WEB-BASED USER INTERFACE FOR QUERY SPECIFICATION IN A VIDEO DATABASE SYSTEM

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ediz Şaykol

September, 2001

ABSTRACT

WEB-BASED USER INTERFACE FOR QUERY SPECIFICATION IN A VIDEO DATABASE SYSTEM

Ediz Şaykol
M.S. in Computer Engineering
Supervisors: Assist. Prof. Dr. Uğur Güdükbay and
Assoc. Prof. Dr. Özgür Ulusoy
September, 2001

The enhancements in multimedia technology have accelerated the development of new techniques for querying and retrieval in video databases. Based on these new techniques, effective user interfaces are designed especially for query specification. Since the Internet is a suitable multiple client environment, most of the video database systems are designed to work over the Internet. In this thesis, a web-based user interface for a video database system is presented. The interface is composed of a hierarchical windows-based structure where logically separable sub-queries are handled in separate windows. For example, spatial and motion queries are handled separately. The separation of the interface aims to facilitate the query specification process. Furthermore, the user is allowed to get partial results for the sub-queries. The retrieval and result presentation process displays the query results in the form of video frame sequences on a separate window with the capability of playing each result separately. As another contribution of this thesis, considering the fact that most of the systems include querying by object features, a novel approach for color and shape querying and retrieval is proposed. The proposed approach is histogram-based and it is shown through the performance experiments that it overcomes the deficiencies of the existing methods. This new approach is supported by an auxiliary object extraction tool in the query-by-feature sub-system of the video database system. The object extraction tool is semi-automatic, hence it can successfully capture salient object regions for most of the images and/or video frames.

Keywords: Web-based query specification interface, object extraction, flood fill for extraction, distance histogram, color histogram, kinematics model for polygon simplification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The progress in multimedia technology has motivated the development of efficient compression and storage techniques for multimedia data (e.g., image and video). Multimedia data is usually stored in a special type of database, called *multimedia database*, which requires the development of sophisticated techniques for querying and retrieval of this data.

As far as querying and retrieval of multimedia data are concerned, spatial, spatio-temporal, semantic, motion (e.g., object trajectories, relative object movements) and object features (e.g. color, texture, shape, size) are taken into consideration. Since the system that we experiment on [10] is a video database system, object motions have a significant place during query formulations as well as spatio-temporal content of video data. The system is intended to serve for multiple clients, thus a multi-user environment is required (e.g., the Internet).

In the scope of the thesis, the user interface for the video database system that we have worked on forms the main point. Based on the multi-user environment idea, the user interface is web-based. Besides, since the system has to be capable of querying by object features (e.g., color, shape, texture), a new approach is proposed for querying by color and shape. Thus, two of the features for the query-by-feature sub-system of our video database system are filled by this new approach. Moreover, an auxiliary tool, *Object Extractor* [44], is designed and

implemented for capturing the object regions on video frames. The output of this tool is used not only by the new query-by-color and shape approach but also in fact extraction [12].

1.1 Motivation

In [27], a basic understanding of multimedia systems is presented. In a multimedia system, the user interface takes a critical role. The main requirement of multimedia user interfaces is basically enabling the combination of various query mechanisms with the support of visual query languages. Since the multimedia data is visual, the user interface should be as much visual as possible with the aid of relevance feedback and user-guided navigation on the data.

In general, for a conventional database containing text and numerical data, the query condition is represented in the form of alphanumerical characters but in the case of multimedia databases, where image, audio, video data is stored as well, such a query representation may not make sense. When the keywords in the query mean something visual, expressing the query in textual form may not reflect its intended meaning to the query. An example can be the query 'photographs with a gray cloud' that is represented in the form of alphanumerical characters. The degree and brightness of the color gray have to be included in the query so that the system returns the correct results. On the other hand, by using Query-by-Example (QBE) [50], no extra information is needed in the query because an example of the cloud with desired gray color is supplied to the query and the data having the same degree and brightness will be returned if exists. As a result, for multimedia data, an example expresses the query more naturally than words.

Moreover, QBE provides an intuitive way of representing the query by enabling the representation of the query condition to resemble the features of the data and offers a better way by eliminating the possibility of incorrect but closest retrievals. Mainly, QBE is mostly used with the retrieval of image data where the above considerations are important.

A comparison between a query-by-example based visual interface and a forms-based interface is made in [20]. The authors conduct experiments on a group of trained students and the comparison criteria include efficiency, accuracy and user satisfaction. It is shown through the tests that visual specification is more efficient, accurate and user-friendly than forms-based specification.

1.2 Organization of the Thesis

The thesis is organized as follows: Chapter 2 summarizes the studies in the literature on content-based retrieval, visual query languages and interfaces, result presentations and interface design instructions. In Chapter 3, the web-based querying interface is explained through the query specification details. Chapter 4 includes the explanation of the object extraction tool that is used to query object-based information in the system. The performance evaluation of the tool is presented in the chapter. In Chapter 5, a novel approach for querying by color and shape is discussed in detail. The approach is evaluated with a well-known shape retrieval technique and the performance results are presented in the chapter. Finally, Chapter 6 concludes the thesis.

Chapter 2

Related Work

2.1 Content-Based Retrieval Systems

There exist quite a number of content-based retrieval (CBR) systems in the literature: QBIC [15], VisualSEEk [42], VideoQ [7], Photobook [32], ImageRover [38], Surfimage [29]. Detailed surveys on CBR systems are available in [46] and [48]. In most of these systems, color is a *must* object feature for the retrieval process. However, some of the systems allowing color queries do not develop new techniques for color but in other areas such as spatio-temporal relations. Section 5.1 presents a detailed summary on color and shape approaches for CBR systems.

In QBIC system [15], the images and videos can be queried with color, shape, texture features and visual sketches. In order to extend the image-based feature specifications to video data, specific techniques are implemented for keyframe processing (*representative frame generation*). This system is a milestone for a complete retrieval system and most of the other systems develop techniques based on that of QBIC system.

Smith and Chang introduced VisualSEEk system for content-based image querying [42]. They proposed *colorsets* for color querying instead of *color histograms*, as it is done in QBIC. Besides, the texture and spatial content of the

images can be queried. Since the system focuses on images, no spatio-temporal querying is possible. The VideoQ system [7] proposed by Chang et al. includes querying by spatio-temporal (e.g., motion) content as well as color, shape, texture and size.

In Photobook [32], shape and texture features are used for querying. The basic motivation of this system is to embed artificial intelligence for semantic querying. Thus, the designers of Photobook concentrated on semantic indexing.

Surfimage [29] contains color, shape and texture features (low-level) as well as eigenimages, flexible images, and image shape spectrum (high-level). The user selects appropriate image features and assign weights representing the importance in query specification.

ImageRover [38] is a search by image content navigation tool on the Internet. The tool includes color and texture queries and both query-by example and query-by-keyword features in the query specification interface.

2.2 Visual Query Languages and Interfaces

In order to represent spatio-temporal relationships among salient objects, various types of data structures are designed for query processing. As a direct consequence, each specific data representation requires a specific query language. Since the spatio-temporal behavior of video data is well-suited for visual representation, the query languages are *visual query languages*. In order to provide a visual query specification environment,*visual query interfaces* are implemented.

The spatio-temporal content of video data not only consists of spatial and temporal relations among salient objects, but also the (relative) motion of the salient objects. Depending on the design and implementation strategy, the data models usually concentrate on one of these features of video data and provide automated environments as much as they can. For example, some of the systems focus on spatial relations for images and/or video frames (e.g., [24]); some of them

deal with motion of a salient object through the video (e.g., [49]); while some of the models include descriptions for all types of spatio-temporal content (e.g., [9]). One common property of the systems including specification of spatio-temporal relations is to use Allen's temporal algebra [1].

In [31], VisualMOQL is presented as the visual query interface of DISIMA distributed image database management system. Since the system was designed their model for images, the time dimension is not incorporated during query specification. The images in the database are categorized into user-defined classes and the query is initiated with the proper selection of the image class. Then, salient objects are selected. The topological relations among the objects are automatically added but the user has to specify the directional relations manually. A simple query is formulated in this fashion and the user can construct compound queries via logical operators in the query interface.

Query-by-Trace [13], is a visual interface for the specification of temporal predicates for temporally changing spatial situations. Two types of primitives are defined, *moving point* and *evolving region*, denoting the motion and change in size for salient objects. Queries between two moving points or two evolving regions are possible as well as queries between a moving point and an evolving region. Having selected the appropriate primitives for the query, the mouse is traced on the canvas and the spatio-temporal predicate sequence is specified according to the mouse locations. As mentioned in [13], this visual specification can be directly used as a visual query interface, or can be integrated into a textual query language for multimedia databases.

SVIQUEL [24] is a spatial visual query language with an easy-to-use query specification interface. An object is represented with its minimum bounding rectangle oriented along the x and y axes, and the spatial relations among the salient objects are deduced according to these rectangles. The interface is equipped with *S-sliders* for specifying the spatial relations. Since the interface is designed for only two salient objects, eight S-sliders corresponding to the top, bottom, left, right coordinate differences would suffice. S-sliders provide an easy way to adjust the relations during query specification. For topological relations, one of the

objects is fixed on the interface and the other is placed into the pre-specified locations around the fixed object. Thus, a spatial query for two salient objects is formalized.

In [19], Multimedia Visual Information Seeking Environment (MMVIS) is proposed for the specification of temporal predicates among events. This specification is based on Temporal Visual Query Language (TVQL) [18]. The events are pre-annotated and the annotated subsets are selected at the beginning of the query specification process. Then, two events are chosen from the subsets. In TVQL, the temporal predicates between two events are specified by sliders denoting the differences between start and end times of the events. In the system, the sliders are enhanced so that they can represent the relative time difference. Another important improvement on the sliders is that *at least* and *at most* type temporal queries can be specified via sliders. In the system, the queries are processed by query filters and visualized in the temporal visualization environment. Similarly, this environment shows the temporal relations between events in the enhanced slider form.

In VIOLONE [49], methods for motion specification and retrieval in video data are proposed. In the system, the motion queries are specified in a query-by-example fashion. In order to specify a trajectory for a salient object, the user moves the mouse on the canvas and the trajectory is the set of mouse locations. During the mouse movement, the velocity of the mouse becomes the velocity of the salient object. Unfortunately, this type of motion specification is inadequate for more than one salient object. The reasons are the difficulty of motion specification in query-by-example manner and the difficulty of the user's concentration on multiple motions. Since the velocity is automatically gathered from the velocity of the mouse, specifying multiple motions may yield incorrect or meaningless relative velocities between motions.

Spatio-Temporal Logic (STL) for the symbolic representation of spatio-temporal relations between salient objects in image sequences is presented in [9] together with a visual query-by-example interface. During database population, each image sequence is associated with a symbolic description representing the

spatio-temporal relations through the frames. Since the authors focus on 3D scenes, a *Scene Editor* with zooming capability to arrange the positions of the objects is designed. The scenes are automatically parsed and expressed in STL. In the retrieval process, the matching between the symbolic descriptions stored in the database is performed.

There exist different approaches for query specification besides these visual query interfaces. In [22], a method that allows the user to give multiple examples with their *goodness* scores is proposed. In this way, the correlations are taken into consideration via the goodness scores. Another approach, IFQ [26], is a query generator rather than a visual graphical query interface. In this system, the user initiates query specification by introducing image objects. Then, the user describes these objects and concludes query specification process by specifying the spatial relations of the objects. The underlying query language provides cognition-based and semantic-based predicates for queries. Santini and Jain propose that the query specification phase has to include the correlation between the intended meaning of an image and perceptual clues that the database can extract [35]. For this purpose, they designed *direct manipulation interface* where the global view of the database images is presented to the user. As a result, the user can easily manipulate the query specification environment. Another work [36] by Santini and Jain presents a contextual way of representing a query. The user configures a set of images on a screen and the distances between images denote their mutual similarity. Thus, the query-by-example image somehow gains a meaning and the query becomes more accurate.

2.3 Presenting Results and Design Guidelines

In most of the visual query processing systems, the presentation of the results is very similar. Following the specification of the query and the initiation of the query processor, a browser-like interface appears where the database images and/or videos are listed in sorted similarity value order. In most of the cases, the size of the query results to be retrieved depends on the user's

selection [7, 15, 29, 32, 38, 42].

VideoZoom system, proposed by Smith in [40], is a web-based video browsing system. It incorporates a new paradigm for video retrieval where browsing and retrieval are highly correlated. A *video view element graph* is presented to the user from which the retrieved *view elements* participate in zooming the video segments. This new paradigm can be integrated to the result presentation phase of a multimedia querying system.

Najjar presents a comprehensive set of guidelines for multimedia interfaces and presentation [28]. Simple, consistent, and user-friendly interfaces are given more importance. It is shown that the user needs the control of the actions, and giving feedback to the user actions is crucial. In [30], the guidelines for powerful screen layouts are presented. Aesthetic measures (e.g., symmetry, balance, regularity and consistency) are introduced for the screen designs.

Chapter 3

Web-based Querying Interface

Before explaining our Web-based Querying Interface, the rule-based video database system that we experiment on is briefly explained in the following section.

3.1 The Rule-Based Video Database System

The video database system, for which the web-based querying interface is developed, is a rule-based system where the spatio-temporal relations among salient objects are modelled as rules [10]. This lies in the fact that rules have been extensively used in knowledge representation and reasoning, particularly in artificial intelligence in computer science. Having stored only some necessary facts in the knowledge-base, the system turns out to be very space-efficient since all of the relations can be derived from the stored facts and inference rules. Besides, the rule-based approach employed in the system provides an easy way to process and understand the knowledge-base structure.

In the video database system, Prolog is used for rule processing and querying of spatio-temporal relations. By rules, it is possible to derive some spatial knowledge, which is not explicitly stored within the database. Thus, some basic facts

are stored in the knowledge-base and the generation of the rest of the knowledge is left to Prolog.

The current method for deciding which facts to store as basic facts in the knowledge-base is as follows [12]: we label each object in a frame in raster-scan order by numbers from 1 to n where n is the number of objects in the frame. This labelling starts from the top-left corner of each frame and takes into account the top-left coordinates of the objects' minimum bounding rectangles* (MBRs). Then, having labelled all the objects in the frame, we compute the directional and topological relations for each possible object pair whose first object's label is smaller than that of the second object and whose label distance is one. The *label distance* of an object pair is defined as the absolute numerical difference between the object labels assigned to the objects in the pair. Objects are paired in raster-scan order as well starting with the object labelled by 1. Initially, we start with an empty set of facts. As we compute each directional and topological relation for the pairs of objects, it is checked that whether it is possible to derive the relation from the current set of facts using our inference rules. If a relation cannot be derived from the current set using the rules, we add it to the set of facts. Otherwise, we just ignore it because it can be derived from the facts in the set. After we exhaust all the pairs with the label distance one in raster-scan order, we do the same operation for the pairs of objects whose label distance is two. The process continues in this manner increasing the distance each time by one and finishes when the distance becomes invalid, i.e., incremented to the value n, the number of objects in the frame. As mentioned before, the first object in each object pair has always a smaller label than that of the second object since the objects are paired in raster-scan order. The set of facts produced after this process contains the facts that must be stored in the knowledge-base. The rest of the relations may be derived from these facts using the rules.

In the video database system, video data is processed in the knowledge-base population phase through the keyframes. Since the querying phase is based on this knowledge-base structure, the processing is facilitated with this structure.

*A salient object is approximated with its minimum bounding rectangle such that the sides of the rectangle are oriented along the x and y axes.

The details of the knowledge-base population can be found in Appendix A.

3.2 The Query Specification Interface

In the video database system, the Query Specification Interface (QSI) is designed for the specification of queries over the Internet. In this way, it becomes possible to have a multi-user querying system sharing a video database. In the query processing part of this multi-user environment, the query processor is designed to cope with the queries from clients. Thus, QSI provides a medium for a client to specify the query and display the results of the specified query.

For a user, query specification via visual sketches is a better way for queries because the knowledge-base contains relations (e.g., spatio-temporal, trajectory) among salient objects. Since the relations are deduced according to the minimum bounding rectangles (MBR), it is better to represent salient objects with their MBRs during query specification. In order to facilitate the user's tasks, most of the relations are deduced automatically based on the user's sketches.

Since the query processor is multi-threaded to accept the queries from clients, QSI is designed to work for multiple clients on the Internet. The web client is a Java Applet that is composed of query specification windows for distinctive types of queries. These types of queries are nothing but *spatial* and *trajectory* queries. The specification and formation of these queries vary significantly, hence, specific windows to handle them are created. In the remainder of this section, the query specification windows for spatial and trajectory queries will be discussed. Since the videos have time dimension, these two types of primitive queries can be combined with temporal predicates in order to query the temporal content of the videos.

3.2.1 Spatial Query Specification

Spatial content of a video keyframe, or generally a two-dimensional image, is the relative positioning of the salient objects with respect to each other on the image. This relative positioning consists of three separate sets of relations: directional, topological, and 3D relations. In order to query the spatial content of a video keyframe, these relations have to be provided in the query within a proper combination. Obviously, this proper combination should be performed with logical `and` since all of the relations have to be present in the resultant video frame(s).

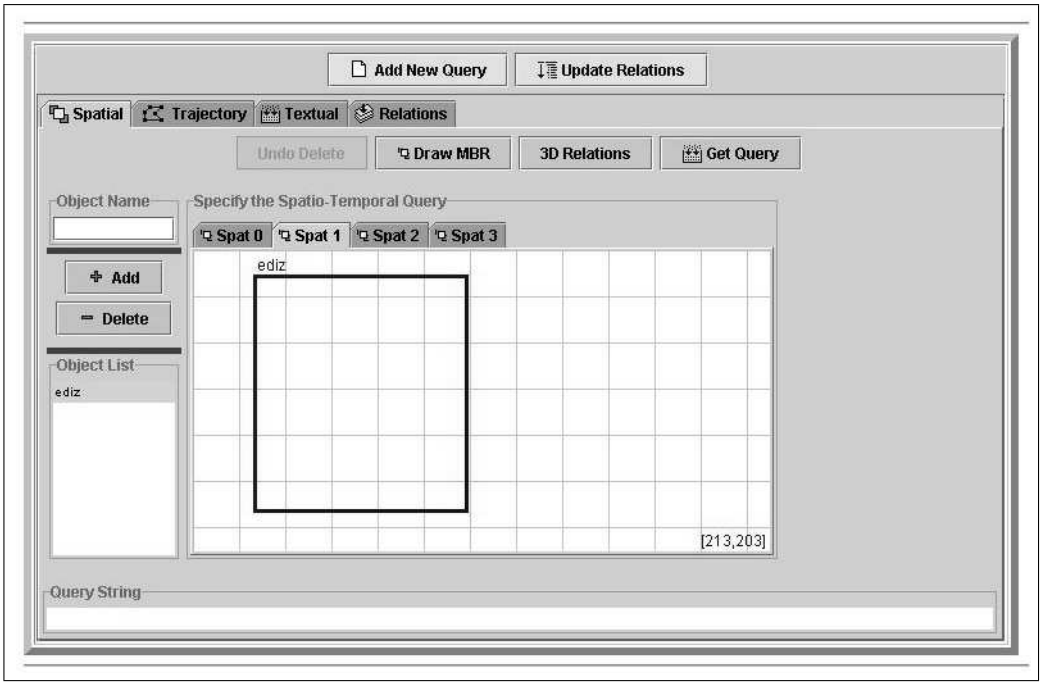


Figure 3.1: Spatial Query Specification Window

In the spatial query specification window shown in Figure 3.1, the user's main task is to draw the salient objects as a rectangle on the sketch. This rectangle represents the minimum-bounding rectangle (MBR) of the salient object. This lies in the fact that each salient object is enclosed within an MBR during the database population phase, and the spatial content of the video keyframe is extracted according to the MBRs. Similar to the database population phase, the

directional and topological relations among the salient objects are deduced automatically in the query specification phase. Since it is impossible to deduce the 3D relations from a 2D data, the users are guided to select the appropriate 3D relations for salient object pairs. Figure 3.2 shows the 3D relation specification window that guides the users.

It is possible that the user may change the locations, sizes and the relative positions of the salient objects during the query specification. The spatial relation deduction process takes place after the final configuration is formalized. Obviously, deleting or hiding a salient object will modify the relation set, and if this modification occurs after the spatial relation deduction process, the relations corresponding to the deleted or hidden object are deleted accordingly from the relation set.

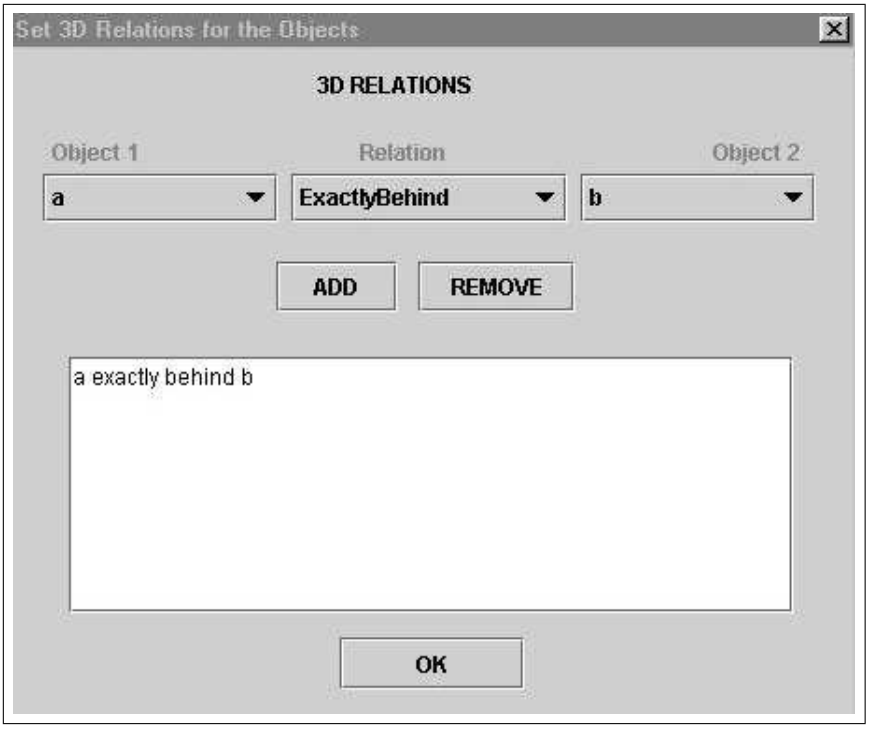
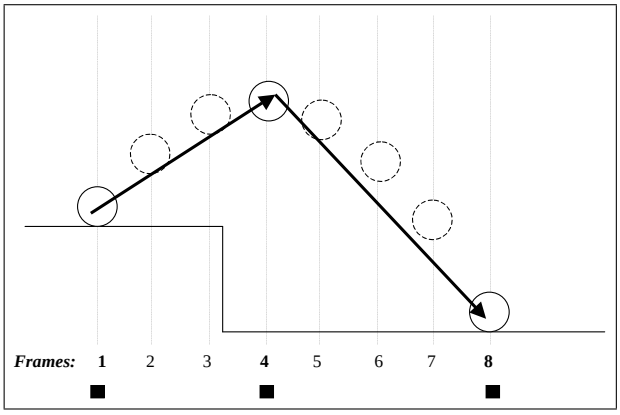


Figure 3.2: 3D Relation Specification Window

3.2.2 Trajectory Query Specification

Trajectory of a salient object is described as a path of vertices corresponding to the locations of the object in different video keyframes. The displacement values and the directions between consecutive keyframes (vertices) are used in defining the *trajectory* fact of an object in the video database system. Figure 3.3 shows a trajectory of a bouncing ball with its corresponding fact. The first item in the fact is the object name, the second is the direction list, third is the displacement value list and the last is the interval list where an interval is set according to two consecutive keyframes. Since the keyframes are 1, 4 and 8 (the ones having black squares below) the positions of the ball at keyframes are considered while forming the trajectory fact of the ball.



`trajectory(`*b*`, [northEast, southEast], [0.7, 1.0], [[1,4], [4,8]]).`

Figure 3.3: Trajectory of a bouncing ball *b* with the corresponding trajectory fact.

In the trajectory query specification window shown in Figure 3.4, the user's main task is to draw the trajectory of the salient objects. The trajectory is a path for the object and each vertex denotes the location of the object for a video keyframe. The distances among the vertices are normalized and the largest one becomes 1. In the trajectory query specification window, the drawn trajectories are dynamic in the sense that any vertex can be deleted or a new vertex is inserted to the trajectory. The locations of the vertices can be changed in order to have a proper trajectory. Figure 3.5 demonstrates the deletion of a vertex

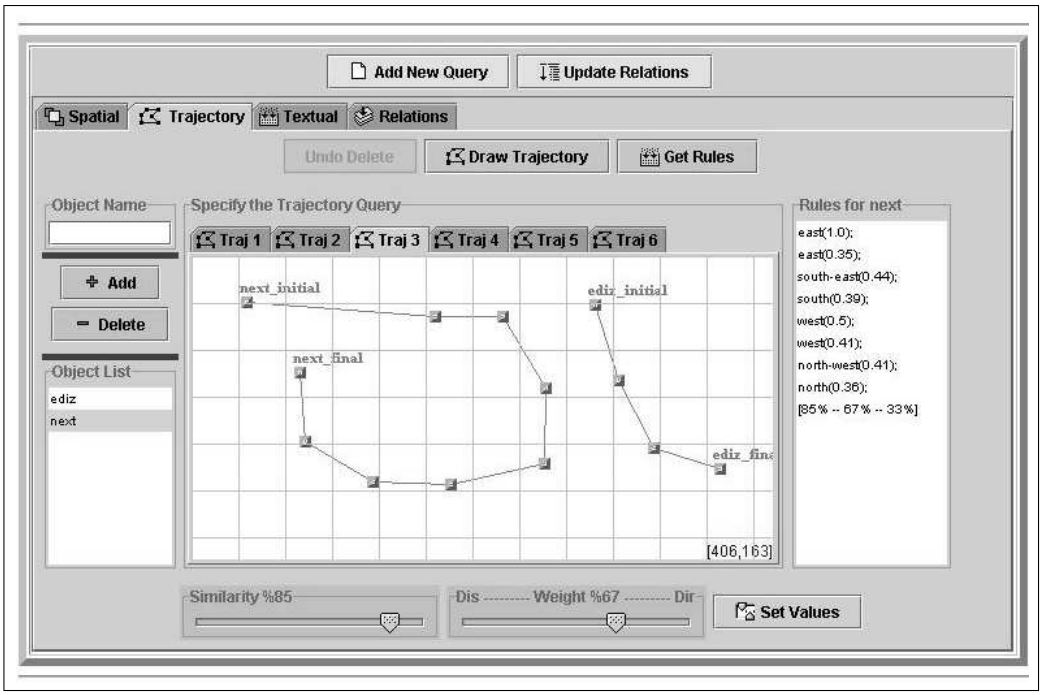
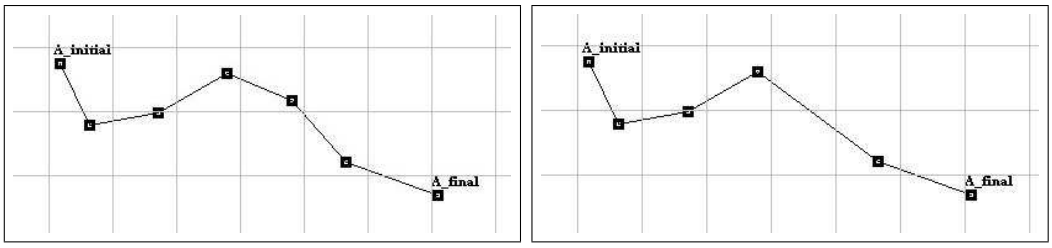


Figure 3.4: Trajectory Query Specification Window

from a trajectory and presents the corresponding rule sequences for the drawn trajectories. It should be noted that the beginning and end of a trajectory is denoted by `objectName_initial` and `objectName_final`.

Retrieval of object trajectories according to the exact match of drawn trajectories is very hard. For this reason, a similarity concept is implemented in the system. The user only specifies a similarity value, between 0 and 100, and does not deal with the actual meaning of the similarity value to the trajectory query. Consequently, the details of the trajectory query processing is out of the scope of the thesis[†]. Since an object trajectory is composed of sets of directions and displacements separately, weights can be assigned to these sets. By default, they have equal weights (i.e., weights are 0.5), however, the user may modify these as far as the weights add up to 1. In order to facilitate both of these specifications, two sliders are placed on the trajectory specification window (see the below part of Fig. 3.4). The first one is for similarity value and its range varies from 0 to

[†]The details of the query processing can be found in [11].



(a) (b)

south,0.69; east,0.71; northEast,0.8; east,0.71; southEast,0.83; east,1.0;

(c)

south,0.45; east,0.47; northEast,0.53; southEast,1.0; east,0.65;

(d)

Figure 3.5: Trajectory example. (a) Trajectory of salient object A. (b) Trajectory of salient object A after deleting the third point from the right. (c) Rule sequence of trajectory in (a). (d) Rule sequence of trajectory in (b).

100 where 100 corresponds to the exact match. The second one is for assigning weights. If the head of the slider is closer to the left end, directions have more importance than displacements and vice versa. Having adjusted any of the sliders, the new values are set by the 'set values' button.

3.2.3 Final Query Formulation

In the user interface, the spatial and trajectory queries are specified from the specific windows. Each of these specifications forms sub-queries and these sub-queries are combined in the final query formulation window. The combination can be not only logical (i.e., `and, or, not`) but also temporal via temporal predicates (see Appendix B.3 for explanation of temporal relations). Then, the final query is sent to the query processor.

The final query formulation window given in Figure 3.6 contains all the specified sub-queries as well as the *appear relations* for each object. The user can combine the sub-queries logically and with temporal predicates. All of the temporal predicates and logical operators (`and, or`), except logical operator `not` are binary operators. At least two sub-queries have to be submitted for each binary

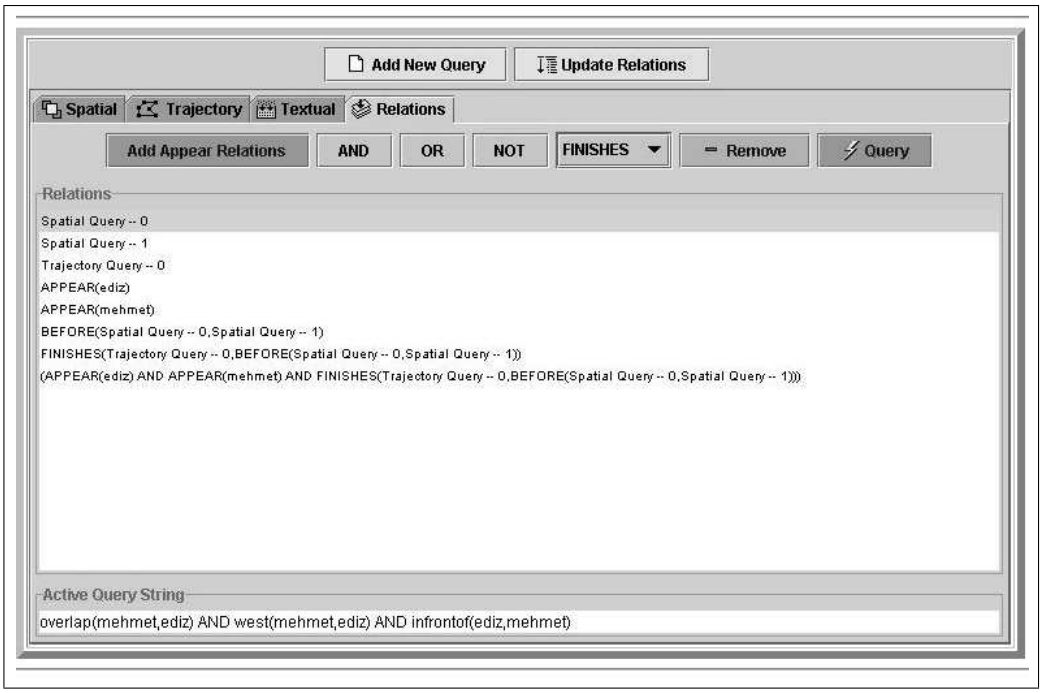


Figure 3.6: Final Query Formulation Window

operator. If more than two sub-queries are submitted, they are combined as consecutive pairs. As soon as the combination is made, the new query is augmented to the list and successive and hierarchical combinations become possible.

Any item corresponding to a sub-query can be sent to the query processor at any time. Having formed the final query in mind, the sub-queries of the final query can be sent in order to get the partial results. By this property, the effects of the sub-queries to the final result are visualized. Moreover, new sub-queries can be combined with the final query after executing the final query in the query processor. Then, the final query becomes a sub-query of the new query and the result becomes a partial result.

Example (*Final Query Formulation*) Let A[‡] and B be two salient objects in a query. First of all, we specify the spatial configuration of the objects for the query in spatial query formulation window. Having drawn the MBRs for the

[‡]The objects' names are written in capital letters in order to clarify the query formulation. In the system, since Prolog is the base for query processing, capital letters denote variables.

objects, assume that the spatial sub-query S_0 is `west(A,B) AND disjoint(A,B) AND infrontof(B,A)`. It should be noted that the last relation is a 3D relation and specified using the 3D relation specification dialog box.

Next step is trajectory specification. Let T_0 and T_2 represent the trajectories of A and T_1 denote that of B. In order not to make the query hard-to-read, their rule sequences are not written in this example.

By the help of 'update relations' button in the final query formulation window, S_0, T_0, T_1 and T_2 are inserted to the relations list as sub-queries. By pressing 'add appear relations' button, the appear relations are inserted, too. At this point the list contains 6 sub-queries since we have 2 salient objects. The reason for adding appear relations separately is based on two reasons: increasing the flexibility while forming the final query and allowing to get the results of the appearance of an object in a video. Any item in the list is a sub-query and its (partial) result can be retrieved.

In order to formulate a more complex sub-query, temporal predicates may be used. For example to express the condition that trajectory of A is before trajectory of B, `before(`T_0 `OR` T_2`,`T_1`)` is inserted to the list. More complex queries can be formulated in a similar hierarchical manner. In this example, the final query is as the following: `appear(A) AND appear(B) AND finishes(before(`T_0 `OR` T_2`,`T_1`),`S_0`)`. □

3.2.4 Discussion

The Query Specification Interface (QSI) is an effective and easy-to-use interface for query specification. The interface is based on the window management strategies, such as the selection of components, which is similar to other windows-based applications. The user actions are handled simply by clicking buttons and the results are displayed in a window (see Figure 3.7).

In [30], a set of general guidelines for screen layouts are presented. Basically, the screens has to obey aesthetic measures such as balance, symmetry, regularity,

unity, etc. The QSI is a collection of windows and each window (i.e., screen) has a balanced and symmetric component layout. As a whole, the interface is consistent and regular. Besides, the guidelines in [28] are taken into account while developing QSI.

Figure 3.7: Result Display Window

Chapter 4

The Object Extractor

Advances in multimedia technology have accelerated the amount of digitized information so as the data stored as image and video content. Both of these data types require application-dependent processing strategies and easy-to-handle storage methods when stored in a database. As far as the querying process of image and video databases is concerned, spatial (for both image and video data) and spatio-temporal (for video data only) as well as semantic information are taken into account. Semantic querying requires more sophisticated techniques that encode the semantic meaning of the image and/or video content. Spatial and spatio-temporal querying necessitate a query pre-processing phase in which the objects and their corresponding features (e.g., color, shape, texture, size) are extracted. The motivation of the tool presented in this chapter is to facilitate the pre-processing phase of the query-by-feature sub-system of the video database querying system that we develop [10].

There exist a considerable number of multimedia data querying systems in the literature [7, 15, 32, 42]. Most of these systems handle only image data and just a few of them deal with both image and video data. The pre-processing phase in most of these systems includes object extraction in order to figure out the hidden object-based information from the image and video data. Based on the type of user interaction in the pre-processing phase, the object extraction methods can be grouped into three categories:

Fully Automatic Extraction Methods: In these methods, the extraction process for image and/or video data is performed automatically. Since the whole process lacks user interaction, not all types of images and video can be handled with this approach. The QBIC system [15] employs a method based on *Foreground/Background* approach for fully automatic object extraction. This method processes only images and video frames having a separable background [3]. Jain and Vailaya employ edge-detection algorithms for extracting object boundaries from images [23]. Their method for edge-detection is the famous *Canny Edge-Detection* algorithm [5]. Chang et al. describe automatic feature extraction methods for color, texture and shape features of images in [6].

Semi-Automatic Extraction Methods: In these methods, the user assists the object extraction process. One way of assistance is to facilitate the object extraction of an image via clicking on the object pixels to visualize the object area. *Flood Fill* algorithm for polygon filling [17] may be adopted to determine the pixels in the object area for the extraction process. Another semi-automatic method is based on the *snakes* concept of computer vision [25]. In this method, the user specifies a bounding polygonal region for an object and the object boundaries are determined from this region automatically. The QBIC system uses these techniques for the object extraction process [3]. The range of images and/or videos handled by these types of methods is larger than that of fully automatic methods but there still exist some types of data that need more user assistance for a proper object extraction.

Manual Extraction Methods: In these methods, the user-computer relation is more than interaction, because the user manages all the extraction process. For example, in order to encapsulate an object region with a (minimum) bounding polygon, the drawings of the users will be used as is. Obviously, any type of image and video data can be handled manually since any kind of data is perceptually comprehensible to the user. However, the manual extraction is a very tedious process and cannot be applied to very large datasets.

The more the user participates in the extraction process, the less the image restrictions and requirements for proper object extraction are needed. The basic aim in object extraction is to minimize the user interaction throughout the extraction process without discarding any type of image or video data. Thus, a powerful extraction system should include tools for each of the above extraction methods not only to extract objects but also to extract the corresponding object features.

4.1 Background Information

One of the most important features of objects in image and video data is *color*. Each pixel in an image has a three-dimensional color vector and different color spaces encode color information based on different approaches. The most famous color space model is the *Red-Green-Blue Model (RGB)* where the color vector of a pixel p is the compound of red, green and blue channels $v_p = (r, g, b)$. Another color space model is the *Hue-Saturation-Value Model (HSV)* that is based on color descriptions rather than individual color components and makes the model unique among other color space models $v_p = (h, s, v)$. The RGB model has a major disadvantage: it is not perceptually uniform. Therefore, most of the systems use color space models other than RGB, such as YIQ [17].

4.1.1 Transformation and Quantization

The color regions are perceptually distinguishable to some extent. The human eye cannot detect some little color differences and may perceive these very similar colors as the same color. This leads to the *quantization* of color, which means that some pre-specified colors will be present in the image and each color is mapped to some of these pre-specified colors. One obvious consequence of this is that each color space may require different levels of quantized colors, which is nothing but a different quantization scheme. In Figure 4.1, the effect of color quantization is illustrated. Figure 4.1 (a) is the original image with RGB color space and (b) is

the image produced after transformation into HSV color space and quantization. A detailed explanation of color space transformations (from RGB into HSV) and quantization can be found in [41].

4.1.2 Color Median Filtering

Not all the colors in the image are dominant. Dominance is in the sense that some of the colors may reside in a region relatively small than the others. The *color median filtering* technique [33], a famous method for neighborhood ranking, eliminates these non-dominant colors and produces a filtered image (Figure 4.1(c)). This technique facilitates the object extraction process because it also eliminates the noise of the color on the object boundaries to some extent.

In order to achieve the best filtering, the color median filter procedure may be applied successively. For most of the types of images, applying color median filtering 3–5 times gives a proper view. Moreover, there exist different types of color median filters. The basic ones are 3×3 box, 5×5 box, 5×5 orthogonal, 7×7 box and 7×7 orthogonal filters.

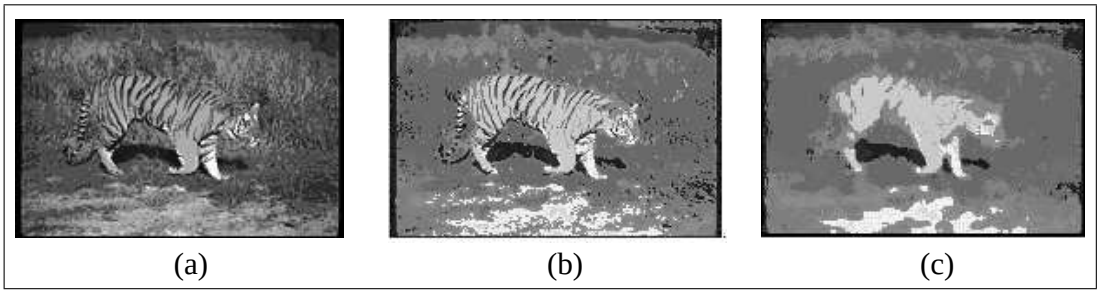


Figure 4.1: Transformation, quantization and color median filtering of an image. (a) Original image. (b) Image produced by applying RGB to HSV color transformation and quantization. (c) Image produced after applying color median filtering.

4.2 The Design of Object Extractor

4.2.1 The Overall Architecture

The *Object Extractor* tool proposed in this chapter extracts objects from images and videos semi-automatically with the help of the improved version of the flood fill algorithm. The overall architecture of the tool is shown in Figure 4.2. Videos are segmented with *Fact Extractor* in the system and keyframes of the videos are processed as images. An image is passed through quantization and color median filtering steps and then the final image, where the object is to be extracted, is produced. This filtered final image is processed with *Flood Fill for Extraction* algorithm to extract the objects along with their features. Since we deal with realistic images and videos in the system, automatic extraction of objects and features would be inadequate, so that the user intervention becomes inevitable. The last step in the process is storing the features of the extracted objects in the object feature database. Thus, *Object Extractor* is one of the basic tools that cooperate with the query-by-feature sub-system of our video database and querying system [10].

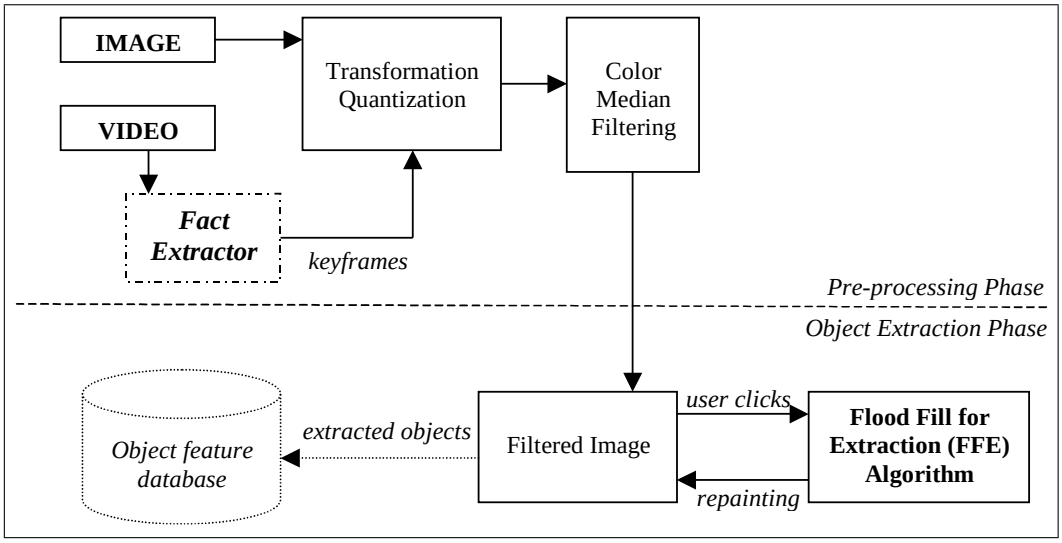


Figure 4.2: The overall architecture of *Object Extractor*

The image whose objects to be extracted passes through a color space conversion step and the color vectors of all of the pixels are transformed from RGB color space into HSV color space. Then, this data is used in the quantization step yielding 18 hue, 3 saturation, 3 value levels and 4 gray levels. The color quantization step employed here is very close to the one proposed in $VisualSEEk$ [42] and $VideoQ$ [7]. After completing this step, the median filtering algorithm is applied to eliminate the non-dominant color regions.

The *Flood Fill for Extraction (FFE)* algorithm, an improved version of the flood fill algorithm, is designed to specify object regions. The color of the user-clicked pixel initiates the process and it forks into four branches corresponding to the four neighboring pixels (north, east, south and west). As soon as the difference between the processed pixel's color and the initiative pixel's color exceeds a pre-specified threshold, the execution stops. The user may continue to specify new initiative pixels as necessary to extract the object.

4.2.2 The User's Assistance

Due to the semi-automatic nature of *Object Extractor*, the user assists the extraction process. The tasks of the user are the identification of the colors in the extracted object and then labeling the object. An appropriate indexing scheme can be adopted for these labeled objects and the indexed data can be queried for the extracted object features such as color and shape. Faloutsos et al. propose an effective querying methodology that is based on quadratic color histogram distance considering the perceptual similarity of colors via cross-correlation [14].

Definition 4.1 *(t-neighborhood) The t-neighborhood of a pixel p with respect to color is a contiguous set of pixels t_p, where the Euclidean distance between color vectors of p and the pixels on the line segment pp_i, such that $p_i \in t_p$, is not greater than a color difference threshold value t. It is obvious that $p \in t_p$.*

In the current implementation of our tool, the user clicks on a pixel p_c on the image and the FFE algorithm is initiated with the pixel p_c and the current

color difference threshold t. During this execution, the pixels in t_p are repainted for p_c. However, if the object bounds unprocessed pixels, the user may click onto another pixel for extracting other parts of the object. Having satisfied with the extracted object region, the user labels the object. The user may specify a different threshold value for each extraction process.

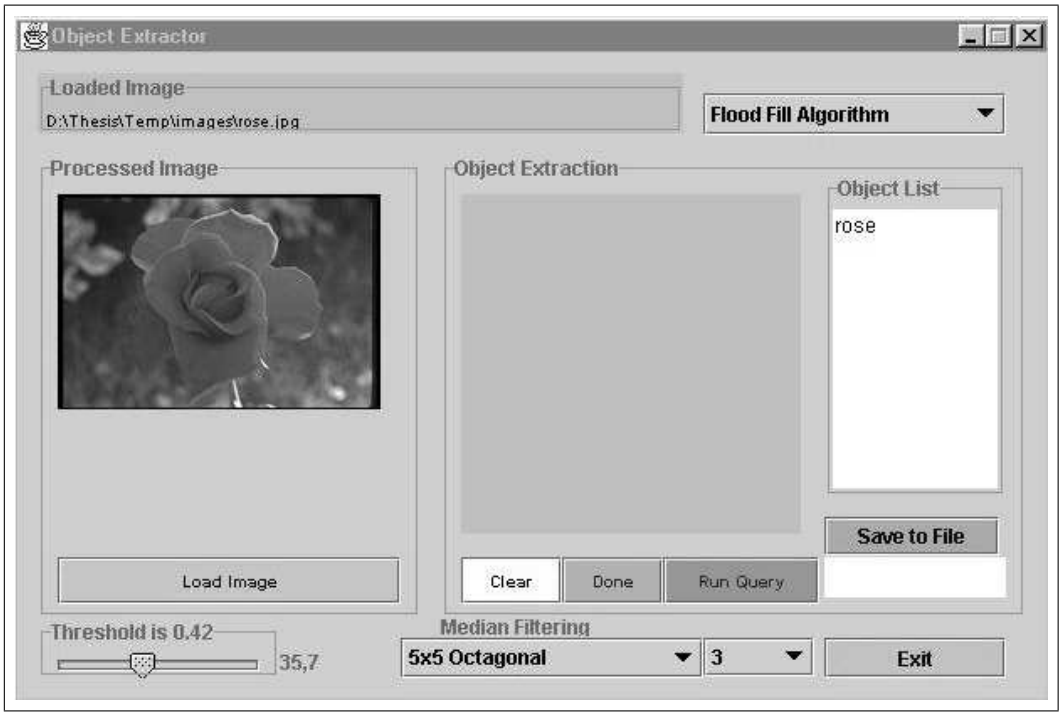


Figure 4.3: The user interface of *Object Extractor*

4.2.3 The User Interface

The user interface of *Object Extractor* has been developed in *Java* programming language and it provides the following functionalities. First of all, since median filtering alleviates color quantization on the image and filters out the non-dominant color regions, the user can see the effects of the median filtering algorithm separately in the tool. This gives the opportunity to the user to decide whether to use median filtering or not. Based on this decision, the color transformation and quantization steps produce a better image. The default color difference threshold

is 0.4, which is determined by a reasonable number of experiments. The user interface of the Object Extractor tool is shown in Figure 4.3.

Moreover, the main usage of this tool is to extract objects for the query-by-feature sub-system of the rule-based video querying system that we develop [10]. As seen in Figure 4.3, 'run query' button activates this querying operation with the previously extracted objects. The image to be queried is also segmented with this tool and the objects are extracted. Since all of the extracted objects are handled with a proper indexing mechanism, the extracted objects of the query image can be queried with the existing objects.

4.3 The Object Extraction Algorithm

The *Object Extractor* tool processes both images and/or video frames. The method it employs for both types of data is very similar since each video frame can be treated as a single image. The *Fact Extractor* tool inside the video database system handles video data and produces video keyframes. Thus, videos can be processed in the *Object Extractor* tool through their keyframes.

```
procedure FloodFillforExtraction(Pixel p)
// INPUT: a single pixel p
// the INITIATIVE_PIXEL is global to the method and
// it holds the user-clicked pixel

1.   if (pixelProcessed(p))
2.       return;
3.   endif
4.   setProcessed(p);
5.   if (thresholdPassed(p, INITIATIVE_PIXEL))
6.       paint(p);
7.       FloodFillforExtraction(left(p));
8.       FloodFillforExtraction(right(p));
9.       FloodFillforExtraction(up(p));
10.      FloodFillforExtraction(down(p));
11.  endif
endprocedure.
```

Figure 4.4: Flood Fill for Extraction (FFE) algorithm

As discussed above, *Flood Fill for Extraction (FFE)* algorithm works with a transformed, quantized and median filtered image. When the algorithm halts, it repaints some of the pixels on the image and the user may continue the extraction as many times as he/she wants. The pseudo-code of the FFE algorithm is given in Figure 4.4. When the user clicks on a pixel in order to initiate the algorithm, this pixel is stored in the $INITIATIVE\_PIXEL$ and used globally in the procedure. Line 1 checks the stopping condition and the lines $4 - 11$ correspond to the recursive part. Each pixel is processed only once due to the `if`-statement at the beginning. Since a pixel may be visited more than once, this fastens the algorithm significantly. On the other hand, the test for the threshold in line 5 is performed by evaluating the Euclidean distance between color vectors of the two pixels, namely p and $INITIATIVE\_PIXEL$. If the test succeeds, the pixel p is repainted and the algorithm calls itself recursively for the neighboring four branches. The whole process stops when there is no executing branch in the recursion tree.

4.4 Experimental Results

The experiments to evaluate the performance of the *Object Extractor* tool are conducted with the images from *Berkeley University Blobworld Project* [4] and *CoffeeCup Software Photo Gallery* [8]. Figure 4.5 gives snapshots of the tool interface during the extraction process of four different images. In all of these experiments, color median filtering is enabled with 5×5 box filters and applied three times to improve smoothing.

The tool gives promising results when the objects are on a separable background in the image. This restriction is inevitable but softened with quantization and color median filtering. This lies in the observation that median filtering facilitates the separation of background from the foreground of the objects. However, the tool performs better in extracting objects containing noise or non-uniformity on the boundaries as well as objects containing holes since it is usually agreed that automatic object extraction tools have difficulty in extracting such objects.

Table 4.1: Objects versus color difference threshold t. X means that the repainted region is larger than the object region when extracted with the threshold.

	Number of Clicks		
Object	t=0.21	t=0.42	t=0.56
Rose	3	1	X
Aircraft	5	2	X
DDeckerBus	7	3	1
AlarmClock	8	2	2
BlackGirl	8	2	X
WorldMap	9	2	X
Tiger	3	1	X

Moreover, most of the objects are extracted with a few clicks in different color regions of an object and the semi-automatic nature of the tool does not have a considerable effect on the speed of the extraction process. In Table 4.1, each row shows the number of mouse clicks during extraction of an object with different threshold values.

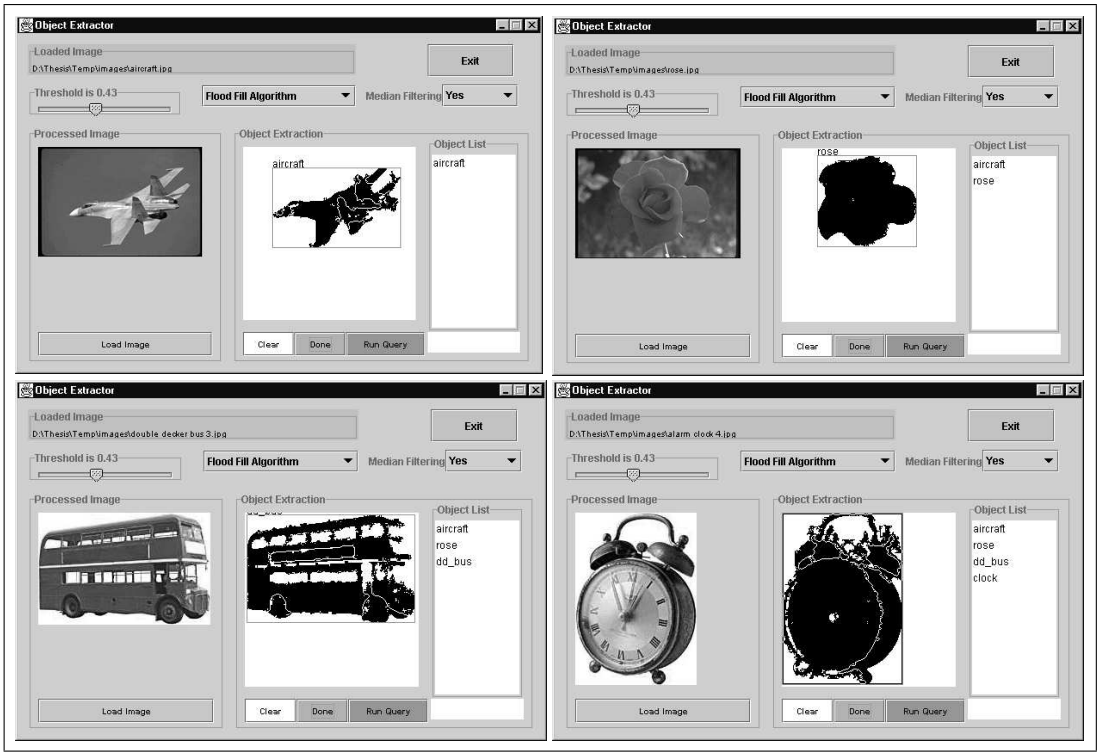


Figure 4.5: Experimental snapshots for four images sampled in *Object Extractor*

Chapter 5

Query-by-Color and Shape Techniques

Most of the existing query specification techniques in multimedia database systems deal with querying of the content of the image and/or video data and the retrieval of the data types according to their content. This type of querying and retrieval is known as *content-based retrieval (CBR)*. Most of the CBR systems include color as the major querying feature. *Color histograms* [15] and *color sets* [7, 42] are some of the methods that are used within these systems. These systems are intended to query the color content of image and video data as a whole. Region-based color queries and queries specifying spatial layout of the color regions are supported to some extent. However, query-by-shape is not supported as frequent as query-by-color. The systems that are capable of handling shape queries use various pattern-matching methods (e.g. moments [45], turning angles [2], Hausdorrf distance [21]) and work with only images. The general shape information of the images can be queried as well as that of objects in these systems. For object-based shape queries, edge-detection is applied to the image and then the polygonal object boundaries are compared. Besides, the texture and size features of the objects can be queried in CBR systems.

In the histogram-based approach presented in this chapter, both image and

video data can be queried. The approach is developed as the core part of the query-by-feature subsystem of our rule-based video querying system [10]. The system handles the spatial relations (in images and videos) and spatio-temporal relations (only in videos) among the objects. The video data is pre-processed with a semi-automatic tool, namely *Fact Extractor* that extracts objects and the relations in video frames. During this process, the keyframes of videos are determined and for each keyframe the shape and color information of the objects are taken into account. Thus, the histogram-based approach treats videos as a collection of images so that each keyframe can be considered as a single image. Another semi-automatic tool, *Object Extractor* (described in the preceding chapter) processes the images and videos –through their keyframes– and extracts the object regions as well as the corresponding object features. Both of the semi-automatic tools are designed to work in parallel for the sake of efficiency.

The contributions of the histogram-based querying approach can be summarized as the following:

Object-Based Query-by-Shape: The method can be applied in a very easy and intuitive way since the shape information of the objects are stored in two histograms. This shape information is gathered from the interior and boundary points of the extracted object. Since we deal with images, these points are the pixels in the extracted object region. For each object region, the center of mass of the object (c_m) forms the basis for shape information of the object. This technique resembles the visualization of the human eye and resolves the drawbacks of the existing methods.

Object-Based Query-by-Color: Since the color information of the object is stored in a color histogram, our querying method is applied for query-by-color as well. All of the histograms for both shape and color information can be filled concurrently, which speeds up the process.

Processed Data Types: As opposed to most of the other methods, our querying method is applicable to not only images but also videos. As discussed above, the extracted keyframes are processed as single images. Thus, the

features of the objects residing in any type of image and/or video data can be queried.

Sensitivity to Noise: The method is sensitive to noise on the boundaries, which other methods suffer from. Since the method gathers information from the pixels, the length and shape of the boundary as well as the presence of holes in the interior have no influence on the method. Besides, there is no polynomial restriction on the object boundaries as in *turning angle* method [2]. Thus, the method is successful for processing noisy objects.

5.1 Related Work

5.1.1 Query-by-Color Approaches

Color is the most frequent feature that is used in multimedia data querying and retrieval systems. The existing CBR systems are enriched with an easy-to-index data structure in order to facilitate the query processing. Specification of a sample query image is widely used in query-by-color (*query-by-example*). The images that are similar enough to the query image are retrieved from the database. Instead of query-by-example, the users may specify the percentages of the color channels (e.g., red, green, blue for RGB color space) in the color space and form a color query. In this case, the images that have closer color percentages to the query color percentages are retrieved.

The *color histograms* [15] are used to represent the color distribution in an image or a video frame. Mainly, the color histogram approach is counting the number of occurrences of each unique color on a sample image. Since an image is composed of pixels and each pixel has a color, the color histogram of an image can be computed easily by visiting every pixel once. By examining the color histogram of an image, the colors existing on the image can be identified with their corresponding areas as number of pixels. In order to store this color information, one possible way is to use three different color histograms for each color channel.

Another way is to have one color histogram for all of the color channels. In the latter approach, the color histogram is simply a compact combination of three histograms and the empty slots can be discarded easily. This histogram approach is commonly used in most of the existing systems supporting query-by-color content. A supplementary information about color for an image is the average color and it may be stored along with the color histogram for the sake of efficiency since it can be computed from the color histogram in one pass.

In [41], *colorsets* are proposed instead of color histograms. The colorsets are binary masks on color histograms and they store the presence of colors as 1 without considering their amounts. For the absent colors, the colorsets store 0 in the corresponding bins. The colorsets reduce the computational complexity of the distance between two images. Besides, by employing colorsets region-based color queries are possible to some extent. On the other hand, processing regions with more than two or three colors is quite complex.

Not only the image as a whole but also the color regions that exist in the image can be queried. Since there may exist more than one region in an image, the spatial relations among the color regions can be queried. Based on this discussion, the color query types can be classified as follows:

Global Color Information

- Find all images that have a color distribution similar to the given sample image (i.e., query-by-example).
- Find all images that have color channels similar to the specified color channel percentages (e.g., `Find images with color percentages 40% Red, 40% Green, 20% Blue`).

Local Color Information

- Find all images that have a region (or a group of regions) with the color distribution similar to the given region (or a group of regions) (e.g., `Find images having a green region and a yellow region`).

Local Color Information with Spatial Relations

- Find all images that have a group of regions with the color distribution similar to the given group of regions, and the group of regions satisfy the specified spatial relation(s) (e.g., `Find images having a green region on the southwest of a yellow region`).

Since the regions may also enclose more than one color, the color channel percentages within regions may be queried as well (e.g., `Find regions having 40% Red, 40% Green, 20% Blue`).

5.1.2 Query-by-Shape Approaches

Shape is an important feature that identifies objects on images. In order to retrieve objects according to their features, color is inadequate in most of the circumstances. Shape strengthens the image retrieval since it encodes the object-based information in a way different than color. Thus, query-by-shape is supported by CBR systems that allow object feature-based queries.

In QBIC system [15] area, circularity, eccentricity, major-axis direction, object moments, tangent angles and perimeter identify the objects in an image database. Specification of these parameters require specific analysis during the database population phase. In [23], the authors propose a method for image-based shape retrieval. Their method necessitates edge-detection on the image and for this purpose, they use the famous edge-detection algorithm proposed by Canny [5]. The identified edges are stored in a histogram for an image. Chang et al. explain a process for gathering object-based shape information including eigenvalue analysis, generation of first and second order moments [7]. In [32], the authors propose a method based on eigenvectors of the object's physical model for shape representation. They use a finite element method [37] to construct the physical model for an object.

Basically, shape retrieval is performed in two different ways. The first way is query-by-example where a sample object is supplied and the objects that have

similar shape structure are returned. The second way is query-by-visual sketch. An object can be specified with a combination of geometric primitives (e.g., rectangle, circle, triangle) and the objects that are close to this sketch are retrieved.

Most of the CBR systems adopt shape comparison and similarity techniques to shape retrieval process. The following section briefly discusses these shape comparison techniques. For a detailed discussion, please refer to [47]. Besides, a taxonomy on shape description techniques is presented in [34].

5.1.3 Histogram Comparison Techniques

In this section, the comparison techniques and metrics for histograms or point-sets are discussed. These comparison techniques are applied to the histograms in our approach for query-by-color and shape in order to determine the similarity between two histograms. In these techniques, the histograms must be normalized to have a proper similarity distance.

Let $H[1..n]$ denote a histogram with n bins enumerated from 1 to n. The normalized histogram $H_N[1..n]$ is defined as follows:

$$H_N[i] = \frac{H[i]}{\sum_i^n H[i]} \tag{5.1}$$

5.1.3.1 Discrete Metric

The discrete metric method defines a distance value between two histograms H_1 and H_2 representing information of two objects as follows:

$$d_D(H_1, H_2) = \left\{ \begin{array}{ll} 0, & if\ H_1 = H_2 \\ 1, & otherwise \end{array} \right\} \tag{5.2}$$

This method is very sharp in the sense that it classifies two point-sets as either similar or not. However, in shape retrieval this is not the case because the partial similarity between objects is needed. Thus, the method is generally inapplicable for shape retrieval.

5.1.3.2 L_p Distance Metric

This method constitutes a class of distance metrics such that if p is 1, L_1 is the *Manhattan* distance, and if p is 2, L_2 is the *Euclidean* distance among the point-sets. The general formulation of the L_p distance between two histograms H_1 and H_2 is as follows:

$$d_{L_p}(H_1, H_2) = (\sum_{i=0}^{n} |H_1[i] - H_2[i]|^p)^{1/p} \tag{5.3}$$

This metric is also called *Minkowski* distance and it requires the inputs to be of the same size. L_2 distance is commonly used within retrieval techniques when L_1 distance is inadequate (e.g., [2]).

5.1.3.3 Quadratic Distance Metric

This technique differs in cross-wise comparisons from the distance metric discussed in Section 5.1.3.2. It is used for calculating color histogram distance in QBIC system [14]. The distance between two histograms H_1 and H_2 is defined as:

$$d_Q(H_1, H_2) = \sum_{i=0}^{n}\sum_{j=0}^{n}(H_1[i] - H_2[i]) \cdot a_{i,j} \cdot (H_1[j] - H_2[j]) \tag{5.4}$$

where $a_{i,j}$ is a coefficient designating the perceptual similarity between histogram bins (e.g., for color histograms, it is based on the perceptual similarity of the corresponding colors). In order to set a value to the coefficient $a_{i,j}$, another coefficient $d_{i,j}$, denoting the distance between i and j that is normalized with respect to the maximum distance, is determined [16]. Thus, assigning $(1 - d_{i,j})$ to $a_{i,j}$ reduces Equation (5.4) to the following:

$$d_Q(H_1, H_2) = -\sum_{i=0}^{n}\sum_{j=0}^{n}(H_1[i] - H_2[i]) \cdot d_{i,j} \cdot (H_1[j] - H_2[j]) \tag{5.5}$$

5.1.3.4 Histogram Intersection

In this technique, two normalized histograms are intersected as a whole. This technique is proposed by Swain and Ballard [43] for color feature. The similarity

between the histograms is a floating point number between 0 and 1. Equivalence is designated with similarity value 1 and the similarity between two histograms reduces when the similarity value converges to 0. Obviously, both of the histograms must be of the same size, otherwise the similarity is undetermined.

Let $H_1[1..n]$ and $H_2[1..n]$ denote two histograms of size n. Let S_{H_1,H_2} denote the similarity value between the histograms H_1 and H_2. Then, the distance between the histograms H_1 and H_2 is given by

$$d_{HI}(H_1, H_2) = \frac{\sum_i^n \min(H_1[i], H_2[i])}{\min(|H_1|, |H_2|)}. \tag{5.6}$$

5.1.3.5 Turning Angle Representation

This method processes two polygonal shapes and produces turning angle representations of the shapes. Then, L_2 distance metric is employed to get a similarity distance from the representations. Formally, the turning angle representation $\Theta_P(s)$ of a polygonal shape P is a collection of turning angles as a function of the arc length s. A turning angle is the counter-clockwise angle between the tangent of P and the x-axis [2]. Figure 5.1 illustrates the turning angle representation of two images initiated from the bottom-left vertex.

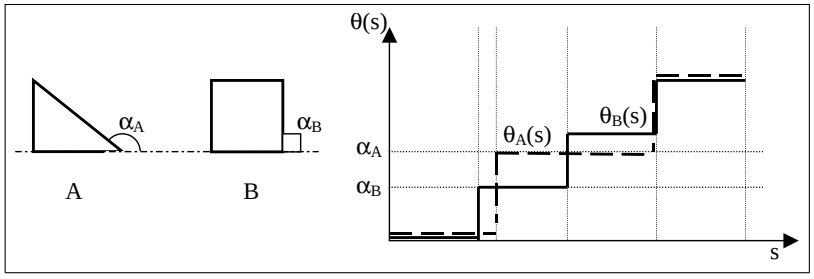


Figure 5.1: Turning Angle Representation

Let $\Theta_A(s)$ and $\Theta_B(s)$ be turning representations for polygonal shapes A and B, respectively. According to the L_p metric, the distance between two polygonal shapes is given as follows:

$$d_T(A,B) = (\int |\Theta_A(s) - \Theta_B(s)|^p \, ds)^{1/p}. \tag{5.7}$$

In order to satisfy invariance under rotation, $\theta_P(s)$ can be shifted with an angular amount θ. The arc length s is scaled to 1 for all of the input shapes, which satisfies scale invariance. The method is also invariant under translation. Thus, it is widely used in shape retrieval systems. In [39], it is argued that this method is superior to the other methods (e.g., moments [45], Hausdorrf distance [21], sign of curvature [39]) due to the human perceptual judgments.

5.2 Histogram-Based Approach for Query-by-Color and Shape

The histogram-based approach is basically motivated with computational geometry concepts and intended to use a similarity measure between images much like the human vision system does. For this motivation, the interrelation among pixels is very important. The main reason is that each pixel provides a piece of information about objects on the image. Since this new approach is applied to the objects extracted by *Object Extractor*, the pixels that do not reside in any object are out of concern.

In this approach, the color information is encoded in HSV color space due to the fact that RGB color space is not perceptually uniform. The three-dimensional RGB color vector is transformed into three-dimensional HSV color vector for each pixel. Then, color quantization and color median filtering techniques are performed on the image. The color information of the filtered image is stored in one color histogram. The histogram-based approach requires two more histograms for shape information. All of the histograms are quantized by considering the human eye perception as it is explained later.

Color Histogram stores the three-dimensional vector $c_i = (h_i, s_i, v_i)$ for all of

the pixels p_i belonging to an object. Since the color information is quantized, color histogram has to store 18 hue, 3 saturation, 3 value levels as well as 4 gray levels.

Distance Histogram stores the Euclidean distance between the center of mass of an object (c_m) and all of the pixels within the object. The distance between a pixel p and c_m accumulates into the corresponding bin in the distance histogram.

Angle Histogram stores the counter-clockwise angle between pixel vectors and the unit vector on the x-axis (e_x). The pixel vector v_p for a pixel p is a vector directed from c_m to p. The unit vector e_x is translated to c_m. As illustrated in Figure 5.2, α is the counter-clockwise angle for p and increments the corresponding bin in the angle histogram.

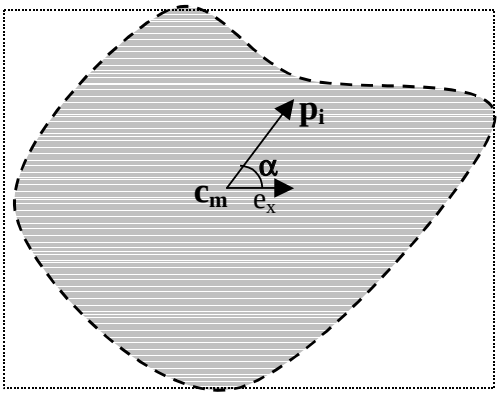


Figure 5.2: Angle Histogram Calculations

The two histograms that store shape information are filled with respect to the center of mass (c_m) of the object. Since the center of mass is a physical property of the object, it is favorable to choose c_m as a reference instead of other points. Obviously, there are more reasons for selecting c_m as a pivot and they can be explained as follows:

- The upper bound of the distance histogram d_u is limited to a reasonable

number due to the fact that the MBR encloses the object with four perpendicular tangents.

$$0 \leq d_u < \sqrt{\max(w^2 + h^2)}, \tag{5.8}$$

where w and h are the width and the height of MBR, respectively. For most of the objects, c_m is not far from the center of the MBR and the upper bound of d_u in Equation (5.8) is very close to $\frac{\sqrt{\max(w^2+h^2)}}{2}$.

- The angle α_i for a pixel p_i is stored in the angle histogram and with a proper quantization scheme, the number of bins in the angle histogram is very close to that of the distance histogram ($0 \leq \alpha_i \leq 2\pi$).

- The location of c_m of an object does not depend on the position and the orientation of the object. Since the coordinates of c_m can be computable in one pass on the pixels of the objects, the color and shape information is correctly referenced by c_m of the object. For an object o, $c_m = (x_{c_m}, y_{c_m})$ can be computed as follows:

$$x_{c_m} = \frac{\sum_i^n x_i}{n},\ y_{c_m} = \frac{\sum_i^n y_i}{n}, \tag{5.9}$$

where n is the number of pixels in o.

The histogram-based approach presented in this chapter takes all the pixels into consideration while gathering color and shape information about the object. All of the pixels belonging to an object participate in the overall information of the object. This way of approaching to the problem is very close to the human eye since the overall information that the eye captured from the image is nothing but the relative positioning and color variance among the pixels. In this method, the relative positioning among pixels encapsulates the shape information in angle and distance histograms. Similarly, the color variance among the pixels is stored in the color histogram.

The angle and distance histograms are quantized and this leads to the fact that relatively smaller sized histograms are enough for the purpose. For example, the angle histogram has 36 bins since the angle quantization value is 10 degrees.

(In Figure 5.3, the $\widehat{c_m}$ angle in R_1 is 10 degrees.) On the other hand, each sampled image is re-scaled before the calculations and it becomes no larger than 200×200 pixels. Thus, the number of bins that the distance histogram stores is no more than that of the color histogram. The quantization values for the histograms are determined by a reasonable number of experiments and evaluations based on the decision to select the best that reflect the human visualization. In Figure 5.4, the histogram-based approach on an object is shown. The object has 5 color regions. The distance histogram is quantized with the value 3, thus the maximum distance between a point p and c_m is 54. Similarly, the angle histogram is quantized with the value 10.

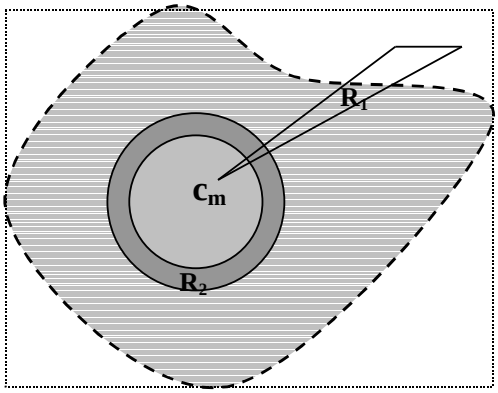


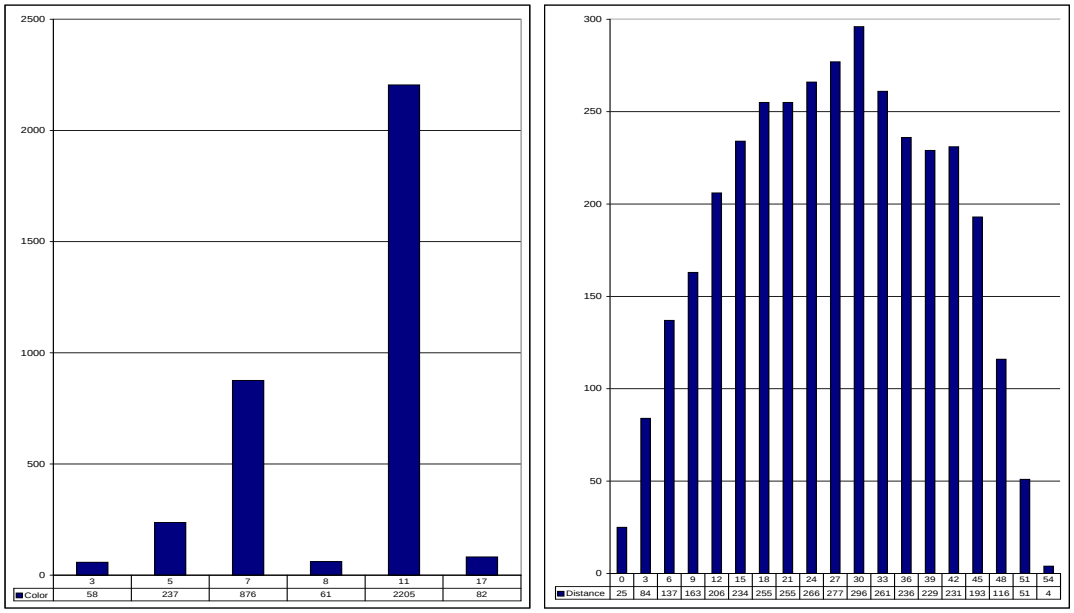
Figure 5.3: Quantization Principles and Equivalent Pixel Classes

Definition 5.1 *(Equivalent Pixel Class) The set of pixels $S_{H[i]}$ forms an equivalent pixel class for an histogram bin $H[i]$ such that for each $p_i \in H[i]$, $p_i \in S_p$.*

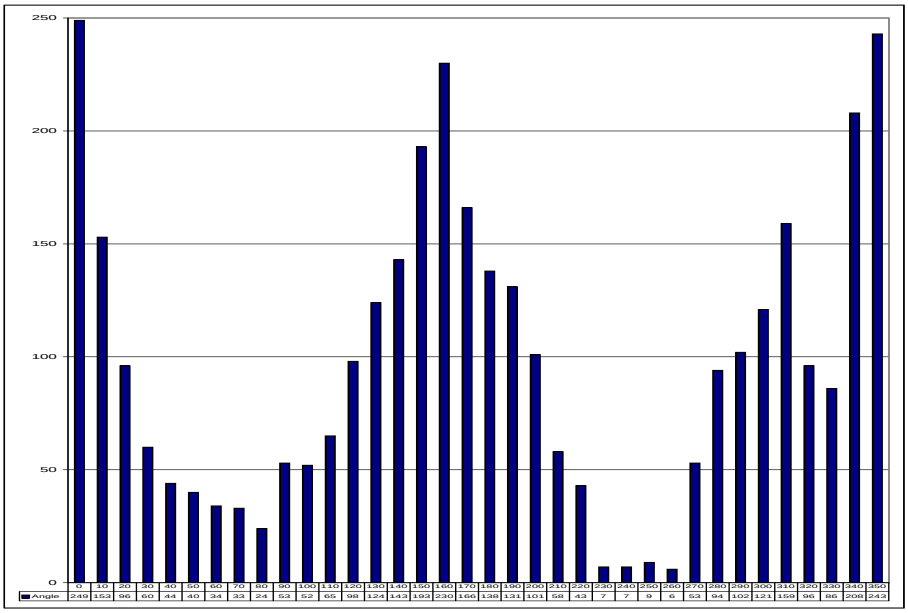
Since we have three histograms, a pixel p_i participates in three different *equivalent pixel classes*. The situation is slightly different for color histograms. The equivalent pixel classes for color histogram have a different meaning such that they are the pixels painted with the same color in the object region. Besides, the equivalent pixel classes for angle and distance histograms form contiguous regions, but color histograms may not. As illustrated in Figure 5.3, the triangular region R_1 is one of the angle histogram equivalent pixel classes. Similarly, the donut-like region R_2 is an equivalent pixel class for the distance histogram.

`Object Details:` $Area = 3519$ pixels, $MBR_{Width} = 103$ pixels, $MBR_{Height} = 66$ pixels

(a)



(b) (c)



(d)

Figure 5.4: Histogram-Based Approach for the `tiger` object in Fig. 4.1. (a)Object details (b)Color histogram (c)Distance histogram (d)Angle histogram.

From the mathematical perspective, the method treats a pixel vector v_p for a pixel p as a five-dimensional vector containing color and shape information as discussed above. The three dimensions of the color information form the first three dimensions of v_p. Since we use two histograms for shape, namely distance and angle histograms, they provide information for the last two dimensions of v_p respectively. Thus, the vector v_p is a five-dimensional vector ($v_p = (h, s, v, d, a)$) in which the first three dimensions represent color information and the last two dimensions represent shape information.

Lemma 5.1 *Shifting the angle histogram of an object one bin left or right specifies a different orientation for the object.*

Proof: Since angles of the pixels are stored in the angle histogram, shifting designates rotating all of the pixels in counter clockwise direction, thus specifies a different orientation. □

Lemma 5.2 *The histogram-based approach for shape is invariant under (i) rotation, (ii) scale, (iii) translation.*

Proof: (i) The color histogram stores color information and color is an orientation-independent property for the object. Thus, the color histogram does not change for different orientations.

The distance histogram stores the *Euclidean* distance between two pixels (p_i and c_m). The locations of both of the pixels vary but the distance between them is preserved for a different orientation of the object.

The angle histogram stores exactly the actual orientation of the object. Due to Lemma 5.1, shifting the angle histogram one bin at a time produces a different orientation. For the similarity distance, all of the orientations have to be considered and this can be achieved via shifting the angle histogram n_a times to the left or right, where n_a is the number of bins in the angle histogram. The similarity distance for angle histogram is the maximum value among these orientations.

(ii) All of the histograms store quantity (i.e., the amount of pixels) thus to satisfy scale invariance, the histograms should be scaled. By the help of the area A_o of the object (i.e., the total number of pixels in the object), it is possible to achieve scale invariance. Every bin in the histograms is divided by A_o and that leads to the scaling of A_o to 1.

(iii) The entities stored in the histograms are location-independent. Thus, the method is translation invariant. □

5.3 Querying Object-Based Color and Shape Information

In this section, the basic algorithms for calculating the similarity value between the histograms of the query object and a database object are summarized. Specifically, histogram-wise comparisons are made by histogram intersection method due to its simplicity and effectiveness. Figure 5.5 presents the pseudo-codes of the similarity algorithms.

The total similarity between the query object and a database object is determined as follows:

$$S_T = \frac{S_C \times w_C + \frac{S_D+S_A}{2} \times w_S}{w_C + w_S} \tag{5.10}$$

where S_C, S_D and S_A denote similarities in color, distance and shape histograms, respectively. The color and shape features are integrated with pre-specified weights w_C and w_S. Since $(w_C + w_S)$ is 1, Equation (5.10) can be written as

$$S_T = S_C \times w_C + \frac{S_D + S_A}{2} \times w_S. \tag{5.11}$$

The execution time for filling the histograms and querying can be estimated as follows: Let $O[n]$ represent an object with n pixels. Since all of the histograms are filled in one pass on the pixels, histogram filling takes $O(n)$ time. Let n_a, n_d and n_c denote the number of bins in the angle, distance and color histograms,

```
Function ColorSimilarity(C_H^q, C_H^d)
1. nominator = 0;
2. qSum = 0;
3. dSum = 0;
4. for i = 1 to length(C_H^q) do
5.     begin
6.         qSum = qSum + C_H^q[i];
7.         dSum = dSum + C_H^d[i];
8.         nominator = nominator + min(C_H^q[i], C_H^d[i]);
9.     end
10. return nominator / min(qSum,dSum);
```

(a)

```
Function DistanceSimilarity(D_H^q, D_H^d)
1. nominator = 0;
2. qSum = 0;
3. dSum = 0;
4. for i = 1 to length(D_H^q) do
5.     begin
6.         qSum = qSum + D_H^q[i];
7.         dSum = dSum + D_H^d[i];
8.         nominator = nominator + min(D_H^q[i], D_H^d[i]);
9.     end
10. return nominator / min(qSum,dSum);
```

(b)

```
Function AngleSimilarity(A_H^q, A_H^d)
1. for j = 1 to length(A_H^q) do
2.     nominator[j] = 0;
3. qSum = 0;
4. dSum = 0;
5. for i = 1 to length(A_H^q) do
6.     begin
7.         qSum = qSum + A_H^q[i];
8.         dSum = dSum + A_H^d[i];
9.         for j = 1 to length(A_H^q) do
10.            nominator[j] = nominator[j] + min(A_H^q[i], A_H^d[j]);
11.    end
12. AngleSimilarity[j] = nominator[j] / min(qSum,dSum);
13. return max_i(AngleSimilarity[i]);
```

(c)

Figure 5.5: Similarity Algorithms. (a) Color (b) Distance (c) Angle.

respectively. Similarly, d_a, d_d and d_c denote the similarity distances for the histograms. In order to compute d_a, the angle histogram has to be shifted n_a times. At each time, histogram intersection method is applied. Thus, d_a is computed in $O(n_a^2)$ time. d_d is computed in $O(n_d)$ time since only one histogram intersection method would be sufficient. The average of d_a and d_d will result in the similarity for shape (S_S). Color histogram requires one histogram intersection method and d_c is computed in $O(n_c)$ time.

5.4 Performance Experiments with Turning Angle Method

In the performance experiments, the histogram-based approach is compared with the well-known shape matching method, the *Turning Angle (TA)* method [2]. In order to be fair and consistent, the best model for converting the output of *Object Extractor* to an input to TA is chosen. Since the histogram-based approach accepts the output of *Object Extractor*, the experiments evaluate the approach accurately. Since the query-by-color approach is almost the same as all the other histogram-based approaches, no evaluation is made for this part.

5.4.1 Kinematics Model for Polygon Simplification

From the Turning Angle (TA) method's point of view, a (polygonal) object is a set of vertices and the shape comparison between two objects is performed based on their turning angle representation. Since objects in images and/or video keyframes are segmented via *Object Extractor* tool (see Chapter 4), a method is required to transform the output of *Object Extractor* to the input of TA method. With this motivation, *Kinematics Model for Polygon Simplification* (KMPS) is implemented.

5.4.1.1 Preliminaries

It is obvious that the extracted objects have at least one boundary point corresponding to every angle with respect to its c_m. This leads to the fact that the output of *Object Extractor* can be treated as a polygon having 360 sides. However, a *360-gon* is quite large for TA method and has to be reduced; i.e., the polygon has to be simplified. An appropriate number for TA method is around 20 and obviously, randomly selecting that number of vertices (points) would not be the correct way. In the KMPS technique, the user provides the number of vertices to be present in the simplified form.

Definition 5.2 *(Boundary Point) Boundary Point is a point which encapsulates x-coordinate, y-coordinate, distance, velocity and acceleration information for itself and is one of the 360 vertices of the given object.*

Definition 5.3 *(Velocity of Boundary Point) The velocity of a boundary point p, V_p, is the rate of change of distance per angle.*

$$V_p = \frac{\Delta d_p}{\Delta a_p} \tag{5.12}$$

In Equation (5.12), d denotes the distance between p and c_m, a denotes the polar angle of p (see Figure 5.2).

Definition 5.4 *(Acceleration of Boundary Point) The acceleration of a boundary point p, A_p, is the rate of change of velocity per angle.*

$$A_p = \frac{\Delta V_p}{\Delta a_p} \tag{5.13}$$

Corollary: Recall that each boundary point in the input falls exactly into one degree of 360 degrees. The velocity of a boundary point p is the difference of the

distances of p and its predecessor point. Similarly, the acceleration of a boundary point is the difference of the velocities of the point and its predecessor point. In both of the cases, the predecessor is surely located in one degree distance from the point p, with respect to c_m. □

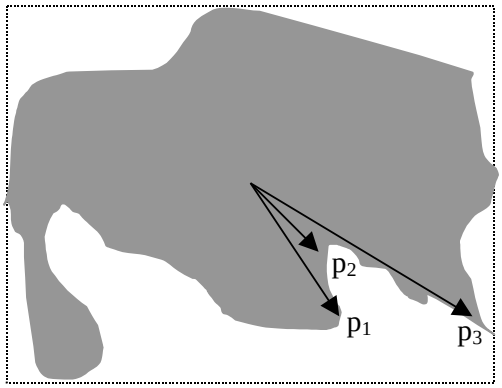


Figure 5.6: Kinematics Model for Polygon Simplification (KMPS) calculations

5.4.1.2 Simplification Method

The more the velocity for a point, the more sharp the object at that point. In Figure 5.6, in order to get a good approximation for the object, the sharp features should be included. In other words, it would be a better choice to include the points p_2 and p_3 in the result set rather than including p_1 and p_2. In order to differentiate between such points, all of the points have to be sorted with respect to their velocities and then a subset of the first M points has to be selected ($N \leq M \leq 360$). The size of this subset (M) can also be specified by the user. After reducing the size of the working set of points to M, the acceleration information of these smaller number of points is used to determine the N points as the final output. The most accelerated points have to be chosen because they are the points where the original object has its sharp features.

In order to visualize the sharpness on objects, Figure 5.7 shows a fluctuating and sharp comb-like figure. This fluctuating and sharp nature of objects can be detected by the change of the acceleration between consecutive points. Returning

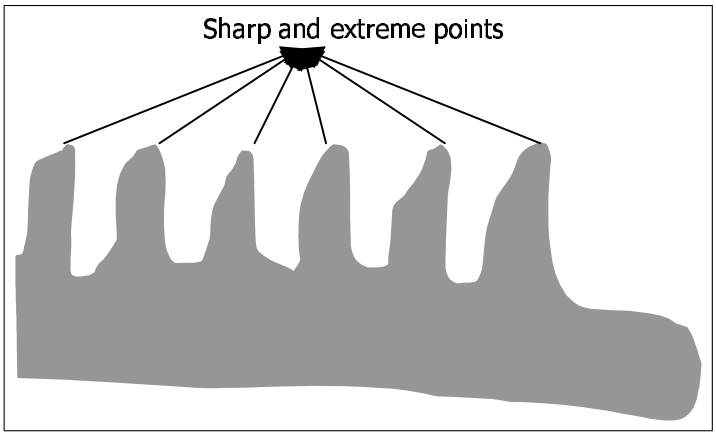


Figure 5.7: Visualizing sharpness on a comb-like figure.

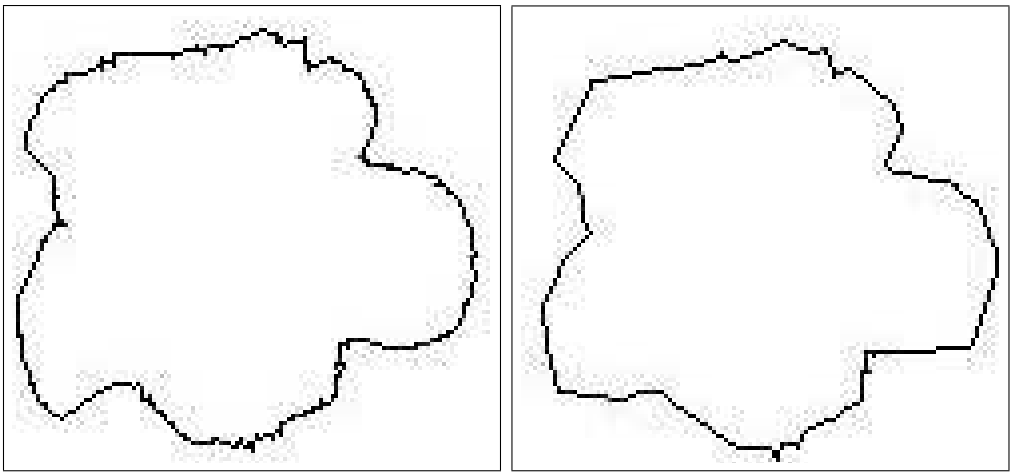
to the comb-like example, if one extreme has a positive acceleration, the other extreme will have a negative one in this sharp region. This causes the acceleration difference between two extremes to be huge. Therefore, among these most accelerated points, the very first N points have to be selected as the final output. These points constitutes the best approximation as an N-gon. Figure 5.8 demonstrates the KMPS method on a sample object. In Figure 5.8 (a), the original object is shown as a polygon of 360 vertices. After applying KMPS method on the object, the object is simplified to 23 vertices as shown in Figure 5.8 (b).

Since the output of object extraction process via *Object Extractor* is certainly an input to this simplification method, KMPS turns out to be the best fit for evaluating the histogram-based approach.

5.4.2 Performance Experiments

In this section, the results of the experiments are presented for a collection of objects. The objects that we experiment on, namely `tiger, double decker bus, alarm clock, rose` and `aircraft`, can be seen in Figures 4.1 and 4.5.

In determining the similarity value between two objects, the trivial but the most comprehensive way is setting 1 to equality and 0 to the least similarity.



(a) (b)

Figure 5.8: Application of KMPS on an input polygon. (a) Original polygon. (b) Polygon after simplification to 23 vertices.

Thus, when the similarity value is closer to 1, the objects resemble each other to some extent. In the histogram-based approach, this way of similarity determination is employed. However in the *turning angle (TA)* method, the way of determining similarity value is different. In the TA method, values less than 0.5 correspond to objects resembling each other. Hence, the similarity values presented in the experiments have to be considered in the light of these discussions.

Since the TA method has drawbacks with 'noisy' objects, the main point in the experiments is to demonstrate the effectiveness of the histogram-based approach with noisy objects. For this reason, the similarity values between two forms of the objects are evaluated. The first form is the original object and the second form is the modified version after discarding relatively small regions from the object region. The modifications on the test objects are as follows:

`Tiger:` Nearly half of the tail is not included.

`Double Decker Bus:` Back wheels are not included.

`Alarm Clock:` Nearly half of a bell on the top is discarded.

`Rose:` Random holes are produced in the center.

`Aircraft:` The bombs at the larger wings are omitted.

Table 5.1: Similarity value results. HBA denotes histogram-based approach and TA denotes turning angle.

Object	HBA	TA
Tiger	0.995	1.484
DDeckerBus	0.982	0.241
AlarmClock	0.974	0.717
Rose	0.978	0.783
Aircraft	0.985	1.262

In Table 5.1, each cell represents the similarity value between the original and modified forms of the object in that row. It is shown through the results that noise on the objects has a crucial impact on the performance of the shape similarity methods. However, the histogram-based approach provides a promising result such that the embedded noise on the object can distort the similarity at most 2.5%. For the TA method, the embedded noise affects the similarity value significantly. In order to have a better understanding for the impact of the noise, the following similarity values gathered from the TA method have to be given. The TA method gives the value 1.30 for the similarity between the objects `tiger` and `rose`. Between `aircraft` and `rose`, the similarity value is 1.05. The last two results are for the original form of the objects. One interesting consequence is that for `aircraft`, the modified version is less similar than `rose` to the object itself. The same argument holds for `tiger`, too. Thus, it can be concluded that the histogram-based approach is more powerful than a well-known -shown to be the best [39]- shape retrieval method, and does overcome its drawbacks.

Chapter 6

Conclusions and Future Work

In this thesis, a web-based query specification interface of a video database system is presented. A novel approach for querying object-based color and shape content is another contribution of the thesis. Since the query-by-color and shape approach requires object-based information, an auxiliary tool for capturing object regions, called *Object Extractor*, is implemented.

The web-based querying interface is logically separated for specifying sub-queries easily. In the interface, spatial and trajectory queries are specified via specific windows. Temporal connectors (e.g., before, meet, starts, etc.) can be used to combine the sub-queries. The results of the queries are displayed on a specific window where each element (i.e., frame sequence) in the result set can be played separately. The separated nature of the interface not only facilitates the query specification but also allows the user to get partial results for the sub-queries. The query specification interface is flexible and effective in the sense that the user can easily formulate any type of, even complex, queries.

The *Object Extractor* tool is a semi-automatic tool used for extracting objects from image and/or video data. In our video database system, the queries that specify object features are processed by the help of this tool. The extracted object features within the extracted object regions are basically the color content and the shape information of the salient objects. The semi-automatic nature of

the tool increases the power and success of the tool in handling various types of images and/or video frames. It is shown through performance results that most of the object regions can be extracted easily with a small amount of assistance.

The query-by-color and shape approach holds histograms for the color and shape content of the objects. The idea is to include the contributions of each pixel in the object region to the color and shape information of the object. Thus, the drawbacks of the existing systems are overcome because most of those systems consider only the boundaries of the objects. One of the contributions of the method is querying object-based information without any pre-processing phase including edge-detection since the output of the *Object Extractor* is used. It is shown through performance results that the approach gives better results in most of the cases than the turning angle method which is known to be one of the best shape comparison techniques in the literature.

The video database system that the user interface is designed for is an ongoing project and the processing of the textual and semantic queries will be included in the near future. Thus, the querying interface would be enhanced accordingly to handle such types of queries. For the time being, only spatial, spatio-temporal and trajectory queries are supported and the querying interface fulfills the requirements of these queries. Moreover, the query-by-feature sub-system would be enriched with other object features (e.g., texture), hence the sub-system would have to be maintained accordingly.

Bibliography

[1] J.F. Allen. Maintaining Knowledge About Temporal Intervals. *Communications of ACM*, 26(11):832–843, 1983.

[2] E. Arkin, P. Chew, D. Huttenlocher, K. Kedem, J. Mitchel. An Efficiently Computable Metric for Comparing Polygonal Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3), 209–215, 1991.

[3] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, D. Petrovic. Automatic and Semi-Automatic Methods for Image Annotation and Retrieval in QBIC. *Proceedings of SPIE-Storage and Retrieval for Image and Video Databases III*, Vol. 2420, 24–35, 1995.

[4] Berkeley University Blobworld Project Start Images, Berkeley University. `http://elib.cs.berkeley.edu/photos/blobworld/start.html`.

[5] J. Canny. A Computational Approach to Edge-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-8, 679–698, November 1986.

[6] S.F. Chang, J.R. Smith, H. Wang. Automatic Feature Extraction and Indexing for Content-Based Visual Query. *Technical Report CU/CTR 414-95-20*, Columbia University, January 1995.

[7] S.F. Chang, W. Chen, H.J. Meng, H. Sundaram, D. Zhong. VideoQ: An Automated Content Based Video Search System Using Visual Cues. *ACM Multimedia'97 Conference Proceedings*, 313–324, Seattle, WA USA, 1997.

[8] CoffeeCup Software Photo Gallery. `http://www.coffeecup.com`.

[9] A. Del Bimbo, E. Vicario, D. Zingoni. Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic. *IEEE Transactions on Knowledge and Data Engineering*, 7(4): 609–621, August 1995.

[10] M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. A Rule-Based Approach to Represent Spatio-Temporal Relation In Video Data. *ADVIS'2000 Proceedings*, LNCS Vol. 1909, T. Yakhno (Ed.), 409–418, 2000, Springer-Verlag.

[11] M.E. Dönderler, Ö. Ulusoy, U. Güdükbay. Rule-Based Spatio-Temporal Query Processing For Video Databases. *submitted for publication*, 2001.

[12] M.E. Dönderler, A Rule-Based Approach to Represent Spatio-Temporal Relations in Video Data, *Ph.D. Progress Report 1*, Department of Computer Engineering, Bilkent University, March 2000.

[13] M. Erwig, M. Schneider. Query-by-Trace: Visual Predicate Specification in Spatio-Temporal Databases. 5^{th} *IFIP Conference on Visual Databases (VDB 5)*, 2000.

[14] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petrovic, R. Barber. Efficient and Effective Querying by Image Content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[15] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petrovic, D. Steele, P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer Magazine*, 28(9), 23–32, September 1995.

[16] J. Hafner, H.S. Sawhney, W. Equitz, M. Flickner, W. Niblack. Efficient Color Histogram Indexing for Quadratic Form Distance Functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, July 1995.

[17] D. Hearn, M.P. Baker. Computer Graphics. Prentice Hall, Inc., Second Edition, C Version, New Jersey 1997.

[18] S. Hibino, E.A. Rundensteiner. A Visual Multimedia Query Language for Temporal Analysis of Video Data. *Multimedia Database Systems: Design*

and Implementation Strategies, K. Nwosu, B. Thuraisingham and P.B. Berra (eds.), 123–159, Norwell, MA: Kluwer Academic Publishers, March 1996.

[19] S. Hibino, E.A. Rundensteiner. MMVIS: Design and Implementation of a Multimedia Visual Information Seeking Environment. *ACM Multimedia'96 Conference Proceedings*, NY: ACM Press, 75–86, 1996.

[20] S. Hibino, E.A. Rundensteiner. User Interface Evaluation of a Direct Manipulation Temporal Visual Query Language. *ACM Multimedia'97 Conference Proceedings*, 99–107, 1997.

[21] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Matching and Machine Analysis*, 15: 850–863, September 1993.

[22] Y. Ishikawa, R. Subrahmanya, C. Faloutsos. MindReader: Querying Databases Through Multiple Examples. *Technical Report CMU-CS-98-119*, Carnegie Mellon University, April 1998.

[23] A.K. Jain, A. Vailaya. Image Retrieval using Color and Shape. *Pattern Recognition*, 29(8): 1233–1244, August 1996.

[24] S. Kaushik, E.A. Rundensteiner. SVIQUEL: A Spatial Visual Query and Exploration Language. *Technical Report CS-TR-9710*, Computer Science Department, Worchester Polytechnic Institute, 1997.

[25] M. Kass, A. Witkin, D. Terzopulos. Snakes: Active Contour Models. *International Journal of Computer Vision*, 321–331, 1998.

[26] W.S. Li, K. S. Candan, K. Hirata, Y. Hara. IFQ: A Visual Query Interface and Query Generator for Object-based Media Retrieval. *In Proceedings of the 1997 IEEE Multimedia Computing and Systems Conference*, 353–361, Ottawa, Ontario, Canada, June 1997.

[27] S. Marcus, V.S. Subrahmanian. Foundations of Multimedia Information Systems. *Journal of the ACM*, 43(3): 474–523, 1996.

[28] L.J. Najjar. Multimedia User Interface Design Guidelines. *IBM Technical Report TR52.0046*, Atlanta, GA: IBM Corporation, September 1992.

[29] C. Nastar, M. Mitschke, C. Meilhac, N. Boujemaa. Surfimage: A Flexible Content-Based Image Retrieval System. *ACM Multimedia'98 Conference Proceedings*, 339–344, Bristol, UK, 1998.

[30] D.C.L. Ngo, L.S. Teo, J.G. Byrne. Formalising Guidelines for the Design of Screen Layouts. *Displays*, 21: 3–15, 2000.

[31] V. Oria, M.T. Özsu, B. Xu, L.I. Cheng, P.J. Iglinski. VisualMOQL: The DISIMA Visual Query Language. *Proceedings of the* 6^{th} *IEEE ICMCS*, Vol. 1, 536–542, Florence, Italy, June 1999.

[32] A. Pentland, R.W. Picard, S. Scarloff. Photobook: Tools for Content-Based Manipulation of Image Databases. *Proc. Storage and Retrieval for Image and Video Databases II*, Vol. 2, 185, 34–47, SPIE, Bellingham, Washington 1994.

[33] J. Russ. The Image Processing Handbook. CRC Press in cooperation with IEEE Press, 1999.

[34] M. Safar, C. Shahabi, X. Sun. Image Retrieval By Shape: A Comparative Study. *IEEE International Conference on Multimedia and Expo (I)*, 141–154, 2000.

[35] S. Santini, R. Jain. Beyond Query by Example. *ACM Multimedia'98 Conference Proceedings*, 345–350, Bristol, UK, 1998.

[36] S. Santini, R. Jain. Integrated Browsing and Querying for Image Databases. *IEEE Multimedia Magazine* (submitted.), 1999.

[37] S. Sclaroff, A. Pentland. A Finite-Element Framework for Correspondance and Matching. *Fourth International Conference on Computer Vision*, 308–313, Berlin 1993.

[38] S. Sclaroff, L. Taycher, M. La Cascia. ImageRover: A Content-Based Image Browser for the World Wide Web. *Proc. IEEE Workshop on Content-based Access of Image and Video Libraries*, June 1997.

[39] B. Scassellati, S. Alexopoulos, M. Flickner. Retrieving Images by 2D Shape: A Comparison of Computation Methods with Human Perceptual Judgments. *In Proc. SPIE Conference on Storage and Retrieval of Image and Video Databases II*, February 1994.

[40] J.R. Smith. VideoZoom Spatio-temporal Video Browser. *IEEE Trans. Multimedia*, 1(2): 157–171, June 1999.

[41] J.R. Smith, S.F. Chang. Tools and Techniques for Color Image Retrieval. *IS&T/SPIE Proceedings*, Vol 2670, In: Sethi, I.K., Jain, R.C., eds., *Storage&Retrieval for Image and Video Databases IV*, 426–437, February 1996.

[42] J.R. Smith, S.F. Chang. VisualSEEk: A Fully Automated Content-Based Image Query System. *ACM Multimedia'96 Conference Proceedings*, 87–98, NY 1996.

[43] M.J. Swain, D.H. Ballard. Color Indexing. *International Journal of Computer Vision*, 7(1): 11–32, 1991.

[44] E. Şaykol, U. Güdükbay, Ö. Ulusoy. A Semi-Automatic Object Extraction Tool for Querying in Multimedia Databases. *to appear in Workshop on Multimedia Information Systems (MIS'01)*, 2001.

[45] G. Taubin, D.B. Cooper. Object Recognition Based on Moment (or Algebraic) Invariants. *In Geometric Invariance in Computer Vision*, J.Mundy and A.Zisserman, (eds.), MIT Press, 1992.

[46] R.C. Veltkamp, M. Tanase. Content-Based Image Retrieval Systems: A Survey. *Technical Report UU-CS-2000-34*, Utrect University, The Netherlands, October 2000.

[47] R.C. Veltkamp. Shape Matching: Similarity Measures and Algorithms. *Technical Report UU-CS-2001-03*, Utrect University, The Netherlands, January 2001.

[48] A. Yoshitaka, T. Ichikawa. A Survey on Content-Based Retrieval for Multimedia Databases. *IEEE Transactions on Knowledge and Data Engineering*, 11(1): 81–93, 1999.

[49] A. Yoshitaka, Y. Hosoda, M. Hirakawa, T. Ichikawa. VIOLONE: Video Retrieval by Motion Example. *Journal of Visual Languages and Computing*, 7(4): 423–443, 1996.

[50] M. M. Zloof, Query-by-Example: A Database Language. *IBM Syst. Journal*, 16(4), 1977.

Appendix A

Facts and Rules in Video Database System

In our video database system, we store the facts in terms of four directional relations, *west*, *south*, *south-west* and *north-west*, and six topological relations, *cover*, *equal*, *inside*, *disjoint*, *touch* and *overlap*, because our inference rules are designed to work on these types of explicitly stored facts. However, we also have rules for *east*, *north*, *north-east*, *south-east*, *right*, *left*, *below*, *above*, *contains* and *covered-by*. These rules do not work directly with the stored facts, but rather they are used to invoke related rules with a proper ordering of objects. For example, let's suppose that we have computed a relation for the pair of objects $\sigma(1, 2)$, such as *east*(1, 2, [1, 1]), where 1 and 2 are object labels and [1, 1] is the interval of the relation, then we check to see if the relation can be derived from the current set of facts using the rules. The rule *east* is used to call the rule *west* with the order of objects switched. That is, we check to see now if *west*(2, 1, [1, 1]) can be derived from the current set of facts using our rules. If this relation cannot be derived from the current set of facts using the rules, we store *west*(2, 1, [1, 1]) as a fact in our knowledge-base, which is equivalent to *east*(1, 2, [1, 1]). For the directional relations, we only consider the fundamental ones in the process of deciding what facts to store in our knowledge-base and leave out the other directional relations *right*, *left*, *above* and *below* because they are simply based on the fundamental

directional relations. By storing some facts in terms of some others equivalent to them in the knowledge-base, we reduce the complexity and the number of our rules in the system.

Our approach greatly reduces the number of relations to be stored in the knowledge-base, which also depends on some other factors as well, such as the number of salient objects, the frequency of change in spatial relations and the relative spatial locations of the objects with respect to each other. Nevertheless, we do not claim that the set of relations we store in our knowledge-base is the minimum set of facts that must be stored because the number of facts to be stored depends on the labelling order of objects in our method and we use the raster-scan order to reduce this number. The heuristic in our algorithm is that if we start with the pairs of objects whose label distance is smaller, we may not need to store most of the relations for the pairs of objects whose label distance is larger because these relations might be derived from the facts already stored in the knowledge-base.

In our system, we define three types of inference rules, *strict directional*, *strict topological* and *heterogeneous directional and topological*, with respect to the relations' types in the rule body. For example, *directional rules* have only directional relations in their body whilst *heterogeneous rules* incorporate rules from both types. We describe our *strict directional rules* and *strict topological rules* in Sections A.1 and A.2, respectively. Our *heterogeneous topological and directional rules* are given in Section A.3. Furthermore, we also provide some examples along with the rule definitions to clarify them. In defining the rules, we have adopted the following terminology: if relation r_1 implies relation r_2, we show it by $r_1 \Longrightarrow r_2$. Moreover, if $r_1 \Longrightarrow r_2$ and $r_2 \Longrightarrow r_1$, we denote it by $r_1 \Longleftrightarrow r_2$.

A.1 Strict Directional Rules

Rule 1 (Inverse Property) The directional relations *west*, *north*, *north-west*, *north-east*, *right* and *above* are inverses of *east*, *south*, *south-east*, *south-west*,

left and *below*, respectively.

$west(A,B) \Longleftrightarrow east(B,A)$ $\quad$ $north(A,B) \Longleftrightarrow south(B,A)$
$right(A,B) \Longleftrightarrow left(B,A)$ $\quad$ $north\text{-}east(A,B) \Longleftrightarrow south\text{-}west(B,A)$
$below(A,B) \Longleftrightarrow above(B,A)$ $\quad$ $north\text{-}west(A,B) \Longleftrightarrow south\text{-}east(B,A)$

Rule 2 (Transitivity) If $\beta \in S$, where S is the set of directional relations, then $\beta(A,B) \wedge \beta(B,C) \Longrightarrow \beta(A,C)$.

Figure A.1 shows an example of Rule 2.

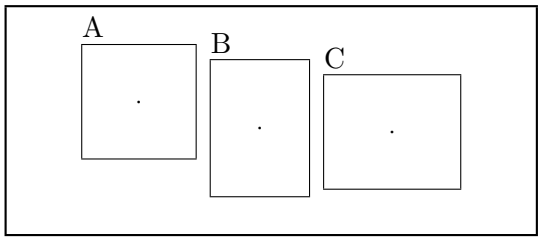


Figure A.1: west(A,B) $\wedge$ west(B,C) $\Longrightarrow$ west(A,C) (Transitivity)

A.2 Strict Topological Rules

The *strict topological rules* can be formally described as follows:

Rule 1 (Reflexivity) The topological relations *equal*, *overlap*, and *cover* are reflexive.

$equal(A,A)$ $\quad$ $overlap(A,A)$ $\quad$ $cover(A,A)$

Rule 2 (Symmetry) The topological relations *equal*, *overlap*, *disjoint* and *touch* are symmetric.

$equal(A,B) \Longleftrightarrow equal(B,A)$ $\quad$ $overlap(A,B) \Longleftrightarrow overlap(B,A)$
$disjoint(A,B) \Longleftrightarrow disjoint(B,A)$ $\quad$ $touch(A,B) \Longleftrightarrow touch(B,A)$

Rule 3 (Transitivity) The topological relations *inside* and *equal* are transitive.

$$inside(A,B) \wedge inside(B,C) \Longrightarrow inside(A,C)$$
$$equal(A,B) \wedge equal(B,C) \Longrightarrow equal(A,C)$$

Rule 4 The topological relations *inside*, *equal* and *cover* imply the relation *overlap*.

$$inside(A,B) \Longrightarrow overlap(A,B)$$
$$equal(A,B) \Longrightarrow overlap(A,B)$$
$$cover(A,B) \Longrightarrow overlap(A,B)$$

Rule 5 The relationships between *equal* and {*cover*, *inside*, *disjoint*, *touch*, *overlap*} are as follows:

a) $equal(A,B) \wedge cover(A,C) \Longrightarrow cover(B,C)$
b) $equal(A,B) \wedge cover(C,A) \Longrightarrow cover(C,B)$
c) $equal(A,B) \wedge equal(C,D) \wedge cover(C,A) \Longrightarrow cover(D,B)$
d) $equal(A,B) \wedge inside(A,C) \Longrightarrow inside(B,C)$
e) $equal(A,B) \wedge inside(C,A) \Longrightarrow inside(C,B)$
f) $equal(A,B) \wedge inside(D,C) \wedge inside(C,A) \Longrightarrow inside(D,B)$
g) $equal(A,B) \wedge disjoint(A,C) \Longrightarrow disjoint(B,C)$
h) $equal(A,B) \wedge overlap(A,C) \Longrightarrow overlap(B,C)$
i) $equal(A,B) \wedge touch(A,C) \Longrightarrow touch(B,C)$

Rule 6 The relationships between *disjoint* and {*inside*, *touch*} are as follows:

a) $inside(A,B) \wedge disjoint(B,C) \Longrightarrow disjoint(A,C)$
b) $inside(A,B) \wedge touch(C,A) \Longrightarrow disjoint(C,B)$

Rule 7 The relationships between *overlap* and {*inside*, *cover*} are as follows (excluding those given by Rule 4):

a) *inside*(B,A) $\wedge$ *overlap*(B,C) $\Longrightarrow$ *overlap*(A,C)
b) *cover*(A,B) $\wedge$ *overlap*(B,C) $\Longrightarrow$ *overlap*(A,C)

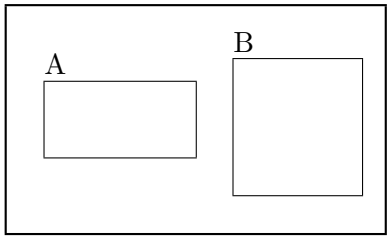
Examples of our strict topological rules are given in Figure A.2.

A.3 Heterogeneous Topological and Directional Rules

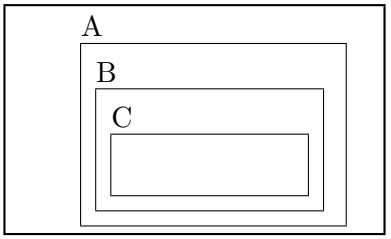
Rule 1 If $\beta \in$ S, where S is the set of directional relations, then *equal*(A,B) $\wedge$ β(A,C) $\Longrightarrow$ β(B,C).

Rule 2 If $\beta \in$ S, where S is the set of directional relations, then *disjoint*(A,B) $\wedge$ *disjoint*(B,C) $\wedge$ β(A,B) $\wedge$ β(B,C) $\Longrightarrow$ *disjoint*(A,C).

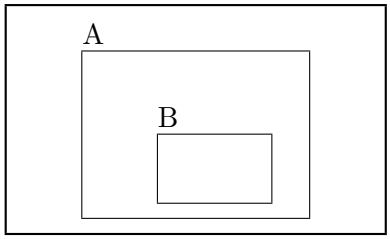
Examples of our heterogeneous rules are given in Figure A.4.



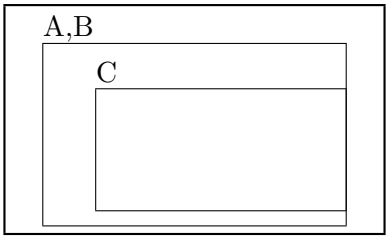
Rule 1: disjoint(A,B), disjoint(B,A)



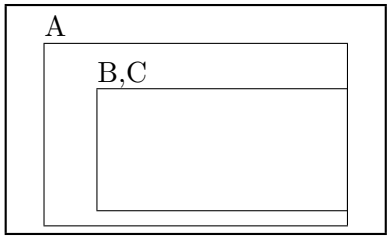
Rule 2: inside(C,B) $\wedge$ inside(B,A) $\Longrightarrow$ inside(C,A)



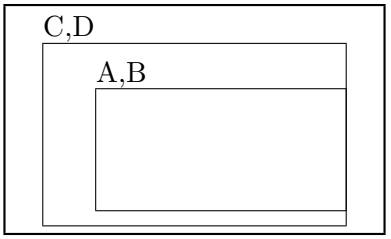
Rule 3: inside(A,B) $\Longrightarrow$ overlap(A,B)



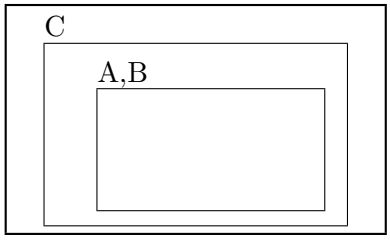
Rule 4(a): equal(A,B) $\wedge$ cover(A,C) $\Longrightarrow$ cover(A,B)



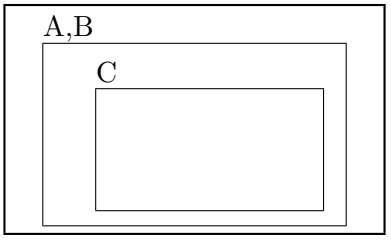
Rule 4(b): equal(A,B) $\wedge$ cover(C,A) $\Longrightarrow$ cover(C,B)



Rule 4(c): equal(A,B) $\wedge$ equal(C,D) $\wedge$ cover(C,A) $\Longrightarrow$ cover(D,B)

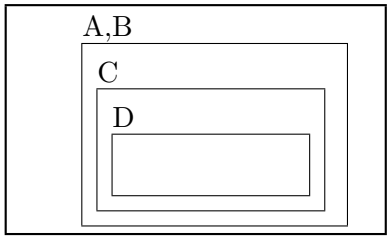


Rule 4(d): equal(A,B) $\wedge$ inside(A,C) $\Longrightarrow$ inside(B,C)

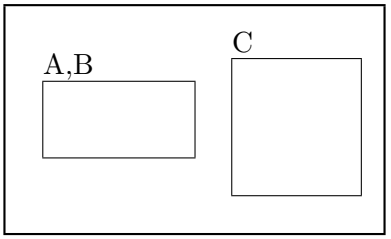


Rule 4(e): equal(A,B) $\wedge$ inside(C,A) $\Longrightarrow$ inside(C,B)

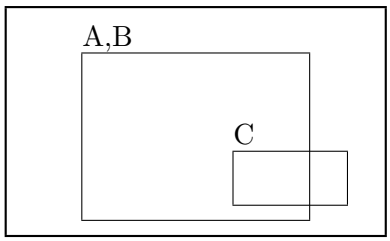
Figure A.2: Strict Topological Rules



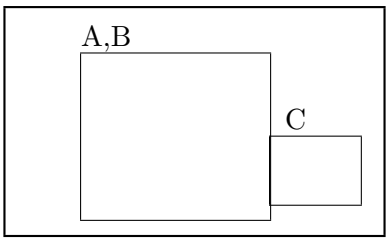
Rule 4(f): equal(A,B) $\wedge$ inside(D,C) $\wedge$ inside(C,A) $\Longrightarrow$ inside(D,B)



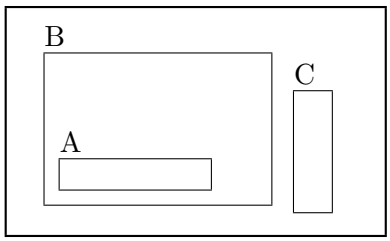
Rule 4(g): equal(A,B) $\wedge$ disjoint(A,C) $\Longrightarrow$ disjoint(B,C)



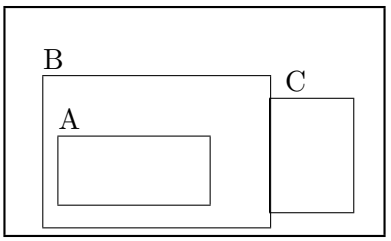
Rule 4(h): equal(A,B) $\wedge$ *overlap*(A,C) $\Longrightarrow$ overlap(B,C)



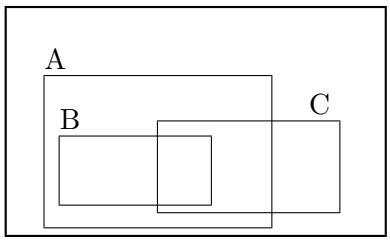
Rule 4(i): equal(A,B) $\wedge$ touch(A,C) $\Longrightarrow$ touch(B,C)



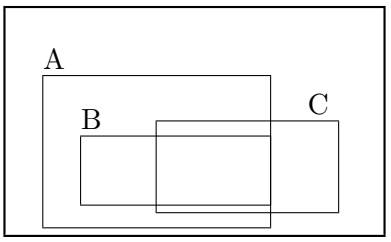
Rule 5(a): inside(A,B) $\wedge$ disjoint(B,C) $\Longrightarrow$ disjoint(A,C)



Rule 5(b): inside(A,B) $\wedge$ touch(B,C) $\Longrightarrow$ disjoint(A,C)

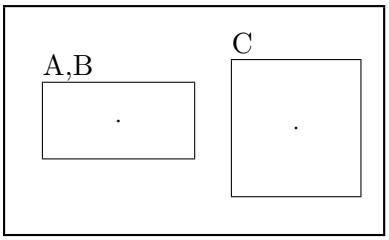


Rule 6(a): inside(B,A) $\wedge$ overlap(B,C) $\Longrightarrow$ overlap(A,C)

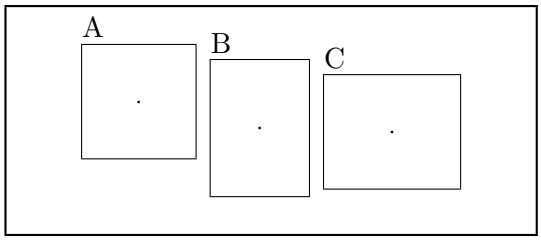


Rule 6(b): cover(A,B) $\wedge$ overlap(B,C) $\Longrightarrow$ overlap(A,C)

Figure A.3: Strict Topological Rules



Rule 1: equal(A,B) $\wedge$ west(A,C) $\Longrightarrow$ west(A,C)



Rule 2: disjoint(A,B) $\wedge$ disjoint(B,C) $\wedge$ west(A,B) $\wedge$ west(B,C) $\Longrightarrow$ disjoint(A,C)

Figure A.4: Heterogeneous Rules

Appendix B

Spatio-temporal Relationships

The ability to manage spatio-temporal relationships is one of the most important features of the multimedia database systems. Many applications depend on spatial relationships between multimedia objects and there has been significant amount of research on spatial relationships in image databases as well as geographical information systems (GIS). In multimedia databases, spatial relationships are used to support content-based retrieval of multimedia data, which is one of the most important differences in terms of querying between multimedia and traditional databases. Spatial relations can be grouped into mainly three categories: topological relations, which describe neighborhood and incidence, directional relations, which describe order in space, and distance relations that describe range between objects. There are eight distinct topological relations: *disjoint*, *touch*, *inside*, *contains*, *overlap*, *covers*, *covered-by* and *equal*. The fundamental directional relations are *north*, *south*, *east*, *west*, *north-east*, *north-west*, *south-east* and *south-west*, and the distance relations include *far* and *near*. We also include the relations *left*, *right*, *below* and *above* in the group of directional relations; nonetheless, the first two are equivalent to the relations *west* and *east*, and the other two can be defined in terms of the directional relations as follows:

Above The relation *above*(A,B) is the disjunction of the directional relations *north*(A,B), *north-west*(A,B) and *north-east*(A,B).

Below The relation *below*(A,B) is the disjunction of the directional relations *south*(A,B), *south-west*(A,B) and *south-east*(A,B).

Currently, we only deal with the topological and directional relations and leave out the distance relations to be incorporated in future. We give the formal definitions for the fundamental directional relations in Section B.1. The definitions for the topological relations are given in Section B.2 using Allen's temporal algebra [1]. Finally, in Section B.3, we explain our approach to incorporate the time component into our knowledge-base to facilitate spatio-temporal and temporal querying of video data, based on Allen's temporal algebra [1].

B.1 Directional Relations

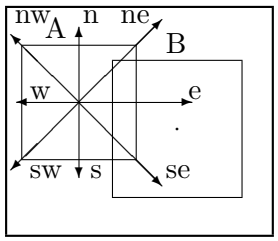
To determine which directional relation holds between two salient objects, we consider the center points of the objects' minimum bounding rectangles (MBRs). Obviously, if the center points of the objects' MBRs are the same, then there is no directional relation between the two objects. Otherwise, we choose the most intuitive directional relation with respect to the closeness of the line segment between the center points of the objects' MBRs to the eight directional line segments. To do this, we place the origin of the directional system at the center of the MBR of the object for which to define the relation as in Figure B.1(a). In this example, object A is to the *west* of object B because the center of object B's MBR is closer to the directional line segment *east* than the one for *south-east*. Moreover, the two objects overlap with each other, but we can still define a directional relation between them as it is seen in Figure B.1(a). Our approach to find the directional relations between two salient objects can be formally expressed as in Definitions B.1 and B.2.

Definition B.1 *The directional relation β(A,B) is defined to be in the opposite direction to the directional line segment which originates from the center of object A's MBR, and is the closest to the center of object B's MBR.*

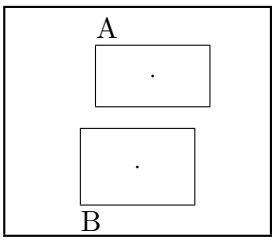
Definition B.2 *The inverse of a directional relation $\beta(A,B)$, $\beta^{-1}(B,A)$, is the directional relation defined in the opposite direction.*

According to Definition B.1, given two objects A and B, if the center of object B's MBR is closer to the directional line segment *east* in comparison to the others, when the directional system's origin is at the center of object A's MBR, then the directional relation between objects A and B, where object A is the one for which to define the relation, is *west*(A, B). Thus, we can say that object A is to the *west* of object B. Using Description B.2, it can be concluded that object B is to the *east* of object A.

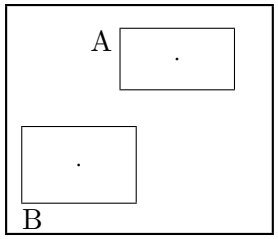
The rest of the directional relations can be determined in the same way. Figure B.1 shows examples of the eight fundamental directional relations.



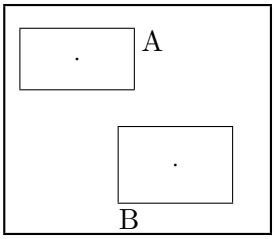
(a) west(A,B), east(B,A)



(b) north(A,B), south(B,A)



(c) north-east(A,B), south-west(B,A)



(d) north-west(A,B), south-east(B,A)

Figure B.1: Directional Relations

B.2 Topological Relations

The topological relations *inside* and *contains* are inverses of each other, and so are *cover* and *covered-by*. In addition, the relations *equal*, *touch*, *disjoint* and

Relation	Symbol	Inverse	Condition
$[s_1, e_1]$ before $[s_2, e_2]$	**b**	**bi**	$e_1 < s_2$
$[s_1, e_1]$ meets $[s_2, e_2]$	**m**	**mi**	$e_1 = s_2 - 1$
$[s_1, e_1]$ during $[s_2, e_2]$	**d**	**di**	$s_1 > s_2 \wedge e_1 < e_2$
$[s_1, e_1]$ overlaps $[s_2, e_2]$	**o**	**oi**	$s_1 < s_2 \wedge s_2 < e_1$
$[s_1, e_1]$ starts $[s_2, e_2]$	**s**	**si**	$s_1 = s_2 \wedge e_1 < e_2$
$[s_1, e_1]$ finishes $[s_2, e_2]$	**f**	**fi**	$s_1 > s_2 \wedge e_1 = e_2$
$[s_1, e_1]$ equal $[s_2, e_2]$	**e**	**e**	$s_1 = s_2 \wedge e_1 = e_2$

Table B.1: Allen's Interval Relations

overlap hold in both directions. In other words, if β(A,B) holds where β is one of these four relations, then β(B,A) holds too.

The topological relations are distinct from each other; however, the relations *inside* and *cover* imply the same topological relation *overlap* to hold between the two objects in both directions:

- *inside*(A,B) $\Longrightarrow$ *overlap*(A,B) and *overlap*(B,A)
- *cover*(A,B) $\Longrightarrow$ *overlap*(A,B) and *overlap*(B,A)

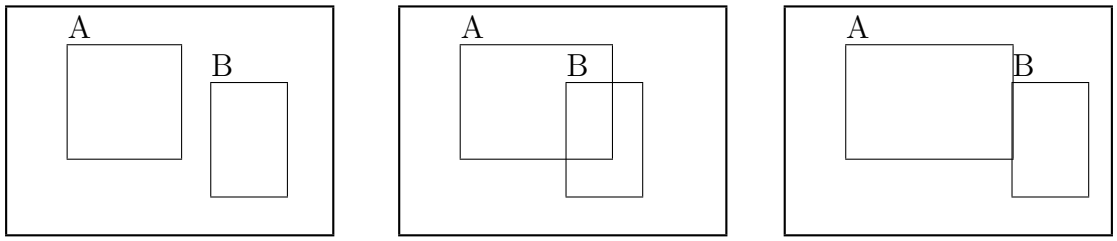
Thus, if *inside*(A,B) or *cover*(A,B) holds, then *overlap*(A,B) and *overlap*(B,A) also hold.

We base our definitions for the topological relations on Allen's temporal interval algebra [1]. Table B.1 gives the interval relations and Table B.2 provides the definitions for the topological relations with respect to these interval relations. In Table B.1, $[s_i, e_i]$ denotes a time interval, where s_i and e_i are the start and end points of the interval, respectively. In Table B.2, A and B represent the objects participating in the relations, and O$[x_1, x_2]$ and O$[y_1, y_2]$ denote the projections of object O on the x and y axes, respectively. Interval relations defined between objects A and B are enclosed by curly brackets and only one of them holds between the objects.

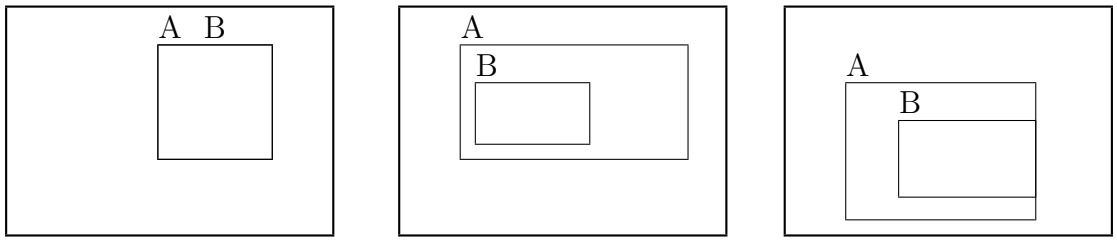
Examples of the topological relations are demonstrated by Figure B.2.

Relation	Condition
A equal B	A[x_1,x_2]{e}B[x_1,x_2] $\wedge$ A[y_1,y_2]{e}B[y_1,y_2]
A inside B	A[x_1,x_2]{d}B[x_1,x_2] $\wedge$ A[y_1,y_2]{d}B[y_1,y_2]
A cover B	(A[x_1,x_2]{di}B[x_1,x_2] $\wedge$ A[y_1,y_2]{fi,si,e}B[y_1,y_2]) $\vee$ (A[x_1,x_2]{e}B[x_1,x_2] $\wedge$ A[y_1,y_2]{di,fi,si}B[y_1,y_2]) $\vee$ (A[x_1,x_2]{fi,si}B[x_1,x_2] $\wedge$ A[y_1,y_2]{di,fi,si,e}B[y_1,y_2])
A overlap B	(A[x_1,x_2]{o,oi}B[x_1,x_2] $\wedge$ A[y_1,y_2]{d,di,s,si,f,fi,e,o,oi} B[y_1,y_2]) $\vee$ (A[x_1,x_2]{d,di,s,si,f,fi,e,o,oi}B[x_1,x_2] $\wedge$ A[y_1,y_2]{o,oi}B[y_1,y_2])
A touch B	(A[x_1,x_2]{m,mi}B[x_1,x_2] $\wedge$ A[y_1,y_2]{d,di,s,si,f,fi,o,oi,m,mi,e}B[y_1,y_2]) $\vee$ (A[x_1,x_2]{d,di,s,si,f,fi,o,oi,m,mi,e}B[x_1,x_2] $\wedge$ A[y_1,y_2]{m,mi}B[y_1,y_2])
A disjoint B	A[x_1,x_2]{b,bi}B[x_1,x_2] $\wedge$ A[y_1,y_2]{b,bi}B[y_1,y_2]
A contain B	A[x_1,x_2]{di}B[x_1,x_2] $\wedge$ A[y_1,y_2]{di}B[y_1,y_2]
A coveredby B	(A[x_1,x_2]{d}B[x_1,x_2] $\wedge$ A[y_1,y_2]{f,s,e}B[y_1,y_2]) $\vee$ (A[x_1,x_2]{e}B[x_1,x_2] $\wedge$ A[y_1,y_2]{d,f,s}B[y_1,y_2]) $\vee$ (A[x_1,x_2]{f,s}B[x_1,x_2] $\wedge$ A[y_1,y_2]{d,f,s,e}B[y_1,y_2])

Table B.2: Definitions of Topological Relations



(a) disjoint(A,B), disjoint(B,A) (b) overlap(A,B), overlap(B,A) (c) touch(A,B), touch(B,A)



(d) equal(A,B), equal(B,A) (e) contains(A,B), inside(B,A) (f) cover(A,B), covered-by(B,A)

Figure B.2: Topological Relations

B.3 Temporal Relations

Time is a very important component of video data, and thus, it is imperative that a video data model incorporates it in an efficient and compact way for processing and querying of video data.

Time information may be represented in two ways, equivalent to each other in terms of completeness to capture the semantics of video data: *time points* and *time intervals*. A *time interval* is identified as the basic anchored specification of time, which, in the case of video data, specifies a set of consecutive frames in video. And, a *time point* is a specific anchored moment in time that corresponds to individual frames for video data.

We use time intervals to model the time component of video data and all directional and topological relations for a video have a time component, a time interval specified by the starting and ending frame numbers, associated with them, during which the relations hold. With this time component attached, relations are not anymore simple spatial relations, but rather spatio-temporal relations that we use to base our model upon and to query video data.

The topic of relations between temporal intervals has been addressed and discussed in [1]. There are seven temporal relations: *before*, *meets*, *during*, *overlaps*, *starts*, *finishes* and *equal*. Inverses of these temporal relations are also defined and the inverse of the temporal relation *equal* is itself. Definitions of these seven temporal relations are given in Table B.1.

B.4 3D Relations

In the video keyframes, the salient objects certainly have different distances to the camera. This distances are encoded via 3D relations. Since automatic deduction of 3D relations from 2D data is impossible, all of the 3D relations are manually extracted by the user during knowledge-base population. Thus, they are stored as facts in the system. The possible 3D relations that the user may select are

listed as follows:

(a) *infrontof*
(b) *exactly-infrontof*
(c) *behind*
(d) *exactly-behind*
(e) *touch-from-behind*
(f) *touched-from-behind*
(g) *at-the-same-level*

In the above list, *exactly* means overlapping of salient objects.