RETRIEVAL OF RELATIONAL STRUCTURES FOR IMAGE SEQUENCE ANALYSIS

Wolfgang Benn and Bernd Radig

Fachbereich Informatik, Universität Hamburg
Schlüterstraße 70, D-2000 Hamburg 13

Abstract

The automatic interpretation of time-varying scenes requires symbolic descriptions of the images in a sequence. Symbolic descriptions can be formalized by relational structures, representing properties of symbols and relationships between symbols. Two tasks are essential for the interpretation of image sequences:

- *Grouping of primitive symbols into object descriptions.*
- *Establishing the correspondence relationship between symbols and objects from subsequent images in the scene.*

This paper introduces a dedicated database management system, specialized in storing and retrieving relational structures. It is especially designed to support the retrieval of inexact matching structures which are likely to occure when comparing images. The database runs on a separate processor embedded in a local minicomputer network.

This project is supported by the "Deutsche Forschungsgemeinschaft".

Introduction

The analysis of real-world time-varying imagery generates such an amount of image describing data structures, that their systematic management with the help of a database becomes mandatory. Usually, a symbolic description of each image is resulting from some kind of a segmentation process. Primitive symbols such as regions, boundary segments, points etc. are grouped together into object and scene models. This grouping can be guided by a-priori knowledge, formulated by specifying prototypes of objects and scenes, e.g. following the paradigm of hierarchical synthesis [1] [2]. Moreover, in image sequence analysis a basic relationship between symbols and objects from different images has to be computed, the correspondence relation. Correspondence between and grouping of symbols are prerequisites for image sequence understanding.

The association of symbols with their properties and the inter-symbol relationships (geometrical, hierarchical, correspondence, ...) are formalized as relations. Relational structures are collections of tuples from those relations which describe a distinct entity, e.g. an object model [3]. Instantiation of objects, given a prototype as a relational structure, or the comparison of image descriptions in order to establish correspondence between symbols, can be understood as a matching of relational structures. In real-world scenes, this matching is usually inexact and incomplete.

Image sequence analysis will benefit from a database system if it supports the storage of relational structures (cf. the entity-relationship model [4]) and the retrieval of relational structures which match a given template most closely. We call this kind of retrieval "Query by Structure Example" (QSE).

QSE is a query scheme for an application program which interacts via a message interface with the database system. The database system interacts only with application programs and does not support a dialogue with a human user. Three properties of the database system are essential in this application context.

- *Since scenes and objects may vary from frame to frame, the database system should be able to retrieve relational structures which are completely specified only at runtime.*
- *The database system should be able to retrieve relational structures which are similar to an example structure (which describes e.g. an object prototype) in that respect that symbol properties do not match exactly the specified ones.*
- *Furthermore, even structural dissimilarities should be tolerated; e.g. when an object becomes partially occluded in the scene, a part of its prototype structure is left unmatched.*

If scenes, prototypes, images, objects, etc. are formally described by relational structures, such an inexact and incomplete query can be

Proceedings of the Tenth International Conference on Very Large Data Bases.

Singapore, August, 1984

regarded as the search for a mapping between two relational structures - one presented to the database system as a query structure by the application program at runtime, the other being a candidate from the database. Those mappings are called RS-morphisms and are specified with respect to their tolerance. RS-isomorphisms require structural identity, RS-monomorphisms identify a substructure within a larger structure, and RS-comorphisms map maximal isomorphic parts within both structures.

Relational structures and RS-morphisms

A relational structure (RS) is defined as a tuple $M = (C, \langle R_1, \ldots, R_n \rangle)$ where C contains symbol identifiers and attribute values, and R_i are t_i-ary relations on C. The RS forms a network of R_i-tuples where relationships between symbols are expressed by referencing their identifiers.

The type of each R_i is known in the dictionary of the database and is automatically checked by the database system analysing the type declarations of the current application program. The composition of M from R_i-tuples is comletely left to the freedom of the application program where various RS's will be formed during the progress of the image analysis process.

A RS, e.g. the symbolic description of an object "line" connecting two points in the image plane, may be regarded as a tuple of a relation LINES which is in non-first-normal-form (NF^2, see [5] [6] [7]).

LINE							
TID	START			END			LENGTH
	TID	X	Y	TID	X	Y	
L1	P1	2	6	P2	2	1	5

Here, components of the NF^2-relation LINES are tuples from the relation POINTS which is already in first-normal-form (1NF); the TID is a unique tuple identifier from which the remaining attributes have full functional dependency. The external NF^2-scheme may be transformed into the conceptual 1NF-scheme by decomposition, e.g.:

LINES := π[TID, START.TID, END.TID](**LINES**) and

POINTS := π[START.TID, START.X, START.Y](**LINES**)
$\cup$ π[END.TID, END.X, END.Y](**LINES**).

A query is that process which matches an example structure - provided by the application program - with structures stored in the database by computing appropriate RS-morphisms.

Let us consider a RS $M' = (C', \langle R_1', \ldots, R_n' \rangle)$ homologous to M. The set C - as well as C' - can be decomposed into two subsets CS and CA of symbol identifiers and attribute values, resp. The RS-homomorphism is composed of submappings that are restricted as follows. Tuples of a relation in M can only be mapped to their homologous counterparts in M'. In the same way symbol identifiers in CS are only mapped to corresponding symbol identifiers in CS'.

$\varphi_{R_i} : R_i \rightarrow R_i' \quad i=1,\ldots,n$

$\varphi_{S_k} : CS_k \rightarrow CS_k' \quad k=1,\ldots,m$

For the attribute values occurring in the relation R_i appropriate compatibility functions

$\theta_i : CA \cup CA' \rightarrow [0,1] \quad i=1,\ldots,n$

are defined which decide if a mapping of two tuples is allowed with respect to their attribute value components and a chosen threshold θ_i.

The RS-homomorphism is defined as a triple $\langle \varphi_S, \varphi_R, \theta_\varphi \rangle$. Depending on the properties of the submappings, we distinguish the following types of RS-homomorphisms:

- An RS-monomorphism $\varphi: M \rightarrow M'$ is an RS-homomorphism with one-one submappings.
- An RS-isomorphism $\varphi: M \rightarrow M'$ is an RS-monomorphism with one-to-one submappings. Thus, there is an inverse RS-monomorphism $\varphi^{-1}: M' \rightarrow M$.
- An RS-comorphism $\varphi: M \rightarrow M'$ is an RS-isomorphism between maximal substructures of M and M'.

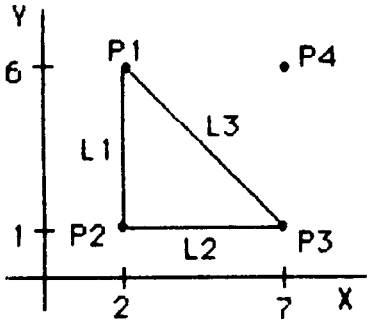
RS-morphisms can be computed by searching cliques in a compatibility graph [8] whose nodes are assigned pairs of tuples r and r' from R_i and R_i', resp., where $\theta_i(r,r') > \theta_i$. Its arcs connect those nodes which are compatible according to the chosen RS-morphism. When a query-RS is transmitted to the database system the compatibility graph is formed and the search for cliques is started. Maximal cliques in this graph are interpreted as the best matches between the query-RS and the RS's in the database. Each matched RS is returned to the application program.

Query by Structure Example

QSE follows in some analogy the lines of QPE [9] and QBE [10] with the important exception that the database system does not prompt a user with a template of one or more relations. In QSE the application program supplies such a template as

a data structure to the database system, and specifies the kind of transaction to be performed together with the kind of RS-morphism if the transaction includes a search.

As an example, a description of an image, stored in the database, consists of a triangle (P1,L1,P2,L2,P3,L3) and a point P4. The application program may be interested to check, if there is a line from point P2 to the point with coordinates x=7 and y=1. It fills this information in an empty tuple and requests an RS-isomorphism. The database system returns the desired information about the line l2.



LINE							
TID	START			END			LENGTH
	TID	X	Y	TID	X	Y	
	P2				*7*	*1*	
L2	P2	2	1	P3	7	1	5

Even if an endpoint of a line is not exactly known as in the request for a line between x=8, y=2 and P1, the returned information will indicate a line between a near point P3 and P1.

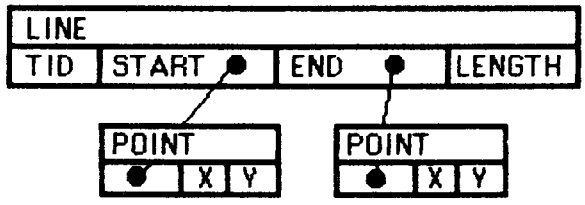
LINE							
TID	START			END			LENGTH
	TID	X	Y	TID	X	Y	
		8	*2*	*P1*			
L3	P3	7	1	P1	2	6	5

This is possible because the RS-isomorphism can be adjusted to tolerate differences in attribute values. In order to tolerate even missing structures as in the request to find a line between P4 and P2, an RS-comorphism has to be specified for the match. Only the information about P2 and P4 is then returned (an RS-isomorphism would have returned an empty answer).

LINE							
TID	START			END			LENGTH
	TID	X	Y	TID	X	Y	
	P4			*P2*			
	P4	7	6	P2	2	1	

Conceptual Scheme

An application program should be able to present any structure as a template to the database system, e.g. when comparing two a priori unknown image structures of a sequence. This flexibility is achieved by allowing the programmer to compose the structure from 1NF-relations (which with respect to the database system can be handled by conventional methods [11]). The link between the 1NF-tuples is done in the application program be references between a TID-subfield of one tuple to the appropriate variable which holds the referenced tuple.



The references are expressed as access type subfields in Ada, the current application language in our image processing network.

```
TYPE point IS RECORD
  tid : systemkey;
  x,y : coordinates;
END RECORD;

TYPE line IS RECORD
  tid   : systemkey;
  start,end : ptr;
END RECORD;

TYPE ptr IS ACCESS point;
```

Identical values of reference variables are bound to identical TID's within the database system. On return, the database system inserts into the structure within the application program for referencing subfields of a tuple reference values, and for referenced tuples their TID value.

To operate on the query-RS formed as a network of referencing and referenced tuples, the database system accesses the type declarations of the involved relations [12]. Our Ada compiler transforms the application program into an intermediate language, called DIANA [13]. A IANA representation is an attributed

network of descriptive nodes which stores the complete information about the source program. From this network, the relevant type declarations are copied into the data dictionary of the database system, if new relations are to be inserted into the database. Otherwise, only the user's declarations are compared with the data dictionary to ensure conformity.

A database transaction is described by a tagged record which is sent via the communication channel from an application program to the database processor. One field of the record is a variant component which contains a reference to the RS involved, e.g. to a record of the type "line". From the value of the tag-field, the database system is able - with the help of the data dictionary - to recognize the relation "line". From the stored declaration of the type "line" the second and third record components are identified as pointers to records of the type "point". In this way, the database system is able to visit all tuples of the given RS which can be reached via the supplied reference pointers. The database system can access all information in the RS which an application program has supplied, and can store the result of a retrieval back into the same structure. This structure may be used in a cursor like fashion to obtain subsequent matches.

Concluding Remarks

A prototype of the database system was completed last year. It was written in PASCAL and served to study the disk management and organisation, basic functions of the database system, and the communication interface between the database processor and the application programs in the network. This year, a first version is in progress which is written in Ada and will include the automatic maintainance of the data dictionary by inspecting the DIANA structure of the application programs.

The transfer of the search for objects and correspondence in symbolic image descriptions from the application program to the database processor will remove a remarkable load from the application processors. A speedup of the total execution time is to be expected due to the parallel execution of databse actions and application programs. This is especially important when experimenting with image sequences where a vast amount of data structures is generated, accessed and manipulated (see e.g. [14]).

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

Proceedings of the Tenth International Conference on Very Large Data Bases.

Literature

[1] H.G. Barrow, R.J. Poppelstone, Relational Descriptions in Picture Processing, Machine Intelligence 6, B.Meltzer, D.Michie, (Eds.), University Press Edinburgh 1971, pp. 377 - 396

[2] H.G. Barrow, A.P. Ambler, R.M. Burstall, Some Techniques for Recognising Structures in Pictures, in Frontiers of Pattern Recognition, S. Watanabe (Ed.), Academic Press New York 1972, pp. 1 - 29

[3] R. Radig, Hierarchical Symbolic Description and Matching of Time Varying Images, Proc. 6th Int. Conf. on Pattern Recognition, München, Oct. 1982, pp. 1007 - 1010

[4] P.P. Chen, The Entity Relationship Model: Toward Unified View of Data, ACM-TODS, Vol.1, 1976, pp. 9 - 36

[5] H.J. Scheck, P. Pistor, Data Structures for an Integrated Data Base Management and Information Retrieval System, Proc. VLDB 1982, Mexico City, Sept. 1982, pp. 197-207

[6] H.J.Scheck, M. Scholl, Die NF^2-Relationenalgebra zur einheitlichen Manipulation externer, konzeptioneller und interner Datenstrukturen, in Sprachen für Datenbanken, J.W. Schmidt (Ed.), Fachgespräch 13th GI-Jahrestagung, Hamburg, Oct. 1983, IFB 72, Springer-Verlag Heidelberg, pp. 113-133

[7] W. Benn, B. Radig, Symbolische Bildbeschreibungen mit nichtnormalisierten Relationen, 6th DAGM Symposium, Graz, Sept. 1984, in press

[8] B. Radig, Image Sequence Analysis Using Relational Structures, Pattern Recognition 17, 1984, pp. 161-167

[9] N.S. Chang, K.S. Fu, Query-By-Pictorial-Example, IEEE Trans. SE-6, 1980, pp. 519-524

[10] M.M. Zloof, Query By Example, IBM-Research Yorktown Heights, TR. RC. 4917, July 1974

[11] C.J. Date, An Introduction to Database Systems, 3rd Ed., Addison-Wesley Reading Mass., 1981

[12] W. Benn, B. Radig, Entwurf einer Relationalen Datenbank zur Unterstützung der Analyse von Bildfolgen, Mitteilung des Fachbereichs Informatik der Universität Hamburg, IFI-M-116, Dezember 1983

[13] G. Goos, W.A. Wulf (Ed.), Diana Reference Manual, Bericht 1/81 der Fakultät für Informatik der Universität Karlsruhe, 1981

[14] B. Radig, R. Kraasch, W. Zach, Matching Symbolic Descriptions for 3-D Reconstruction of Simple Moving Objects, Proc. 5th Int. Conf. Pattern Recognition, Miami Beach, Dec. 80, pp. 1081-1084

Singapore, August, 1984