Mitteilung 115

RETRIEVAL OF RELATIONAL STRUCTURES FOR IMAGE SEQUENCE ANALYSIS

Wolfgang Benn und **Bernd Radig**

IFI-HH-M-115

November 1983

Universität Hamburg

Schlüterstraße 70

D-2000 Hamburg 13

Inhaltsangabe:

Die automatische Verarbeitung von Realweltszenen erfordert die symbolische Beschreibung von Bildern einer Szene und deren Bildinhalten. Eigenschaften von Bildinhalte beschreibenden Symbolen und deren Beziehungen untereinander lassen sich durch Relationengebilde formalisieren. Die Interpretation von Bildfolgen stützt sich im wesentlichen auf die Gruppierung einfacher Bildsymbole zu komplexen, Bildobjekte beschreibenden Strukturen und die Aufstellung von Korrespondenzbeziehungen zwischen Symbolen und Objekten aufeinanderfolgender Szenenbilder.

Die genannten Vorgänge liefern eine der Komplexität von Realweltszenen entsprechende Menge von Beschreibungsdaten und erfordern dadurch eine systematische Datenorganisation. Das hier vorgestellte Datenbanksystem erfüllt diese Forderung durch Spezialisierung auf Handhabung und Verwaltung von Relationengebilden zum Suchen und Vergleichen von nicht vollständig übereinstimmenden Bildbeschreibungen. Die Datenbank ist in ein Prozessrechnernetz integriert und residiert auf einem eigenen Prozessor.

RETRIEVAL OF RELATIONAL STRUCTURES FOR IMAGE SEQUENCE ANALYSIS

Wolfgang Benn and **Bernd Radig**
Fachbereich Informatik, Universität Hamburg
Schlüterstraße 70, D-2000 Hamburg 13

1.0 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.

Interpretation of real-world, outdoor scenes tends to produce such an amount of data that the systematic management of symbolic descriptions becomes mandatory. 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".

2.0 INTRODUCTION

Picture database systems are specialized on storing large amounts of pictures together with descriptional information [1]. Conventional database concepts are sufficient to access the pictures by queries related to their descriptions [2]. Enhancing the query language by graphical elements allows the users to include pictorial information directly in their queries. Chang and Fu built their Query-by-Pictorial-Example (QPE) [3] on top of Zloof's Query-by-Example (QBE) approach [4]. Applications in cartography often include special operators in the data manipulation language which compute relationships between picture objects such as distance between two points [5].

A different utilization of a database management system is its integration in a picture understanding system. Especially 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 [6] [7]. 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 (geometrical, hierarchical, correspondence, ...) relationships are formalized as relations. Relational structures are collections of tuples from those relations which describe a distinct entity, e.g. an object model [8]. 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.

Storing and retrieving of relations is well supported by existing database concepts. Image sequence analysis will benefit from a database system if it supports also the storage of relational structures (cf. the entity-relationship model [9]) and the retrieval of relational structures which match a given template most closely. We call this kind of retrieval "Query by Structure Example" (QSE).

3.0 RELATIONAL STRUCTURES

A set C contains all elements - symbol identifiers, attribute values - which are needed for the description of an image sequence and of object prototypes.
$R_1 \ldots R_n$ are relations defined on C, where each element of R_i is an t_i-ary tuple:

$$R_i \subseteq C^{t_i} \ .$$

$$RS = [C, \langle R_1, R_2, \ldots, R_n \rangle]$$

denotes a relational structure. The tuple $\langle t_1, t_2, \ldots, t_n \rangle$ is the type of RS which indicates for each relation in the structure the number of its components.

Matching of relational structures is reformulated as the task of finding a mapping between both structures; such a mapping will be called a RS-morphism [10]. Let

$$RS' = [C', \langle R'_1, R'_2, \ldots, R'_n \rangle]$$

be a homologous relational structure. To compare both structures, e.g. for the purpose of change detection between two picture objects, mappings between the two relational structures have to be generated. To facilitate the computation of these mappings, and to select eventually the best of them according to some criterion to be defined, they are restricted in some natural way. Each mapping

$$\varphi: \ RS \rightarrow RS'$$

is a many-to-one, in general. The mapping φ is composed of several submappings. Since it is only meaningful to associate tuples of a relation in RS with tuples of the same kind of relation in RS', n disjoint submappings are

$$\varphi_{R_1} : R_1 \rightarrow R'_1 \; \ldots \; \varphi_{R_n} : R_n \rightarrow R'_n$$

The set C - as well as C' - can be decomposed into two subsets CK and CQ. CK contains the unique *K*eys by which symbols are identified, CQ holds values of attributes which assign some kind of *Q*uality to the relations where applicable, e.g. **SMALLER (region1, region2, fraction)**. CK may be further split into disjoint subsets each containing only symbol identifiers of the same class, e.g. regions or line segments.

$$CK_{SI1} = \{SI1_1, \ldots, SI1_N\} \; \ldots \; CK_{SIk} = \{SIk_1, \ldots, SIk_M\} \, .$$

Between the sets C and C', φ maps only the symbol identifiers such as

$$\varphi_{SI1} : CK_{SI1} \rightarrow CK'_{SI1} \; \ldots \; \varphi_{SIk} : CK_{SIk} \rightarrow CK'_{SIk} \, .$$

For the attribute values appropriate compatibility functions

$$\theta : CQ \cup CQ' \rightarrow [0,1]$$

are defined which decide if mapping of two tuples is allowed with respect to their attribute value components and a chosen threshold $\boldsymbol{\theta}$.

In general, each RS-morphism is a mapping-triple of

$$\varphi : \langle \varphi_R , \varphi_K , \theta_q \rangle$$

and the following condition holds

$$t' = \varphi_R(t), \; si' = \varphi_K(si) \text{ and } \theta_i \; (\ldots, ch, \ldots, ch', \ldots) > \boldsymbol{\theta}_i \, ,$$

$$\text{where } t = (\ldots, si, \ldots, ch, \ldots) \in R, \; t' = (\ldots, si', \ldots, ch', \ldots) \in R',$$

$$si \in CK, \; si' \in CK', \; ch \in CQ \text{ and } ch' \in CQ' \, .$$

θ_i and $\boldsymbol{\theta}_i$ are the compatibility function and the threshold for the i-th relation, respectively.

When comparing two relational structures $\boldsymbol{RS}$ and $\boldsymbol{RS}'$ this formal definition means

* Every tuple of a relation in $\boldsymbol{RS}$ is mapped to at most one tuple in $\boldsymbol{RS}'$ which has the same number of components due to the homology of $\boldsymbol{RS}$ and $\boldsymbol{RS}'$.
* Mapped tuples are compatible with respect to their attribute values.
* The mapping of symbols, induced by the mapping of tuples, is a many-to-one mapping, too.

Since the definition of RS-morphisms is still too general for the intended application, it is specialized in the following way:

* RS-*homomorphisms* map all tuples of all relations in $\boldsymbol{RS}$ to some tuples of all relations in $\boldsymbol{RS}'$; the whole structure $\boldsymbol{RS}$ is part of $\boldsymbol{RS}'$.
* RS-*monomorphisms* are one-to-one RS-homomorphisms. Structure $\boldsymbol{RS}$ corresponds exactly with a substructure of $\boldsymbol{RS}'$.
* RS-*isomorphisms* are RS-monomorphisms with a monomorphistic inverse mapping. It is a bijective one-to-one mapping $t_i \longleftrightarrow t'_j$ with $t_i \in R_i$ and $t'_j \in R'_i$.
* RS-*comorphisms* are RS-monomorphisms of a subset $\boldsymbol{RS}^* \subseteq \boldsymbol{RS}$ into $\boldsymbol{RS}'$ and the subset $\boldsymbol{RS}^*$ is maximal. It is maximal if there exists no other RS-monomorphism $\varphi^+ : \boldsymbol{RS}^+ \longrightarrow \boldsymbol{RS}'$ such that $\boldsymbol{RS}^* \subseteq \boldsymbol{RS}^+$:
  $\boldsymbol{RS}^* \subseteq \boldsymbol{RS}^+ \subseteq \boldsymbol{RS} \iff \forall i = 1..n : R_i^* \subseteq R_i^+ \subseteq R_i$.

To retrieve a relational structure from the database, the user program has to specify a relational structure $\boldsymbol{RS}$ and the kind of RS-morphism to apply. The database system then will return all relational structures RS' which it is able to match with RS. Especially incomplete and inexact matching structures are obtained searching for an RS-comorphism with compatibility thresholds $\theta_i < 1$.

The most simple query-RS consists of just one tuple of a relation. A complex RS may contain the symbolic description of a whole image, and the database system is requested to find another stored image description with largest common substructures.

4.0 REQUIREMENTS ON THE DATABASE SYSTEM

The most simple transactions which are currently implemented, are elementary operations to handle single tuples. On this base the more complex transactions for storing and retrieving of complex relational structures will be implemented.

4.1 Elementary Functions

The use of our database system is demonstrated by the following example from the analysis of an image sequence [11]. The image of a moving car (Fig. 1) is marked by critical points. They are described by a relational structure containing only the relation LOCATION which assigns to each point a unique identifier, the image number, and its coordinates . Fig. 5 shows an ADA-type delcaration of LOCATION. Corresponding points are linked together from image to image. A correspondence relation is added to the relational structure. It is visualized in Fig. 2. From the point sequences depth information is obtained (motion parallax) and the object - now represented as a cloud of 3D-points - is approximated by the convex hull of this cloud (Fig. 3). The convexe hull is built from triangles. Their relational structures consist of a triangle-symbol, three attached sides of type line-segment and their endpoints of type location (Fig. 4a). To store and retrieve this kind of relational structures on a tuple-by-tuple basis, elementary functions are implemented:

* Insert and delete relations and tuples
* Modify tuples
* Retrieve tuples in sequential order or selected by primary keys.

The internal links of tuples allowing system defined sequential tuple access have to be updated after insertion or deletion. Each tuple gets a system defined primary key for identification (see Fig. 5: record component TID of type SYSTEMKEY). All operations except sequential retrieval utilize this identifying key for a fast access. Sequential transactions need an access-mode specification like "FIRST", "NEXT" or "LAST" [12].

4.2 Conceptual Database Scheme

A user relation, stored in the database, is described by a tuple of the data-directory relation. One component of the directory tuple references an attributed graph (see Fig. 6: DB-ATTRIBUTE) which is also stored as a relational structure. This graph describes the attributes of the user relation. It is automatically generated from the inspection of each application program. The user programs which interact with the database system are written in ADA. Our ADA-compiler generates the intermediate ADA-program representation according to the specifications of the DIANA [13] language. For each type declaration, also for record types which serve to declare a user relation, a DIANA-subtree has been created. These subtrees are stripped from unnecessary nodes (from the database point of view) and stored as part of the data-directory. The complete set of stored subtrees represent the conceptual database scheme which is used to guide the transformations between internal and external schemes.

An example declaration of an ADA-record in Fig. 5 is represented by the DIANA-subtree in Fig. 6. A compatibility check between the user-program view of a relation and the conceptual scheme will be performed by testing for a RS-isomorphism between the stored and the compiler generated subtree.

4.3 Memory Addresses And Primary Keys

The image analysis program which generated the Figs. 1 - 3 is written in PASCAL and now transformed to ADA. A typical representation of the triangle's relational structure in memory during program execution is depicted in Fig. 4b.

The triangle symbol is represented as a record whose components are a symbol identifier (TID4) and three pointers to three line segments (TID1, TID3, TID6) which constitute the sides. Each line segment in turn references its two endpoints (TID2, TID5 or TID7).

When transfering such a relational structure to the database the references which are memory addresses loose its meaning and have to be replaced by another figure identifier: the tuple identifier TID. The corresponding structure in the database (see Fig. 4c) consists of three relations, here called LINE, LOCATION and TRIANGLE, with the same attributes as the records but the pointers replaced by the TID of those symbols the pointers referred to.

Within the relational structures, TID's serve as image identifiers for symbols. Furthermore, the value of a TID is determined by the database management system. This offers the opportunity to use this value as an indicator for the disk address where the identified tuple resides. Thus retrieval of an tuple by its TID is extremly fast.

4.4 Query By Structure Example

Our database system is not designed as a dialogue system for human interaction. Its only purpose is to relieve any image analysis program of transfering symbolic descriptions of images, models, prototypes, etc. between memory and background storage.

Storing a relational structure requires from the database system to replace pointers by TID's and to assign TID's to those tuples which enter the database the first time.

Retrieving a relational structure requires from the user programm to provide an example structure ***RS*** and to indicate which kind of RS-morphism the database system should use to match ***RS*** with the stored relational structures ***RS'***. A structure resulting from this mapping overwrites the query structure in the user memory. In doing so the TID's in referencing attributes are replaced by pointers so that the user program immediately is able to operate on the retrieved relational structure.

If the database system resides - as in our implementation - on a seperate processor with its own disk, storing and retrieving can be performed in parallel with the execution of the user program, thus increasing the performance of the image analysis system.

The exchange of TID's with pointers is performed in the database processor.

This mapping is done by a composition of two binary, one-to-one relations.

We define the set of tuples belonging to a requested relational structure as

$$T \subseteq RS$$

and the set of addresses which reference the set T in the memory of a computer

$$RT = T \times R \text{ with } R = \{\mathbf{adr} \mid \mathbf{adr} \in [\mathbf{addressrange}]\}.$$

Attributes which link the tuples of the relational structure are values from the domain RT which will be called D_{RT} . In a distributed system this domain is different for every computer of the network. So D_{RT} will be marked by **AP** and **DB** to denote the application program computer and the database computer.

To build up an example structure the application program uses a standard procedure to create tuple representing records in its memory, say in the heap. The link of records by pointers causes all referencing attributes to get values from D_{RT}^{AP} .

Now the database gets the transaction command. With

$$\psi_R : D_{RT}^{AP} \longrightarrow D_{RT}^{DB} \implies \psi_R(\mathrm{adr}) = \mathrm{adr}' \text{ and } (\mathrm{adr} \in D_{RT}^{AP} , \mathrm{adr}' \in D_{RT}^{DB})$$

the database transfers the requested structure from the application program into the database computer. This projection ψ_R will be represented by an internal binary relation Ψ_R with key values from the domain D_{RT}^{AP} .

Then the tuples will be examined wether they contain a value in their attribute *systemkey* from the domain

$$TID: \{\mathbf{key} \mid \mathbf{key} \in [\mathbf{keyrange}]\}$$

where the interval [**keyrange**] will be implementation defined (see 4.3). If there is a value it is assumed that the identified tuple is a member of the relational structure to be retrieved.

References in the stored data are from the domain TID so that a new projection

$$\psi_K : D^{DB}_{RT} \longrightarrow \mathit{TID} \Longrightarrow \psi_K(\mathrm{adr}') = \mathrm{tid} \text{ with } (\mathrm{adr}' \in D^{DB}_{RT} \text{ and } \mathrm{tid} \in \mathit{TID})$$

has to be created and represented by another binary relation Ψ_K with keyvalues from the domain D^{DB}_{RT} . ψ_R and ψ_K can be composed to give

$$\psi: D^{AP}_{RT} \longrightarrow \mathit{TID} \Longrightarrow \psi(\mathrm{adr}) = \mathrm{tid}$$

which is a bijective projection, represented by a new relation Ψ which is created by a natural join over the attribute values from D^{DB}_{RT} . We choose the natural join operation to clarify that ψ is a bijective projection, but in this special case an equi join also satisfies the bijectivity condition because ψ_R and ψ_K are bijective, too, during their existence at the time of a transaction.

To store a relational structure, ψ is applied. To retrieve a relational structure, ψ_R is applied to transfer the example structure from the application to the database processor and ψ_K^{-1} reads candidate structures from the database into the database processor. After a successful match is found, ψ_R^{-1} facilitates the transfer of the structure to the user program.

5.0 SYSTEM INTEGRATION

The programs for the analysis of image sequences are executed on a minicomputer network consisting of four DIETZ MINCAL 621x2. They are connected by a 16 bit parallel bus. The peripheral equipment allows the recording and digitization of TV-frames. Since application programs up to now have been written in PASCAL, the PASCAL runtime supports contains modules to control the video-periphery, the raster-scan display systems and the network communication. For user and system programs a rewriting from PASCAL to ADA is in progress.

One of the network computers is dedicated as a back-end processor for the database management system. It is equipped with its own 60 MByte disk and half a Megabyte of working memory. The physical separation of database system processes and user programm - connected only by a message exchange facility -

eases program development of the database system. Moreover, a true advantageous parallel execution of a user program and database transactions is possible. This is an enormous improvement, since the time consuming search for inexact or incomplete matching structures can be left entirely to the database processor, when the Query-by-Structure-Example will be completely implemented. The database management system is designed as a bunch of interacting processes within the database computer [14]. Four fundamental processes are realized which

- handle the communication to the other network components,
- perform disk transfers,
- implement the storing at retrieving of relations and tuples,
- and monitor the database system.

A further process will

- analyse the DIANA representation of ADA application programs. All information relevant to the database system is extracted automatically from the program. A special data definition language becomes unnecessary. A next process will
- implent the search for RS-morphism [10, 15] between relational structures as the basis for the Query-by-Structure-Example.

Due to the modular design and the clearly defined interfaces between the processes, a gradual upgrading of the database managment system is possible.

6.0 LITERATURE AND FIGURES

[1] H. Tamura, N. Yokoya
Image Database Systems: A Survey
Proc. 6th ICPR
München, Oct.1982; Pattern Recognition 17 (1984)

[2] Y.E. Lien, D.F. Utter jr.
Design of an Image Database
Proc. IEEE Workshop on Picture Data Description and
Mangement, pp. 131-136, 1977

[3] N.S. Chang, K.S. Fu
Query-By-Pictorical-Example
IEEE Trans. SE-6 (1980) 519-524

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

[5] M.I. Chock
A Data Base Management System for Image Processing
Dissertation University of California, 1982

[6] 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

[7] 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

[8] B. Radig
Hierarchical Symbolic Description and Matching of Time Varying Images
Proc. 6th ICPR, München, Oct. 1982, pp. 1007 - 1010

[9] P.P. Chen
The Entity Relationship Model: Towarad unified view of data
ACM-TODS, Vol.1, 1976, pp. 9 - 36

[10] B. Radig
Image Sequence Analysis Using Relational Structures
Pattern Recognition, 17 (1984), see also Mitteilung IfI-HH-M-106
des Fachbereichs für Informatik der Universität Hamburg, 1983

[11] L. Dreschler, H.-H. Nagel
Volumetric Model and 3-D-Trajectory of a Moving Car Derived
from Monocular TV-Frame Sequences of a Street Scene
Comp. Graphics and Image Proc. 20 (1982) 199 - 228

[12] W. Benn
Bildfolgenbank
Projektberichte 1 bis 7, Fachbereich Informatik
der Universität Hamburg, Januar 1982 bis April 1983

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

[14] W. Benn, B. Radig
Entwurf einer Relationalen Datenbank zur Unterstützung der Analyse von Bildfolgen
Mitteilung des Fachbereichs Informatik der Universität Hamburg, in preparation

[15] B. Radig
Symbolische Beschreibung von Bildfolgen I:
Relationengebilde und Morphismen
Bericht IFI-HH-B 90/82 des Fachbereiches Informatik der Universität Hamburg

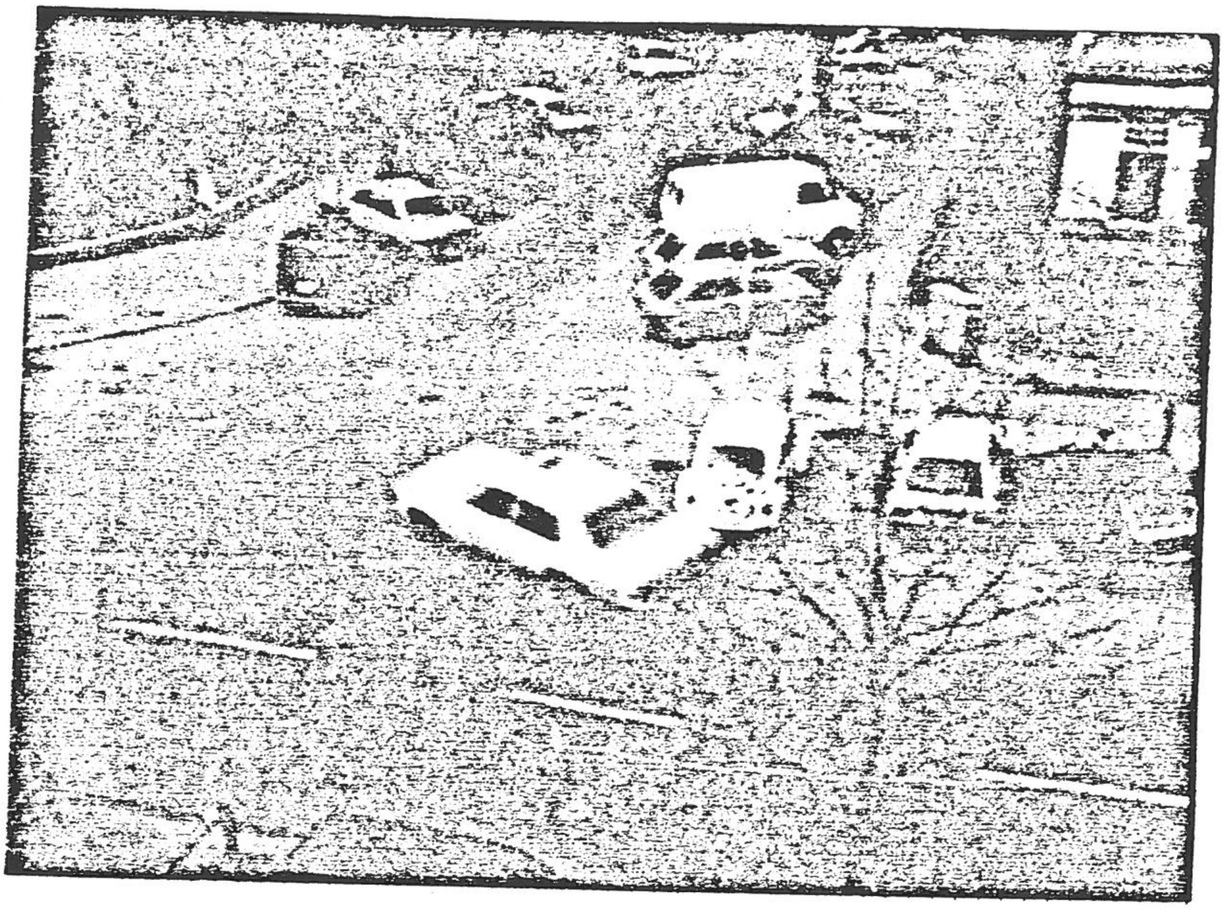
Figure 1 : original picture from an outdoor-scene

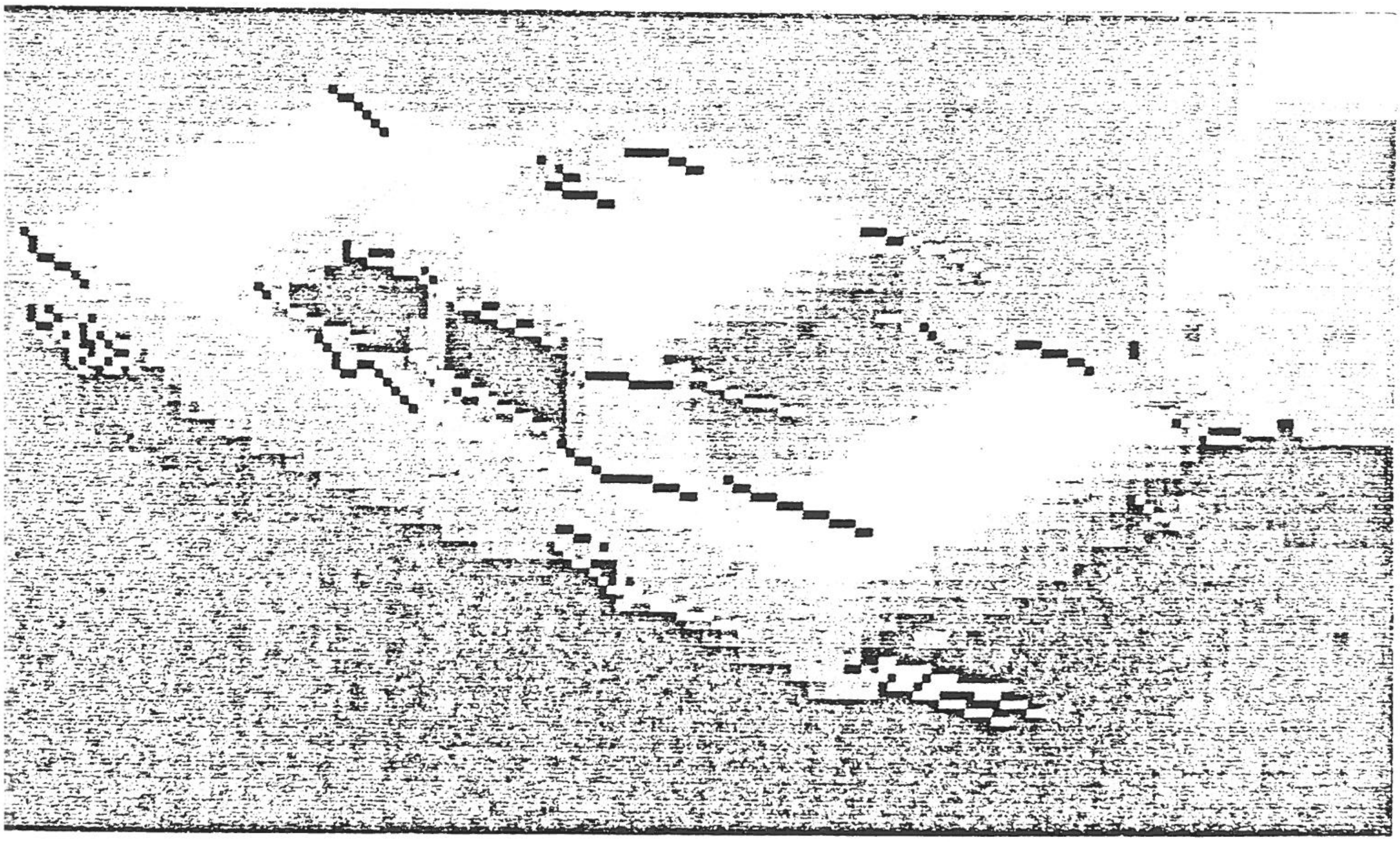
Figure 2 : chains of correspondency

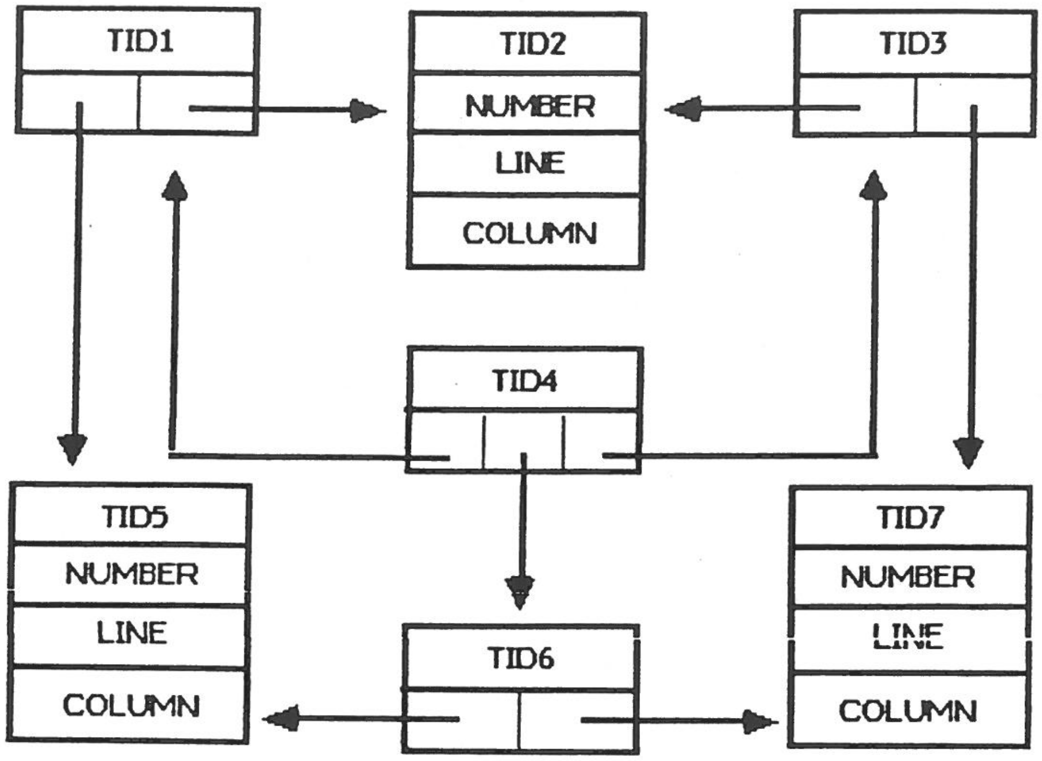

Figure 4b : triangle as net of records

TRIANGLE

TID	SIDE1	SIDE2	SIDE3
tid4	tid1	tid3	tid6

LINE

TID	STARTLOC	ENDLOC
tid1	tid5	tid2
tid3	tid2	tid7
tid6	tid5	tid7

LOCATION

TID	NUMBER	LINE	COLUMN
tid2	---	---	---
tid5	---	---	---
tid7	---	---	---

Figure 4c : triangle as a set of relations

```
type SYSTEMKEY is new integer;

type LOCATION is record
     TID       : SYSTEMKEY;
     NUMBER    : NUMBERRANGE;
     LINE,
     COLUMN    : COORDINATERANGE;
end record;
```

Figure 5 : Exampledeclaration in ADA

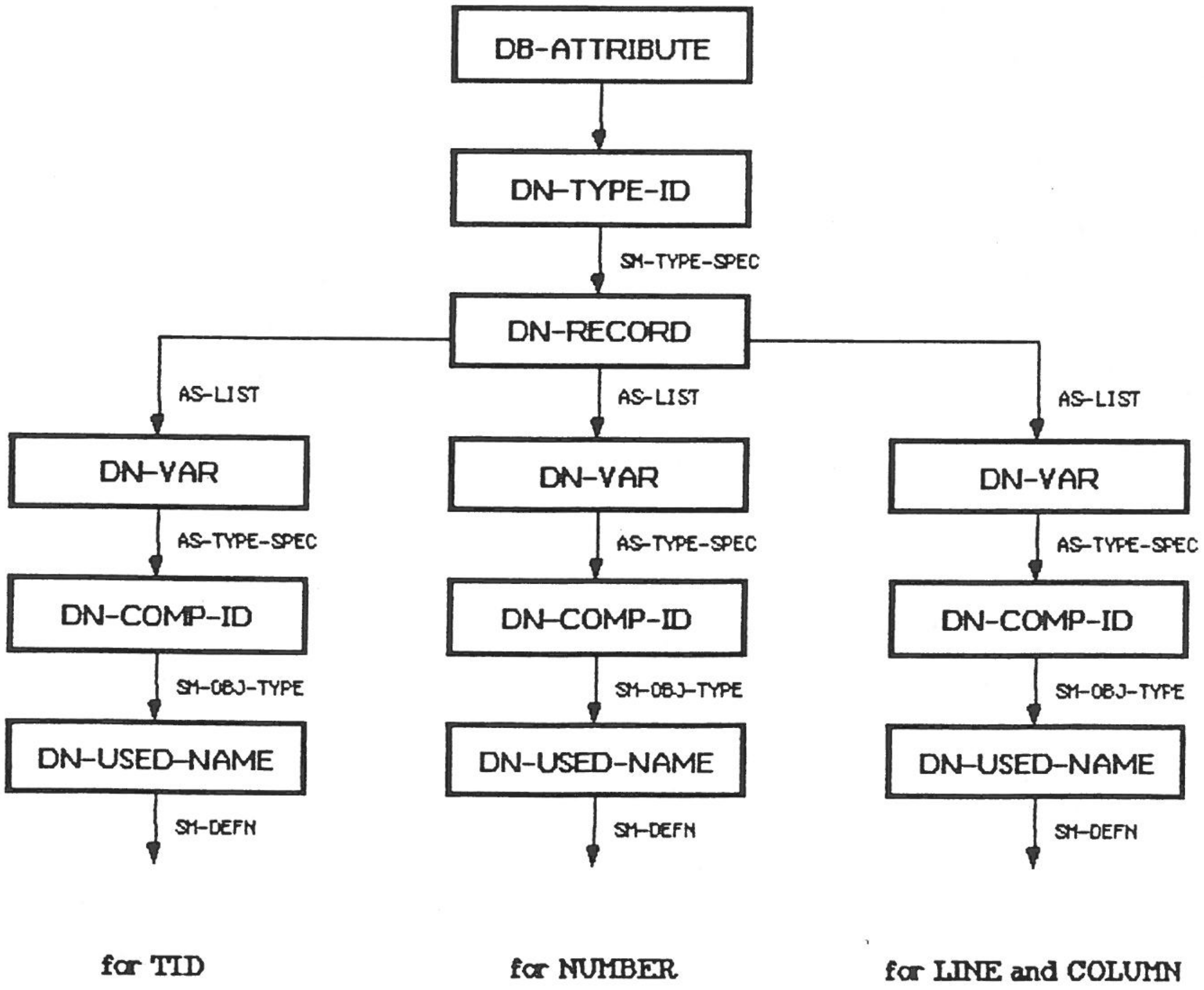

Figure 6 : simplified DIANA-representation for tuple LOCATION

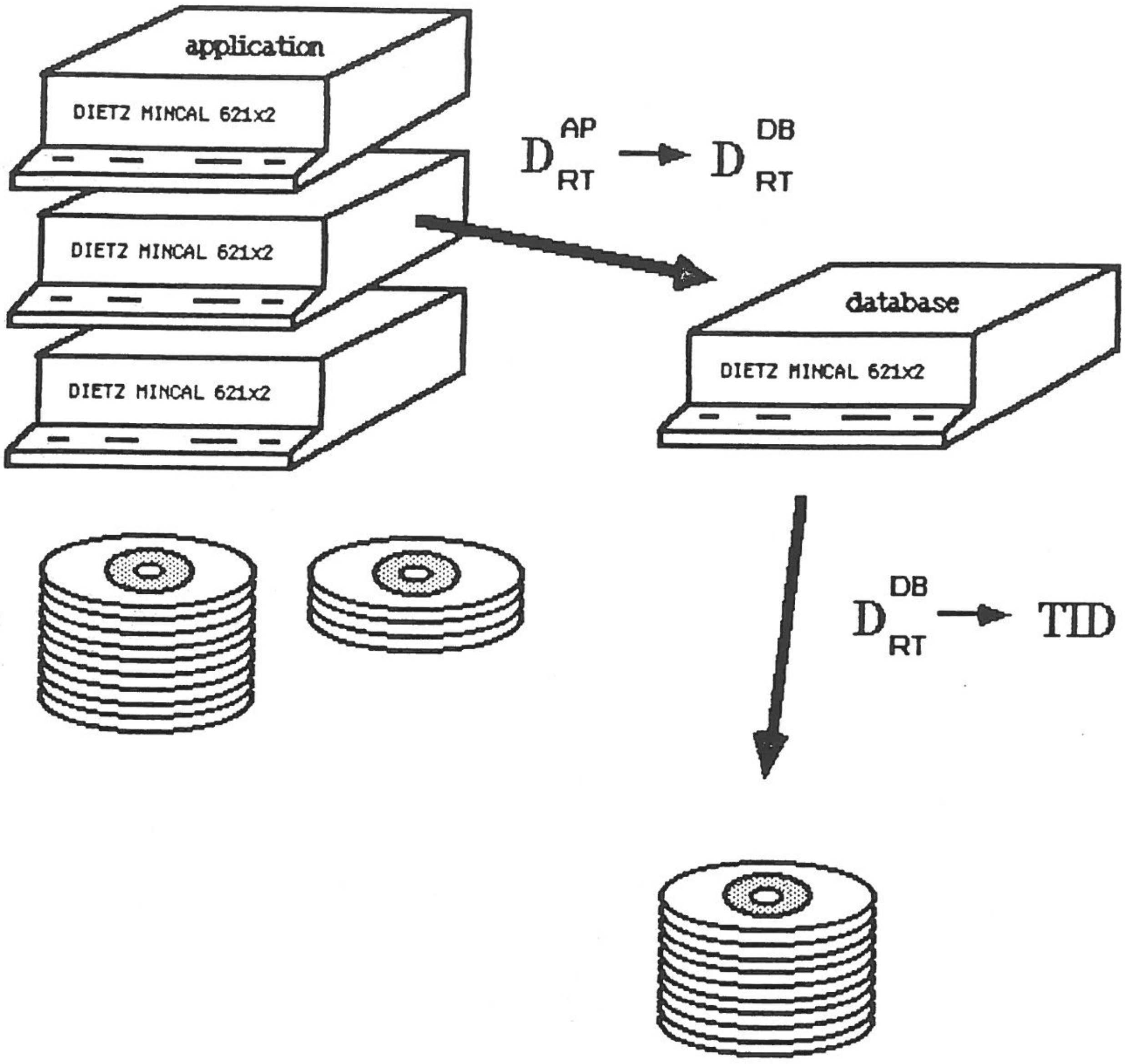

Figure 7 : projection of references