methods and features into a consistent architecture:

- Formulation of queries in natural language (NL);
- Transformation of NL queries into formal database queries using operators from fuzzy set theory;
- Extraction of *semantic* information from both queries and documents (as opposed to pure keyword search);
- Matching of query and database contents on the semantic level;
- Utilisation of references between the images and the text of a single document or between documents;
- Automatic pre-indexing and structural enhancement of text and image information.
- Support of run-time search on raw data (i.e. non-pre-indexed documents) using dedicated high-speed VLSI search processors.

Establishing a semantic match between the query and one or more documents in the database presupposes *meta knowledge* by means of which both documents and questions can be formally interpreted. This meta knowledge is obviously domain dependent: queries over a database of medical documents (e.g. patient files) require a model that is different from the model permitting queries over a data base of technical data sheets.

The need of a model for meta knowledge does not imply, however, that a *full understanding* of the documents is necessary. It is sufficient to know that the user query can be transformed into concepts that find their corresponding matches in the model instantiations of the documents. This obviates the need for implementing full-blown text understanding and image-understanding subsystems.

Matching concepts in NL-queries and documents is not only a departure from classical deterministic and probabilistic retrieval techniques that essentially match keywords; it is the only way of unifying the different information representations within one document or across a multimedia document database. Today, such databases usually contain texts and pictures; in the near future they will also contain audio and video signals.

Before introducing the system architecture, we take a brief look at our document domain.

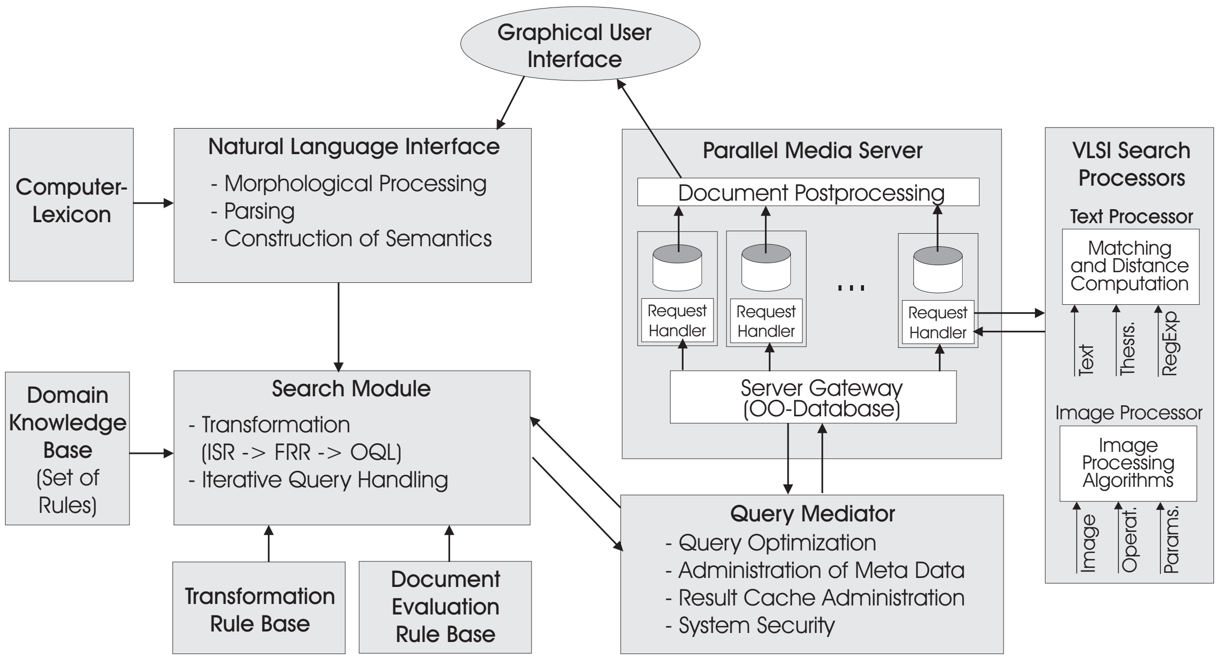

Graphical User Interface
Computer-Lexicon
Natural Language Interface
- Morphological Processing
- Parsing
- Construction of Semantics
Parallel Media Server
Document Postprocessing
Request Handler
Request Handler
...
Request Handler
Server Gateway (OO-Database)
VLSI Search Processors
Text Processor
Matching and Distance Computation
Text
Thesrs.
RegExp
Image Processor
Image Processing Algorithms
Image
Operat.
Params.
Domain Knowledge Base (Set of Rules)
Search Module
- Transformation (ISR -> FRR -> OQL)
- Iterative Query Handling
Query Mediator
- Query Optimization
- Administration of Meta Data
- Result Cache Administration
- System Security
Transformation Rule Base
Document Evaluation Rule Base

Figure 1: Architecture of the HPQS

1.2 Query and document domain

For the purpose of testing and practical demonstrations of the system but also for guiding the design and implementation, a prototypical document domain had to be chosen carefully. After exploring the potential of several alternatives, we picked the domain of meteorological data/documents: i.e. all kinds of printed weather reports (short, mid and long-term) containing text, graphics, images and large amounts of measurement data. This may be supplemented (at a later stage of the system development) by TV video and radio weather reports as well as by collections of data from the national weather services.

The range of sensible queries is large, starting from simple requests for documents about the current weather, through queries about weather developments in the past fulfilling certain constraints, to questions about special weather conditions in certain locations at an arbitrary point in time. Sample queries that can be answered are:

- *Show me a map of the region with the hottest temperature readings yesterday!*
- *What additional weather data do you have about that region?*
- *Show me weathermaps with a similar temperature distribution!*
- *Can you show me a satellite image of upper Bavaria that has a 50% coverage of clouds?*

Weather data have the advantage that they are – to some extent – comprehensible for the non-meteorologist also. The correct operation of the system may thus be verified easily even by the layman.

2 Overview of the system design

The user communicates with the system by means of a graphical user interface, which allows to type in queries and displaying results in a reasonable, i.e. ergonomically adequate form. The query text is fed into the NL-analyser (see fig. 1), which subjects it to a syntactic-semantic transformation comprising a morphological analysis, parsing of the lemmatized sentence and constructing a semantic graph. The whole process is supported by a (partially domain-specific) computer lexicon. The results of the syntactic-semantic analysis are represented by means of the multilayered semantic network formalism MESNET [9, 10]. The MESNET graph is then transformed into the Intermediate Semantic Representation ISR, which contains information about query classes, speech act types, attributes, relations and temporal characteristics.

The ISR expressions serve as input to the search module, which firstly matches them with rules from the transformation rule base. The premises of these rules recognise subexpressions of the ISR (temporal and local specifications, domain concepts,...); their consequents encode corresponding expressions in the so-called Formal Retrieval Representation FRR. The latter provides representational means for expressing fuzzy operators. The FRR expressions then trigger appropriate methods for evaluating the database documents under the given query by applying fuzzy search techniques (the available evaluation methods are specified in the document evaluation rulebase). This evaluation, in turn, triggers preliminary retrievals from the database. The investigation of these documents results in one (or more) documents to be selected as the answer to the query. It may give rise to further queries as well. Thus, the main task of the retrieval module is the generation of formal queries to the database, the management of interrelated queries and the evaluation of results. The retrieval module also supports user queries which refer to an underlying domain model (of meteorology, in our case) which combines information extracted from associated documents. This domain knowledge is provided by the domain knowledge base.

To keep the interface between the database and the search module manageable, the module does not communicate with the database directly but via the Multimedia Query Mediator (MQM). The MQM accepts queries formulated in ODMG-OQL, optimizes them and forwards them to the Media Server Gateway. It receives the query results from the Media Server and sends them back to the search module according to the CORBA object exchange standard. Furthermore, it keeps a record of meta data, handles security and user privileges and maintains the consistency of a result cache.

Mass data, i.e. documents and their corresponding evaluation methods are dealt with on the Parallel Media Server. The gateway to the server essentially consists of an O_2 database that (upon requests from the mediator) generates requests to 16 parallel processor nodes, which control both the storage devices and the VLSI search processors by analysing the requests from the gateway. Issues here are throughput optimisation, load balancing and coherency maintenance. If a query requests computationally intensive operations, then a processor node forwards it to the VLSI processors, otherwise it is executed directly.

The VLSI search processors perform searches on texts. Word specifications according to the ANSI standard Z39.58 enriched by operators for approximate and associative search are transformed into a string of processor instructions. The processor and its storage technology speeds up text search by a factor of up to 1000. About the same speed improvement over software solutions is achieved by the image processors. They realise standard low-level and intermediate-level image processing techniques on colour images (and later image sequences). In their final version a single processor will be able, for example, to filter more than 25 full-resolution HDTV-images per second.

In the following sections we describe the design of the five main modules of the HPQS system in more detail.

3 The natural language interface (NLI)

3.1 The case for an NL interface

The acceptance of a multimedia system depends crucially on the design of its user interface. Ideally, the user interface should totally hide the complexity of the program, thus providing the view of an easy-to-use “information assistant”. Furthermore, the NLI must be well adapted to the needs of “naive users” who may be competent specialists in their field, but who do not know or do not want to learn the peculiarities of retrieval techniques or languages. The usual way of querying information systems in terms of keywords and Boolean connectives does not meet these requirements. It is not well suited to querying in a multimedia domain either. There are several reasons for that:

- *Absence of keywords.* First of all, there are no words in the images the user looks for, and hence keyword matching is not as easy as in the usual case of purely textual documents. In the framework that we envision, descriptors are assigned automatically by means of application-dependent methods operating on the multimedia documents. These image analysis methods typically provide *more* than keywords; e.g. local relations between regions of interest can be computed along with a description of the regions themselves. This relational information plays a crucial role in image interpretation. E.g., it may make a difference to users planning their vacation in Italy whether “there are clouds” on the current weather image or whether “there are clouds in Italy”. Keywords and boolean connectives, however, are obviously not sufficient for expressing such structured, model-based information.

- *Semantic querying.* Keywords with Boolean connectives are a rather cumbersome way of specifying a user's search request. Even in the textual case, users are seldom interested in documents in which only some search keywords happen to occur. Put succintly, texts are more to humans than a set or distribution of word occurrences; they are pieces of natural language with an associated meaning and informational content which may or may not fulfill a user's information needs. Ideally, users should be permitted to query an information system on the level of meaning. In other words, the whole process of querying and result presentation must be embedded in the user's system of mental concepts. In practise, it might be a long way until these ambitious goals can be accomplished. A clear benefit of using an NL-interface is the focus it puts on research efforts as it *forces* us to take seriously the issue of conceptual, semantic querying.

- *Intuitive interface.* Another problem with the usual query interfaces is their lack of user-friendliness. Although some of these interfaces (e.g. GLIMPSE [11]) offer powerful querying mechanisms like approximative search, regular expression search, adjacency operators and search in specific document fields, most users are presumably not able to take full advantage of these features because they do not know when to apply them. By contrast, natural language provides an intuitive interface because everybody knows how to use his native language without effort. Therefore, providing an NL front-end not only relieves the user from learning yet another querying language, but also removes technical barriers in accessing the more advanced features of an information retrieval system.

3.2 The NatLink interface

The NL-interface *NatLink* aims at the construction of adequate semantic representations for a broad class of acceptable queries. In contrast to many linguistically motivated grammar systems, emphasis is laid on the semantic acceptablility of NL input, not on its grammaticality. In particular, the robustness issue plays an important role. This includes the proper treatment of unknown words as well as that of elliptic sentences.
Robustness of NL analysis conflicts to some extent with the goal of generating rich and deep semantic representations. For example, if a word is not found in the lexicon, morphologic analysis and syntactic context will usually provide very shallow information about the semantics of the given word. In the approach taken here, the depth of semantical analysis dynamically adapts to the syntactic and semantic information being available, resulting in a trade-off between robustness and depth of semantic representations. A prerequisite of this approach is that the target formalism supports semantic representations on different levels of granularity or specificity. MESNET [10, 12] a multi-layered extension of semantic networks, has been designed to fulfill these requirements. Due to its multidimensional structure of classificatory knowledge, it is also possible to handle generic and individual concepts in MESNET as well as intensional or extensional aspects of meaning. MESNET has proved useful as

- the semantic representation for computer lexica [13];
- the target language for NL analysis [14, 15];
- the basis for a translation into formal queries to information retrieval systems [16, 17].

$$\left[\begin{array}{ll}
\textit{word} & \\
\textsc{morph} & \left[\begin{array}{ll} \textit{morph} & \\ \textsc{base} & \textit{Wetter} \\ \textsc{root} & \textit{Wetter} \end{array}\right] \\
\textsc{syn} & \left[\begin{array}{ll} \textit{n-syn} & \\ \textsc{agr} & \left[\begin{array}{ll} \textit{np-agr} & \\ \textsc{num} & \textit{sg} \\ \textsc{gend} & \textit{neut} \end{array}\right] \end{array}\right] \\
\textsc{semsel} & \left[\begin{array}{ll} \textit{semsel} & \\ \textsc{sem} & \left[\begin{array}{ll} \textit{sem} & \\ \textsc{entity} & \left[\begin{array}{ll} \textit{nonment-stat-abs-situation} & \\ \textsc{proper} & \textit{noproper} \end{array}\right] \\ \textsc{lay} & \left[\begin{array}{ll} \textit{lay} & \\ \textsc{card} & \textit{nocard} \end{array}\right] \end{array}\right] \\ \textsc{c-id} & \textit{wetter.1} \\ \textsc{select} & \langle\rangle \\ \textsc{compat-r} & \{\textit{trel lext loc circ ctxt conf}\} \end{array}\right] \\
\textsc{entry} & \textit{1}
\end{array}\right]$$

Figure 2: Example of a typical lexical entry; here for the German noun "Wetter"

The natural language input is analyzed according to the principles of the *word-class controlled functional analysis* (WCFA) introduced by Helbig [18]. Like other word-oriented approaches [19, 20], WCFA supports incremental parsing which results in an improved robustness of the system.
The *word-class functions* (WCFs), on which WCFA is based, roughly correspond to the traditional

parts of speech, but are usually more fine-grained (they comprise for instance common nouns, proper names, nominal interrogatives, attributive interrogatives etc.). All morpho-syntactic and semantic information relevant to the analysis process is stored in the computer lexicon. An example for a lexical entry is given in fig. 2. The words in the lexicon are hierarchically ordered, ensuring re-use and extensibility of information by means of multiple inheritance with defaults [21].
WCFA parsing is *expectation oriented*. After morphological and lexical processing of a word, an "opening act" is performed which generates syntactic and semantic expectations (valency, agreement etc.). The WCF also specifies "completing acts", which perform the saturation or refinement of the expectations stipulated during the opening act. When performing a "closing act", the analysis of the current constituent is completed and may then be used by the WCF parser to fill other expectations.
In NatLink, semantic representations of so-called *semantic kernels* (e.g. for noun phrases, prepositional phrases) are constructed as soon as possible during analysis in order to resolve syntactic ambiguities in an early processing phase. Semantic kernels constitute the minimal units to which a semantic representation can be assigned in the course of incremental parsing.
NatLink is *modularly structured* in that the process of NL analysis is decomposed into different stages. It covers a large fragment of German syntax relevant for natural language access to data bases while lexically concentrating on that part of German which corresponds to the given application domain. In the next project phase, the lexical coverage will be extended and NatLink will be equipped with an appropriate discourse model.

4 Semantic search module

The *semantic search module* translates ISR query representations generated by the NL interface into OQL queries, which are then passed to the mediator module for evaluation. Perpendicular to this flow of transformations, the natural language concepts of the ISR must be given an operational interpretation. For this purpose, direct search methods are supplied for each concept in order to determine its (fuzzy) extension based on the raw document data. Due to the methods relying on fuzzy set theory the gradual correspondence between NL concepts and available modelling tools (database attributes, direct search methods) can be expressed. In addition, information from different document domains can be aggregated. For performance reasons, these methods are implemented and actually executed in lower layers of the system, namely on the parallel media server and (whenever possible) on the VLSI hardware.

4.1 The formal retrieval representation (FRR) level

As Fuhr [22] observes, information retrieval (IR) differs from database querying not only in that the objects involved are far less structured, but also by the fact that with IR the relevance of an object with respect to a given query must be treated as an inherently gradual relation not expressible by Boolean logic used for database queries. With NL input, there are additional sources of gradual information. Therefore, the inherent fuzziness or vagueness of natural language concepts [23] has to be dealt with on the query side ("Show me some…" vs. unspecific queries as "What was the weather like?"). On the interpretation side, knowledge about the constituting features of the concepts in the query is often incomplete. This results in a partial mismatch between the semantic representation of the question and the target concept. Instead of postulating "strict" methods using

Boolean evaluations, it seems more natural to represent such knowledge by fuzzy rules and judge the degree of applicability of these rules by means of gradual evaluations. Consequently, the well-structured view of the multimedia database system provided by the mediator component must be augmented by additional means for representing vague queries and gradual search results. In order to meet these requirements, a formal retrieval representation (FRR) was developed, which extends the object query language connectives from fuzzy logic and operators for information aggregation and data fusion. These operators can be applied in order to establish content-related links between associated documents and to combine information extracted from different document sources or by different extraction methods—tasks which are crucial to the management of the various levels of abstraction within our system. The FRR is implemented by appropriate classes and methods in the object-oriented database schema.

4.2 Query translation step

The interpretation process can be decomposed into a translation step (resulting in queries on the FRR level) and a subsequent query execution step. The translation process includes

- *normalisation*, mapping of terms to their database correlates (e.g. names of cities are mapped to geographic identifiers)
- *anchoring in discourse situation* (e.g. resolution of temporal deictic expressions like "to-day")
- *default assumptions* whenever needed, e.g. in order to limit the scope of a region search to Europe;
- *user interaction* in order to validate decisions on alternative interpretations.

The transformation is accomplished by application-specific transformation rules which construct the query FRR from the ISR graph. The premise part of each transformation rule specifies the structure of subgraphs to which the rule applies (e.g. temporal or local specifications, domain concepts). The consequent parts of the rules provide the corresponding FRR expressions from which the query FRR is constructed. This basic approach will be further elaborated along the lines of Rayner [24, 25] in order to enhance the treatment of incomplete queries which only partly express the user's information needs.

4.3 Generic FRR methods

The generic part of FRR consists of domain-independent methods from which direct search methods for the concepts of a given domain can be built, including

- *a sophisticated text-search module*, which builds on the very fast implementation of edit-distance matching provided by the VLSI text search processors. The text-search module supports simultaneous use of approximate search (typing errors), associative search (evaluating related search terms by using a fuzzy thesaurus [26, 27]), wildcard search, regular sequences of word specifications, and various boolean and fuzzy connectives. Argument tags can be placed in order to build templates for *information extraction*. A flexible treatment of

word specifications	
?	any number of characters within a token
$?n$, $?(n)$	at most n arbitrary characters within a token
#	exactly one arbitrary charcter
$\sim$	switches to approximate search (default)
!	switches to exact search
$-$	flexible token boundary (used to separate compound words)
complex word specifications	
$w\ \mu$	weighting of word specification w by weight $\mu \in [0,\ 1]$
$\{\ w_1,\ \ldots,\ w_n\ \}$	simultaneous search for associated search terms $w_1, \ldots, w_n$ (possibly weighted)
operations on regular sequences in word specifications	
x_1, x_2	concatenation of word specifications
x*	an arbitrary number of consecutive occurrences of a word specification
x*n	at most n consecutive occurrences of a word specification
x+	at least one occurrence
x+n	at least one and at most n consecutive occurrences
$<b>$	specifies separating characters between tokens in search sequences. b is a string of:
	`b` (Blank) whitespace characters required between tokens (blank, tab,...)
	`p` (Punctuation) permit occurrence of punctuation characters between tokens
	`h` (Hyphen) the hyphen character "-" is permitted to separate tokens; special treatment because it also discerns positive/negative numbers
	`g` (Graphical Characters) graphical characters are permitted between tokens
	Default: <`ghp`>
(fuzzy-) logical connectives and adjacency operators	
`and`	fuzzy conjunction, realised by `min`
`or`	fuzzy disjunction, realised by `max`
`not`	fuzzy negation, realised by $1 - x$
`atleast` n	n-th best result
`average`	weighted arithmetic mean of all search term relevances
`W`, `W`n	distance of two search terms $\leq n$ tokens, respecting order (without n: system-defined distance)
`N`, `N`n	distance of two search terms $\leq n$ tokens, ignoring order (without n: system-defined distance)
information extraction	
`extract`	keyword, activates information extraction
\$( )	marks search sequence as argument of information extraction (bindings = matches of sequence in a text will be extracted)

Table 1: Query Interface of the Text-Search Component

word boundaries ensures an adequate handling of composite words/phrases and word separation. The external interface of the text search component is shown in table 1. It should be noted, however, that the interface is completely shielded from the user by means of the NL interface. It only serves to implement NL concepts in domain-specific search expressions.

- *image processing primitives* (partly realized in VLSI hardware), e.g. two-dimensional convolution, edge detection, median filter, histogramming, transformations between image types and colour spaces, image segments, scaling, resampling,...;

- *discrete and parametrized fuzzy sets* with corresponding fuzzy connectives ($\min$, $\max$, $1-x$, arithmetic sum, arithmetic product, α-cut, level-operation,...);

- *aggregation methods* which combine the results computed for a set of objects, a time series, or an image region (e.g. $\mathrm{average}$, median,...);
- *interpolation methods* which estimate a continous-time parameter curve from a discrete time-series;
- *information fusion methods* [28, 29] which combine related information taken from different sources (e.g. weighted average, fuzzy integral, and other fuzzy information combination operators [30]). Such methods prove useful for example, to combine cloud classifications based on information from different spectral channels;
- *quantificational operations* in order to evaluate quantifying expressions in NL queries ($\mathrm{almost\ everywhere}$, never,...);
- *operations on geographic images:* cartographic projections, transformation between projections.

These generic FRR methods are modularly implemented and distributed over currently 42 program modules written in C with about 400 declared functions (16000 lines of C-code total). For performance reasons, the methods are executed in lower layers of the HPQS system, namely the parallel media server and (whenever possible) on the VLSI hardware. In order to enable external access from the mediator, suitable C++ classes are derived from CORBA communication classes of the mediator, which provide transparent access to all functions registered in the parallel server. After importing these FRR-classes into the database scheme of the mediator, FRR level functionality can be directly accessed using OQL queries. Implementation of new domain-specific concepts and adaption to a given application domain is greatly facilitated by these primitives and the object-oriented class structure.

Due to our current query domain, there is an emphasis on search for image regions, time series, and processing of geographical images with associated cartographic projections. Our main concern, however, are fuzzy information combination operators which are a prerequisite of

- the evaluation of complex NL queries involving expressions of quantification, comparison, or other types of aggregation;
- the processing of NL queries which refer to an underlying *domain model*. In the meteorological domain, for example, users are typically interested in certain weather conditions in a specified local and temporal region. Weather conditions, however, are not fully specified by any single document in the database; for example, satellite images might provide the required data on cloudiness, while ground temperature readings might be obtained from temperature maps. In addition, more than one document might describe the same aspect of a weather situation. Therefore, operators for information combination are required which establish content-related links between the available documents and allow for a fusion of (possibly conflicting) information extracted from several document sources.

4.4 Fuzzy quantifiers

4.4.1 Quantified expressions in NL queries

In traditional information retrieval systems, the set of operators for the aggregation of search results is essentially restricted to the Boolean connectives **and**, **or** and **not**.[2] These connectives may, of course, also occur in NL queries, and they form an integral part of the meaning of these queries. For example, when computing search results for an NL query containing a condition "not cloudy", the connective **not** must not be ignored.

The "modes of combination" of natural language, however, i.e. the various ways in which concepts might be interrelated, are by no means restricted to these connectives. In particular, the meaning of NL queries depends heavily on the *quantifying expressions* involved, as e.g. witnessed by the different meaning of "there are few clouds over Italy" vs. "there is a lot of clouds over Italy", which both could be part of queries addressed to our system. Therefore, the meaning of quantifying expressions must be accounted for if content-based retrieval, which reflects the semantics of NL queries, is to be achieved. Let us note that many of these expressions are *approximate* or *fuzzy* in nature:

- **often**, **rarely**, **recently**, **mostly**, **almost always** ,... (temporal)
- **almost everywhere**, **hardly anywhere**, **partly** ,... (local)
- **many**, **few**, **a few**, **almost all**, **about ten**, **about 40 percent**,... (approximate specification of the cardinality of a set, or a proportion of cardinalities).

These expressions are best modelled as resulting in gradual evaluations.

Another class of quantificational expressions is not fuzzy itself, but an application of these quantifiers to fuzzy concepts presupposes a reasonable and systematic generalisation of their semantics to the case of fuzzy arguments. For example, **everywhere**, **nowhere**, **always**, **ten times**, **at least ten**, **all**, **less than 20**,... can be adequately modelled in the framework of classical (two-valued) logic. The query, however,

Is the weather fine in all of upper Bavaria?

requires the two-valued quantifier **all** to be applied to the fuzzy regions **upper Bavaria**, **fine weather**. In order to do so, the semantics of **all** must be extended to fuzzy arguments.

4.4.2 Methodology and proposed solution

Our solution [31] to this problem of systematic interpretation of fuzzy quantifying expressions is based on the Theory of Generalized Quantifiers (TGQ) [32, 33, 34, 35], certainly the most elaborate extensional theory of natural language quantification. In TGQ, quantifying expressions are generally called "determiners".

An n-ary determiner on a set of "entities" E is a function $D : \mathcal{P}(E)^n \longrightarrow 2$, where $2 = \{0, 1\}$ is the set of truth values and $\mathcal{P}(E)$ is the set of subsets of E.

[2] Adjacency (**near**) is not an aggregation operator because it applies to search terms instead of logical evaluations.

A determiner D thus assignes to each n-tuple $(X_1, \dots, X_n)$ of subsets of E a corresponding truth value $D(X_1, \dots, X_n) \in 2$. A well-known example is

$$\mathbf{all}(X_1, X_2) = 1 \quad \Leftrightarrow \quad X_1 \subseteq X_2 .$$

E might be any set of objects; in our protoype system, a set of time points, a set of pixel coordinates, a set of image regions, or a set of text search expressions.

Let us now turn to the fuzzy case. A *fuzzy subset* X of a set E assigns to each $x \in E$ a gradual membership value $\mu_X(x) \in I$, where $I = [0, 1]$. The set of fuzzy subsets of E, denoted by $\widetilde{\mathcal{P}}(E)$, is therefore in one-to-one correspondence with the set of membership functions $\mu : E \longrightarrow I$, i.e. $\widetilde{\mathcal{P}}(E) \cong I^E$.

By an n-ary fuzzy determiner on a set E we denote a function $D : \widetilde{\mathcal{P}}(E)^n \longrightarrow I$.

A fuzzy determiner thus maps any n-tuple $(X_1, \dots, X_n)$ of fuzzy subsets of E to a corresponding fuzzy membership value $D(X_1, \dots, X_n) \in I$. This definition is the obvious generalisation of two-valued determiners (in the sense of TGQ) to the fuzzy case. Nevertheless, these fuzzy determiners pose a problem. We all have strong intuitions about the proper definition of determiners on “crisp” subsets of E; for example, the above definition of $\mathbf{all} : \mathcal{P}(E) \times \mathcal{P}(E) \longrightarrow 2$ stated above is certainly uncontroversial. In the fuzzy case, however, such intuitions are lacking: which choice of $\mathbf{all} : \widetilde{\mathcal{P}}(E) \times \widetilde{\mathcal{P}}(E) \longrightarrow I$ should be considered the “correct” one and why?

Our solution to this problem is based on the following idea. Suppose that in order to define a fuzzy determiner $D : \widetilde{\mathcal{P}}(E)^n \longrightarrow I$, we would only have to specify its behaviour on “crisp” arguments, i.e. on $\mathcal{P}(E)^n$, because we had a consistent mechanism $\mathcal{F}$ which generalises our partial definition to a fuzzy determiner $\mathcal{F}(D)$ which is defined on fuzzy arguments also. For example, we could start with $\mathbf{all} : \mathcal{P}(E) \times \mathcal{P}(E) \longrightarrow 2 \subseteq I$ and automatically obtain its fuzzy analogon $\mathcal{F}(\mathbf{all}) : \widetilde{\mathcal{P}}(E) \times \widetilde{\mathcal{P}}(E) \longrightarrow I$. Then the above problem of the proper definition of $\mathbf{all}$ as a fuzzy determiner would have a unique answer *relative to* $\mathcal{F}$, namely $\mathcal{F}(\mathbf{all})$, and conditions on *reasonable* or *intended* interpretations of fuzzy quantifiers could be stated as axioms governing the behaviour of $\mathcal{F}$. To make this idea work, we firstly introduce fuzzy pre-determiners, and then describe the generalisation mechanism.

An n-ary fuzzy pre-determiner on a set of entities E is a function $D : \mathcal{P}(E)^n \longrightarrow I$.

Hence in order to define a fuzzy pre-determiner, only results for the application of D to “crisp” arguments (n-tuples of two-valued sets) must be specified. Therefore, an appropriate interpretation of NL determiners is much easier to obtain than in the case of fuzzy determiners. In particular, every two-valued determiner (i.e. determiner in the sense of TGQ, such as $\mathbf{all}$), is a fuzzy pre-determiner: we need not guess any new definitions.

A systematic generalisation of these fuzzy pre-determiners to corresponding fuzzy determiners is accomplished by means of a determiner fuzzification scheme.

A determiner fuzzification scheme (DFS) $\mathcal{F}$ assigns to each n-ary fuzzy pre-determiner $D : \mathcal{P}(E)^n \longrightarrow I$ an n-ary fuzzy determiner $\mathcal{F}(D) : \widetilde{\mathcal{P}}(E)^n \longrightarrow I$ subject to a number of axioms ensuring its reasonable behaviour (see table 2)

Axioms for determiner fuzzification schemes	
• $\mathcal{F}(D)\vert_{\mathcal{P}(E)^n} = D$	• $\mathcal{F}(\neg) = 1 - x$
• $\mathcal{F}(\wedge)$ is a t-norm	• $\mathcal{F}(\neg D) = \neg\mathcal{F}(D)$
• $D \le D' \Rightarrow \mathcal{F}(D) \le \mathcal{F}(D')$	• $\mathcal{F}(D \wedge D') \le \mathcal{F}(D) \wedge \mathcal{F}(D')$
• compatibility with argument transpositions	

Table 2: DFS Axioms (Selection)

The scheme actually used in the HPQS prototype is defined by

$$\begin{aligned}
\mathcal{M}(D)(X_1, \ldots, X_n) &= \int_0^1 D_\gamma(X_1, \ldots, X_n)\,\mathrm{d}\gamma \\
D_\gamma(X_1, \ldots, X_n) &= \mathrm{m}_{\frac{1}{2}}\{D(Y_1, \ldots, Y_n) : Y_1 \in \mathcal{Y}_1^\gamma, \ldots, Y_n \in \mathcal{Y}_n^\gamma\} \\
\mathcal{Y}_i^\gamma &= \{Y \subseteq E : (X_i)_\gamma^{\min} \subseteq Y \subseteq (X_i)_\gamma^{\max}\} \\
(X_i)_\gamma^{\min} &= \begin{cases} (X_i)_{\ge \frac{1}{2}+\frac{1}{2}\gamma} & : \gamma \in (0, 1] \\ (X_i)_{> \frac{1}{2}} & : \gamma = 0 \end{cases} \\
(X_i)_\gamma^{\max} &= \begin{cases} (X_i)_{> \frac{1}{2}-\frac{1}{2}\gamma} & : \gamma \in (0, 1] \\ (X_i)_{\ge \frac{1}{2}} & : \gamma = 0 \end{cases}
\end{aligned}$$

where $\mathrm{m}_{\frac{1}{2}}$ is fuzzy median [36, 30], $(X)_{\ge\alpha} = \{x : \mu_X(x) \ge \alpha\}$ is α-cut and $(X)_{>\alpha} = \{x : \mu_X(x) > \alpha\}$ is strict α-cut. It can be shown that the integral is well-defined, regardless of D.

$\mathcal{M}$ has some nice theoretical properties; in particular, all of the above axioms (and more) are fulfilled.[3] For the well-known quantifiers $\forall$, $\mathbf{all}$ and $\mathbf{atleast\ n}$, the following results are obtained:

$$\begin{aligned}
\mathcal{M}(\forall)(X) &= \min_{x \in E} \mu_X(x) \\
\mathcal{M}(\mathbf{all})(X_1, X_2) &= \min_{x \in E} \max(1 - \mu_{X_1}(x), \mu_{X_2}(x)) \\
\mathcal{M}(\mathbf{atleast\ n})(X) &= \mu_{(n)},
\end{aligned}$$

where $\mu_{(n)}$ is the n-th largest element in the ordered sequence of membership values of X (including duplicates). In the general case, $\mathcal{M}(D)$ cannot be described by such closed-form expressions. However, $\mathcal{M}(D)$ always permits an efficient operationalisation on the basis of histogram computations if D is quantitative.[4] In the HPQS system, we have implemented 28 fuzzy determiners in this fashion.

4.5 Domain methods

A prototypical implementation in the meteorological domain serves to validate the fuzzy search mechanism. These sample evaluation methods comprise:

- cartographic projections of the images classes under consideration;

[3]For details on DFS axioms, the above definition of $\mathcal{M}$, and proofs, we invite the reader to consult [31].

[4]i.e. if D depends only on the cardinalities of boolean combinations of its arguments.

- access to temperature readings in false-colour ground-temperature maps;
- objective (“more than $20°$ Celsius”) and subjective (“warm”) classification of temperatures;
- estimation of cloud-top height and cloud density in satellite images;
- estimation of degrees of cloudiness.

Combined with the generic FRR methods described above, temperatures and degrees of cloudiness can be interpolated; average, minimal and maximal values can be determined; evaluations can be aggregated across time and space using fuzzy quantifiers etc.—so a basic stock of methods for the first HPQS prototype is already available. As the user queries are matched against a weather model which combines information extracted from a related documents, selection or rejection of a document under a given query may depend on information provided by other documents. By considering these content-related links between associated documents, a much finer-grained and purposive search can be achieved compared to standard methods.

4.6 Example of semantic search in HPQS

In order to show how a semantic search based on interrelated documents is actually performed in HPQS, we consider the following sample query:

Show me pictures of cloud formation over Bavaria taken in August 1997!

This query requests weather images C from the database with an associated date $C.\mathrm{date}$ in August 1997 subject to the condition $R(C)$

C *witnesses a weather situation of cloud formation over Bavaria.*

So while the user simply requests *documents* of the database (as opposed to question-answering in question-answer systems and expert systems), the condition R imposed on these documents refers to an underlying *domain model* (in this case, of meteorology) in which the concept of “cloud formation” over some specified region is given a procedural interpretation.

Considering a single image C is not sufficient to compute $R(C)$ because “formation” is essentially a matter of change and thus can only be judged relative to other images (in this case, images of relevance to the weather situation *before* $C.\mathrm{date}$). Therefore, at least the weather image immediately preceeding C must be taken into account.

The task of detecting the relevant documents is decomposed by the retrieval component as follows: “cloud formation” in a given local region is interpreted as a strong increase in cloudiness in that region in two consecutive weather maps. The systems thus scans through the sequence of images in the specified time interval (August 1997). For each image C under consideration, it firstly determines the predecessor image P. The following operations are then applied to C and P :

a) compute estimates C_1 , P_1 of the density of low clouds in C and P;

b) transform the results into the cartographic projection of the region B under consideration – in this case, $B = \mathbf{Bavaria}$, yielding C_2, P_2;

c) compute the fuzzy evaluations (grey-value images) $C_3 = \mathbf{cloudy}(C_2)$, $P_3 = \neg\mathbf{cloudy}(P_2)$

d) combine C_3 and P_3 *relative to the given region* B by a suitable quantifier to form the gradual result R:

$$R \quad = \quad \min\{\,\mathcal{M}(\geq 70\,\%)(B\,,C_3)\,,\,\mathcal{M}(\geq 70\,\%)(B\,,P_3)\,\}\ .$$

Hence R is the fuzzy conjunction of the conditions that at least $70\,\%$ of Bavaria are sunny in the predecessor image P and that more than $70\,\%$ of Bavaria are cloudy in the current image C.[5] Let us note that in this case, the universe of quantification is the set of pixel coordinates $E = \{0, \,,\ldots\,, W-1\} \times \{0\,,\ldots\,,H-1\}$, where W is the width and H the height of the images B, C_3 and P_3 under consideration. Hence when applying quantifiers to these grey-value images, we think of these as fuzzy subsets of E which to each pair of pixel coordinates $(x\,,y) \in E$ assign a gradual membership value in I.

Intermediate results of the search process are shown in fig. 3. As indicated by the search results in fig. 4, the system detects images in which there is a clearly-visible increase in cloudiness.

[5]The choice of these percentages is of course to some degree arbitrary. However it should be noted that each reasonable choice yields an improvement compared to an approach which does not utilize such modelling tools.

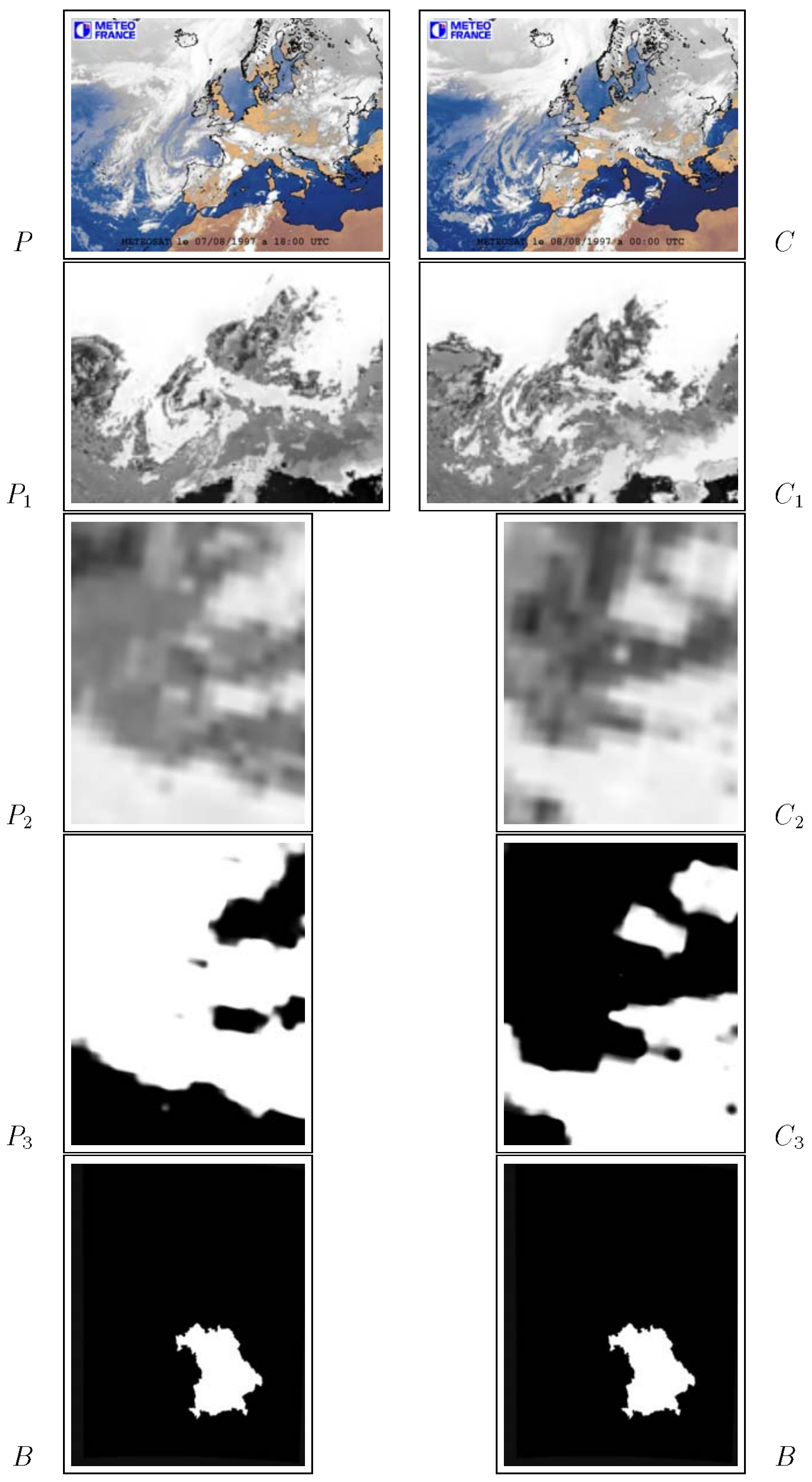
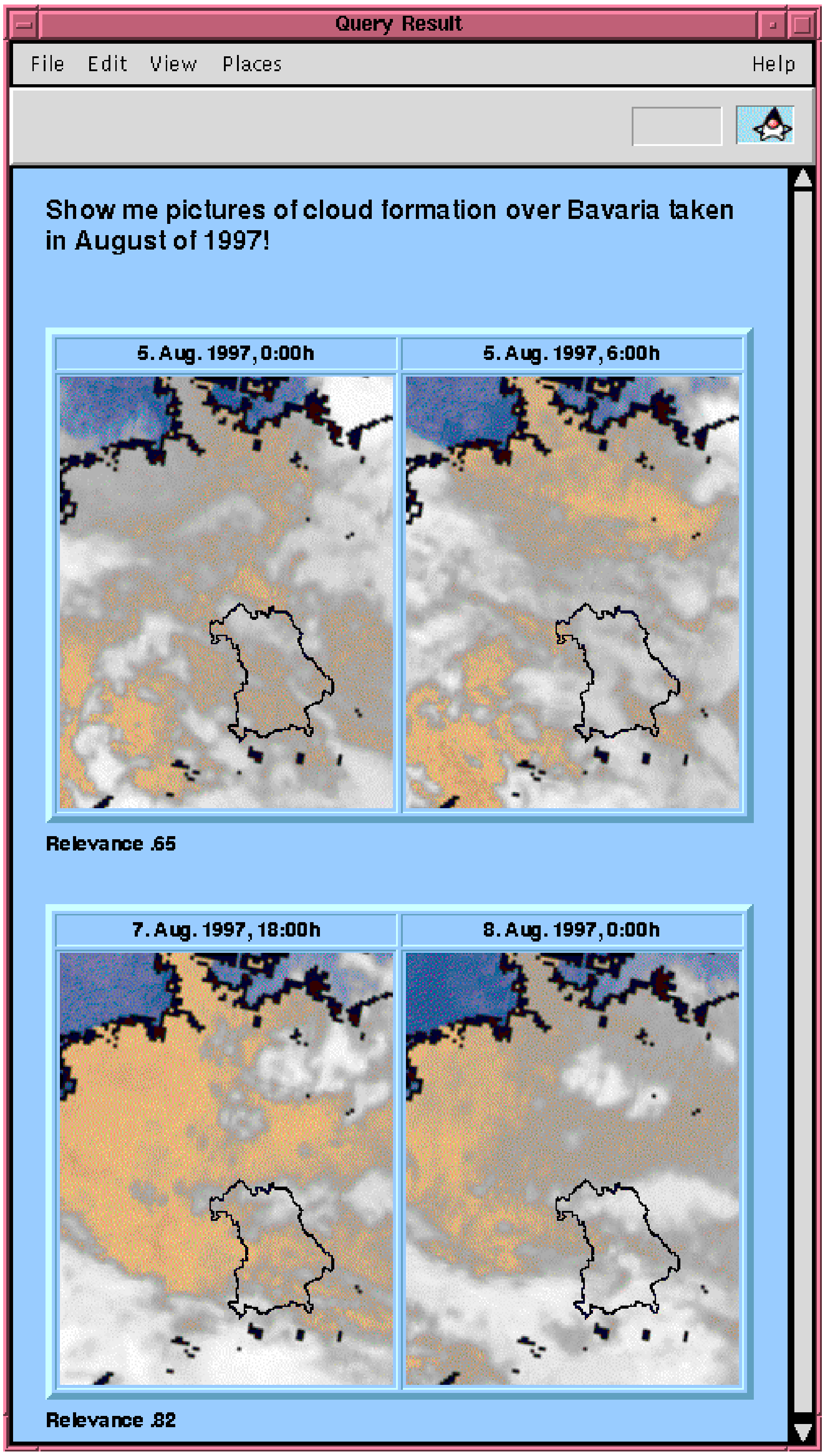

METEO FRANCE
METEOSAT le 07/08/1997 a 18:00 UTC
METEO FRANCE
METEOSAT le 08/08/1997 a 00:00 UTC
P
C
P_1
C_1
P_2
C_2
P_3
C_3
B
B

Figure 3: Intermediate search results



Show me pictures of cloud formation over Bavaria taken in August of 1997!

Relevance .65

Relevance .82

Figure 4: Detected documents (relevant segments)

5 The object-oriented multimedia query mediator

5.1 A well-structured view of the multimedia system

To keep the interface between the media server and the search module manageable, querying instances must be shielded from all the intricacies which have to be dealt with in the presence of heterogeneous media, application-specific document classes, and distribution of document sources. These include:

- determining physical document location;
- communication protocols required to access a document;
- choice of an appropriate method which instantiates a search term (searchable concept) for a document under consideration;
- enforcement of security constraints based on the security mechanisms of the external sources.

The HPQS multimedia mediator [37, 38] serves as an integration tool which establishes a well-structured view of that part of the underlying multimedia system (the parallel media server), which is relevant to the retrieval module. On the side of the querying instance, the mediator has the appearance of an ODMG-93 database [39] augmented by additional features, e.g. concerning security constraints on document access. The well-structured view of the mediator greatly facilitates an implementation of the mapping from ISR query representations as constructed by NL analysis to executable OQL database queries.

The mediator is built on top of an existing object database management system (ODBMS), namely O_2 , which is utilized for storage and retrieval of information about the multimedia objects in an underlying database. The mediator's application-oriented view is defined by appropriate multimedia structuring schemas which can be likened to that of DB schemas in database theory. Indeed, multimedia schemas are transformed by the mediator into database schemas of the underlying ODMG database. By means of this transformation, object-orientation and other benefits of the ODL (object description language) are inherited from the underlying ODBMS, in particular the distinction of an internal, conceptual and application layer which ensures physical and logical data independence.

5.2 Proxy objects

In the HPQS system, all documents accessible through the mediator are stored externally (comparable to the situation in WWW search). In the database, these documents are represented by proxy objects. Each proxy is used to hold information about its associated document; in particular, the physical location of a document and its type can be determined from its proxy. The mediator maintains a full proxy state of the documents stored in the parallel media server. In our system, the only way of changing the mass data is by using the mediator's maintenance routines (insertion, deletion…). Therefore, it is relatively easy to keep the proxy state consistent and up-to-date.

5.3 Query evaluation

In the first implementation stage, the so-called “one-pass approach” of query evaluation has been realized. In this approach, the OQL query is processed by the built-in routines of the ODBMS unless operations invoking external communication are requested. In the case of such an operation, the corresponding request is sent to the parallel server and processing is suspended until an external result is returned. Unfortunately, this approach does not make full use of the parallelity offered by the media server. Therefore, a two-pass approach will be implemented which is characterised by syntactic analysis of the OQL query, bundling of requests, and asynchronous fetching of results. Query optimisation methods are used in order to ensure efficient utilisation of the media server.

5.4 Materialisation

In HPQS, evaluation of queries is computationally intensive because model-based search anchored in direct search methods is supported. For that reason, it often pays off to materialize attribute values and results of applying operations by making them persistent: if a materialized attribute value or operation result is queried again, it can be looked up in the database and need not be recomputed. This caching mechanism plays a similar role in HPQS as that of pre-computed indexes in more traditional retrieval systems.

6 The parallel media server

Another prerequisite of the HPQS is that of a fast storage and retrieval medium, which provides a scalable number of documents to a scalable number of external clients in accordance with specified quality-of-service requirements. This system is currently being developed based on a parallel HTTP server [40, 41, 42]. The parallel server is responsible for

- *storage, maintenance, and delivery of mass data*, in particular of high-resolution digitized images. The documents – as well as search methods associated to a certain class of documents – are identified as objects and automatically stored into the appropriate subsystems. Due to the close coupling between the database mediator (dealing with object IDs and proxy objects) and the corresponding mass data as stored by the media server, all changes in the mass data (insertion, deletion etc.) are initiated by the database mediator.
- *execution of search and interpretation methods* operating on the mass data, as requested by the object-oriented database mediator. Due to this division of work, the object-oriented database is relieved from computation-intensive methods of the considered application.
- *document pre- and postprocessing*. Several image- or text processing tasks requiring immense computational power are routed to the VLSI hardware (see section 7.1). The obligatory preprocessing of the data and the postprocessing of results have to be performed by the compute server.

6.1 Computational platform

A parallel computer system (Parsytec CC) with distributed memory serves as the underlying computational platform. The system currently in use comprises 16 processor nodes. Local hard-disks

(4 Gbyte per node) are used to store mass data and temporary evaluation results. The processors are linked by a fast communication network based on High-Speed Serial Links (IEEE 1355). There is no global memory in the system. Queries to the parallel server are triggered by requests of external clients. In order to enable client-server communication, some of the processor nodes are equipped with appropriate interfaces.[6]

6.2 Operational model

The operational model of the parallel query and document server may be sketched as follows:

- External clients are initiating requests to the parallel server. Possible types of requests are “get” (read a named object), “put” (insert a named object, i.e. a piece of mass data), “del” (delete a named object), and “execute” (apply a method on the data according to the specified parameters).
- The request is routed to a processor node in the parallel server. Selection of the processor depends on the methods and documents involved. As a large number of jobs can be handled simultaneously, clients are allowed to tag their request with priorities in order to partly control evaluation order. However, actual planning and execution are determined by a dynamic mapping algorithm inside the parallel server, because it is only there that the current workload is known in detail.
- After mapping onto a node of the parallel computer, the request is executed on that node.
- At the end of a transaction, results are sent back to the client.

6.3 Main components of the parallel server

The parallel server comprises the following main software modules:

- *communication interfaces*. Communication of clients with the media server is based on the CORBA object broker architecture. In addition, some of the processor nodes need to access the dedicated VLSI search hardware. Communication of these nodes with the search processor card is accomplished according to the PCI-bus specification.
- *directory service element (DSE).* The DSE maintains a directory of the whole server's content, which provides the mapping from logical names (object names as used by the database) to corresponding physical addresses within the parallel media server. The DSE supports data caching and redundant copies of documents in the local storage subsystems.
- *scheduler.* The routing process is controlled by a scheduler. The choice of a processor node a given request should be mapped to is based on criteria of processor and harddisk workload, locality of data required to process the query, requirements on compute-time, communication bandwith and others. Migration of requests is not supported, i.e. scheduling is assumed to be non-preemptive.

[6]In the next project stage (1998+), a high-performance SCI cluster of more than 64 processors (PC systems or workstations) will be used, which provides a solid basis for investigations on scalability and efficiency issues.

- *load balancer.* Data and methods are allowed to migrate in the storage subsystem. Based on information accumulated by the monitor module, the load balancing process attempts to achieve optimal utilisation of the server by re-organizing physical storage locations of mass data within the server. Objects migrate to processors with low workload by moving the data and changing the mapping from object name to physical addresses accordingly.
- *distributed monitor.* The distributed monitor keeps track of the behavior of the parallel query server and makes available information on access times, processing times, etc. to other components of the server.

7 Dedicated VLSI Search Hardware

7.1 The case for search hardware

One of the key objectives in the development of the HPQS is the realization of a search facility for regions in images with certain domain-specific properties. As the methods which implement these properties may be parametrized, pre-computation of results for all methods and choices of parameters is not feasible both in terms of computational effort and storage requirements. Consequently, it is not possible to base the search procedure on indexing only; instead, part of the search must be performed directly on the document data by using domain-specific interpretation rules. Efficient processing of these evaluation methods is crucial to the performance of the query server. However, the direct search stage is extremely compute-intensive. Up to now, search for regions in images or figures could not or only partly be realized even on highly-parallel DSP or RISC platforms because the required computational power may well reach the Tera-Ops range. Only dedicated, full-custom VLSI implementations are possible.

7.2 VLSI circuits for text and image search

There is a broad range of methods used in computer vision (CV), especially on the higher processing levels of image interpretation. Due to the diversity of the methods required, core functionality had to be identified which is frequently used in applications and so compute-intensive as to make a dedicated VLSI-implementation pay off. The final selection of methods for VLSI implementation comprises two-dimensional convolution, weighted median filtering, histogramming and morphological operations.
The implementation of *two-dimensional convolution* may profit from the fact that in image processing, convolution kernels are generally vertically symmetric or vertically antisymmetric. By multiple usage of its functional units, the image processor [43] is able to support convolution kernels of varying size. In this way, computational resources are used economically, i.e. there is a dynamic trade-off between processing speed and kernel size. A similar design is used in the implementation of *weighted median filtering*, where the size of the mask can again be chosen according to the requirements of the application. Computation of the weighted median is based on even/odd-transpositions (“bubblesort”).

In addition to the image processor, a processor for approximate and associative text search is being built [44] which implements the computationally intensive edit-distance matching procedure of the text search module (see section 4.3).

The image- and text search processors are implemented using a full-custom design [45, 46] in order to fulfill the extreme demands on processing speed. Prototypes are manifactured in moderately-fine CMOS technology within the EUROPRACTISE-programme.

7.3 Integration into the HPQS

The search processors are integrated into the parallel media server by means of a PCI card. Due to the very high data rate of the processors, this card must be equipped with highly optimized local memory, which is also under development. The desired bandwith will be achieved by using modern DRAM architectures. Other methods like FFT, block-matching, or motion estimation [47] are available as dedicated VLSI implementations. Such hardware will be added to the processor card in order to further speed up image processing.
Since the dedicated hardware solution should prove useful in a variety of application domains and regardless of the specific text or image data formats, application-dependent or data-format specific aspects have been separated from the core algorithms. Therefore, some preprocessing may be necessary before a hardware search method can be applied to a given document. This preprocessing is done on the parallel media server.

8 Current state of the development

As of this writing, all main modules of the HPQS are operational as prototypes:

- The NL-interface transforms the set of example queries into ISR expressions. The computer lexicon contains ca. 300 wordstem entries, and is continuously expanded.
- The semantic search module provides methods for text search and image evaluation as well as for the handling of geographical (satellite) images. A number of rules for image region classification, such as clouds, have been implemented and tested. Furthermore, the DFS approach to fuzzy quantification described above has been developed and fully implemented.
- The query mediator accepts OQL-queries and forwards them to the media server. Communication is organised according to the CORBA-standard. The query mediator does not make any optimisations; queries are processed strictly sequentially.
- Gateway and request handler[7] of the media server have been fully implemented. Once a set of relevant documents has been detected, these can be obtained from their URLs by means of a parallel http interface. The server can make up to 400 of such connections per second.
- There is a full software emulation of the VLSI-processors available, which also serves as the reference specification for the hardware design.

In the near future these components will be fully integrated into a working prototype that responds to a subset of the query types outlined in section 1.2.

[7] providing access to the processing nodes of the parallel server

9 Conclusions and future research

We have presented a system for structuring and querying large databases of multimedia documents. Our approach attempts to cover the full path from natural language input to the presentation of retrieved documents. Constructing the modules of such a server and integrating them into a working prototype is a complex endeavour, even if the initial functionality is limited. Apart from the obvious refinements, such as larger rule bases and lexica, the following goals will be pursued in the future:

- Cognitively adequate presentation of the documents, i.e. results are presented so as to best match the user's expectations;
- Implementation of continuation queries in which the user refers to a result previously computed;
- Fine grained retrieval precision: extraction of only those bits of information from one or several documents that are relevant to the user. The system output is not a whole document as stored in the database but exactly the information the user requested, i.e. a new document specifically compiled for answering the query.

References

[1] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkhani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9), September 1995.

[2] Virage Inc. http://www.virage.com.

[3] A. Del Bimbo, M. Campanai, and P. Nesi. A three-dimensional iconic environment for image database querying. *IEEE Trans. on Software Eng.*, 19(10):997–1011, October 1993.

[4] A. Del Bimbo and P. Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 19(2):121–132, February 1997.

[5] S. Iyengar and R. Kashyap. Special section on image databases. *IEEE Trans. on Software Eng.*, 14(5):608–688, May 1988.

[6] T. Hermes, C. Klauck, J.Kreyß, and J. Zhang. Image retrieval for information systems. In *Proc. SPIE's Symp. on Electronic Imaging*, San Jose, February 1995.

[7] IBM Corp. http://www.software.ibm.com/data/mediaminer/immn0b15.html.

[8] C. Meghini, F. Rabitti, and C. Thanos. Conceptual modeling of multimedia documents. *IEEE Computer*, pages 23–30, October 1991.

[9] Hermann Helbig. Der MESNET Primer - Die Darstellungsmittel der Mehrschichtigen Erweiterten Semantischen Netze. working paper, FernUniversität Hagen, Hagen, Germany, January 1997.

[10] Hermann Helbig and Marion Schulz. Knowledge representation with MESNET: A multilayered extended semantic network. In *AAAI Spring Symposium on Ontological Engineering*, pages 64–72, 1997.

[11] U. Manber and S. Wu. GLIMPSE: A tool to search through entire file systems. TR 93-34, Department of Computer Science, University of Arizona, Tucson, Arizona, 1993.

[12] Hermann Helbig. Multilayered extended semantic networks as an adequate formalism for meaning representation and cognitive modelling. In *5th International Cognitive Linguistics Conference*, Amsterdam, July 1997.

[13] Marion Schulz and Hermann Helbig. COLEX: Ein Computerlexikon für die automatische Sprachverarbeitung. Informatik-Berichte 210, FernUniversität Hagen, Hagen, Germany, December 1996.

[14] Hermann Helbig and Sven Hartrumpf. Word class functions for syntactic-semantic analysis. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP-97)*, Tzigov Chark, Bulgaria, September 1997. Forthcoming.

[15] Hermann Helbig and Andreas Mertens. Word Agent Based Natural Language Processing. In Loe Boves and Anton Nijholt, editors, *Proceedings of the 8th Twente Workshop on Language Technology – Speech and Language Engineering, Twente, 1 and 2 December 1994*, pages 65–74, Enschede, 1994. Universiteit Twente, Fakulteit Informatica.

[16] Hermann Helbig, Carsten Gnörlich, and Dirk Menke. Realization of a user-friendly access to networked information retrieval systems. In *Proceedings of the AAAI Spring Symposium on Natural Language Processing for the World Wide Web*, pages 62–71, Stanford, CA, 1997.

[17] Hermann Helbig, H. Böttger, F. Ficker, P. String, and F. Zänker. The natural language interface NLI-AIDOS. *Journal of New Generation Computing Systems*, pages 221–246, 1990.

[18] H. Helbig. Syntactic-semantic analysis of natural language by a new word-class controlled functional analysis (wcfa). *Computers and Artificial Intelligence*, 1:53–59, 1986.

[19] M. Eimermacher. *Wortorientiertes Parsen*. PhD thesis, TU Berlin, Berlin, 1988.

[20] N. Bröker, U. Hahn, and S. Schacht. Concurrent lexicalized dependency parsing: the ParseTalk model. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, 1994.

[21] S. Hartrumpf. Partial evaluation for efficient access to inheritance lexicons. In *Proceedings of the 2nd International Conference on Recent Advances in Natural Language Processing (RANLP-97)*, Tzigov Chark, Bulgaria, September 1997.

[22] N. Fuhr. Probabilistic Datalog - a logic for powerful retrieval methods. In E. Fox, P. Ingwersen, and R. Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 282–290, New York, 1995. ACM.

[23] M. Pinkal. On the limits of lexical meaning. In R. Bäuerle, C. Schwarze, and A. Stechow, editors, *Meaning, Use and Interpretation of Language*, pages 400–422, Berlin, New York, 1983. Walter de Gruyter.

[24] M. Rayner. *Abductive Equivalental Translation and its Application to Natural Language Database Interfacing*. PhD thesis, Royal Institute of Technology, Stockholm, September 1993.

[25] M. Rayner and H. Alshawi. Deriving database queries from logical forms by abductive definition expansions. In *Third Conference on Applied Natural Language Processing*, pages 1–8. Association for Computational Linguistics, 1992.

[26] S. Miyamoto. *Fuzzy Sets in Information Retrieval and Cluster Analysis*. Kluwer Academic Publishers, Dordrecht, 1990.

[27] H.L. Larsen and R.R. Yager. The use of fuzzy relational thesauri for classificatory problem solving in information retrieval and expert systems. *IEEE Transactions on System, Man, and Cybernetics*, 23(1):31–41, Jan./Feb. 1993.

[28] A. Knoll, R. Schröder, and A. Wolfram. Fusion of data from fuzzy integral-based active and passive colour stereo vision systems for correspondence identification. In *Proceedings of the VIII European Signal Processing Conference (EUSIPCO-96)*, Trieste, Italy, Sept. 10-13 1996.

[29] A. Knoll and J. Meinkoehn. Data fusion using large multi-agent networks: An analysis of network structure and performance. In *Proc. First IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI-94)*, Las Vegas, 1994. IEEE Press.

[30] I. Bloch. Information combination operators for data fusion: a comparative review with classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1):52–67, 1996.

[31] Ingo Glöckner. DFS – an axiomatic approach to fuzzy quantification. Technical report, Technische Fakultät, Universität Bielefeld, November 1997.

[32] J. Barwise and R. Cooper. Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4:159–219, 1981.

[33] J. van Benthem. Determiners and logic. *Linguistics and Philosophy*, 6, 1983.

[34] J. van Benthem. Questions about quantifiers. *Journal of Symbolic Logic*, 49, 1984.

[35] E. Keenan and J. Stavi. A semantic characterization of natural language determiners. *Linguistics and Philosophy*, 9, 1986.

[36] W. Silvert. Symmetric summation: A class of operations on fuzzy sets. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:657–659, 1979.

[37] J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. A mediator for multimedia systems. In *Proceedings 3rd International Workshop on Multimedia Information Systems*, Como, Italia, September 1997.

[38] J. Biskup, J. Freitag, Y. Karabulut, and B. Sprick. Query evaluation in an object-oriented multimedia mediator. In *Proceedings 4th International Conference on Object-Oriented Information Systems, Brisbane, Asutralia*, Berlin, November 1997. Springer.

[39] R.G.G. Cattell. *The Object Database Standard - ODMG 93*. Morgan Kaufmann, 1996.

[40] P. Berenbrink, R. Lüling, and V. Rottmann. A comparison of data layout schemes for multimedia servers. In *Proc. of the European Conference on Multimedia Applications, Systems and Technologies (ECMAST)*, pages 345–366, 1996.

[41] P. Berenbrink, R. Lüling, and V. Rottmann. A simple distributed scheduling policy for parallel interactive continuous media servers. In *Workshop on Parallel Computing and Multimedia, held on the 11th Int. Parallel Processing Symposium (IPPS)*, 1997. to be published in the journal “Parallel Computing”.

[42] R. Lüling. Load balancing problems for scalable interactive media servers. Invited talk at Symposium on Parallel Computing for Solving Large Scale and Irregular Applications (STRATAGEM), 1996. Sophia-Antipolis, France.

[43] C. Henning and H.-M. Blüthgen. Implementierung eines schwach programmierbaren Bildprozessors. Internal report, Lehrstuhl für allgemeine Elektrotechnik und Datenverarbeitungssysteme, RWTH Aachen, July 1997.

[44] H.-M. Blüthgen. Implementierung eines schwach programmierbaren Textprozessors. Internal report, Lehrstuhl für allgemeine Elektrotechnik und Datenverarbeitungssysteme, RWTH Aachen, June 1997.

[45] W. Wilhelm and T.G. Noll. A new mapping technique for automated design of highly efficient multiplexed FIR digital filters. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2252–2255, Hong-Kong, June 1997.

[46] M. Gansen, F. Richter, O. Weiß, and T.G. Noll. A datapath generator for full custom macros of iterative logic arrays. In *Proceedings of the IEEE 1997 International Conference on Application Specific Systems, Architectures, and Processors*, pages 438–447, July 1997.

[47] J.P. Berns and T.G. Noll. A cascadable 200 GOPS motion estimation chip for HDTV applications. In *CICC 96*, San Diego, May 1996.

Bisher erschienene Reports an der Technischen Fakultät
Stand: November 1997

94–01	Modular Properties of Composable Term Rewriting Systems (Enno Ohlebusch)
94–02	Analysis and Applications of the Direct Cascade Architecture (Enno Littmann und Helge Ritter)
94–03	From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction (Robert Giegerich und Stefan Kurtz)
94–04	Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern (Andrè Wolfram und Alois Knoll)
94–05	Searching Correspondences in Colour Stereo Images — Recent Results Using the Fuzzy Integral (Andrè Wolfram und Alois Knoll)
94–06	A Basic Semantics for Computer Arithmetic (Markus Freericks, A. Fauth und Alois Knoll)
94–07	Reverse Restructuring: Another Method of Solving Algebraic Equations (Bernd Bütow und Stephan Thesing)
95–01	PaNaMa User Manual V1.3 (Bernd Bütow und Stephan Thesing)
95–02	Computer Based Training-Software: ein interaktiver Sequenzierkurs (Frank Meier, Garrit Skrock und Robert Giegerich)
95–03	Fundamental Algorithms for a Declarative Pattern Matching System (Stefan Kurtz)
95–04	On the Equivalence of E-Pattern Languages (Enno Ohlebusch und Esko Ukkonen)
96–01	Static and Dynamic Filtering Methods for Approximate String Matching (Robert Giegerich, Frank Hischke, Stefan Kurtz und Enno Ohlebusch)
96–02	Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language (Alois Knoll, Bernd Hildebrandt und Jianwei Zhang)
96–03	Correctness in System Engineering (Peter Ladkin)
96–04	An Algebraic Approach to General Boolean Constraint Problems (Hans-Werner Güsgen und Peter Ladkin)

96–05	Future University Computing Resources (Peter Ladkin)
96–06	Lazy Cache Implements Complete Cache (Peter Ladkin)
96–07	Formal but Lively Buffers in TLA+ (Peter Ladkin)
96–08	The X-31 and A320 Warsaw Crashes: Whodunnit? (Peter Ladkin)
96–09	Reasons and Causes (Peter Ladkin)
96–10	Comments on Confusing Conversation at Cali (Dafydd Gibbon und Peter Ladkin)
96–11	On Needing Models (Peter Ladkin)
96–12	Formalism Helps in Describing Accidents (Peter Ladkin)
96–13	Explaining Failure with Tense Logic (Peter Ladkin)
96–14	Some Dubious Theses in the Tense Logic of Accidents (Peter Ladkin)
96–15	A Note on a Note on a Lemma of Ladkin (Peter Ladkin)
96–16	News and Comment on the AeroPeru B757 Accident (Peter Ladkin)
97–01	Analysing the Cali Accident With a WB-Graph (Peter Ladkin)
97–02	Divide-and-Conquer Multiple Sequence Alignment (Jens Stoye)
97–03	A System for the content-based retrieval of textual and non-textual documents based on natural language queries (Alois Knoll, Ingo Glöckner, Hermann Helbig und Sven Hartrumpf)