University of Southampton

Video Analysis for Content-Based Applications

by

Stephen Chi Yee Chan

B.Sc. (Hons)

A thesis submitted for the degree of

Doctor of Philosophy

in the

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

August, 2001

University of Southampton

ABSTRACT

Faculty of Engineering and Applied Science

Department of Electronics and Computer Science

Doctor of Philosophy

Video Analysis for Content-Based Applications

by Stephen Chi Yee Chan

B.Sc. (Hons)

The analysis of video for semantics presents significant advantages for many content-based applications. An indicator of this is the recent trend towards the extraction of video semantics in the video processing community. Techniques for the extraction of semantics are varied and often application specific. Efficient interaction and interpretation of video is only as good as the underlying video content representation.

This thesis is concerned with the development of video analysis techniques which support the extraction of semantics such as object and event identity from video. A new methodology for video analysis in which video is processed for high-level content through a hierarchy of visual analysis modules is proposed. Each module is constrained by a context which improves the performance of visual recognition. The design, implementation and testing of the new methodology resulted in a video analysis platform called The VCR System (The Video Content Recognition System).

This thesis proposes a novel way to generate and handle video-objects, which is a step towards overcoming some of the problems in current machine vision. Problems in the manipulation of temporal data such as moving objects in video are examined. The problems include temporal correspondence of objects from one frame to the next, object occlusion, and false-positive object detections. The VCR System contains algorithms which will link objects in a generic way, suppress

false detection of objects, and predict missing objects (either due to occlusion or missed detections).

The contributions to the area of video analysis are demonstrated and evaluated by developing the VCR System to locate faces in video. A further more substantial demonstration is presented by analysing videos of snooker.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I would like to express my thanks and admiration to my supervisor Dr. Paul Lewis for his tremendous support and guidance during this work. The opportunities that have arisen during the course of this work have yielded memorable contributions beyond the research environment, and for which I can only thank Paul again.

I would also like to say thank-you to my dearest parents for their support and encouragement. Thanks Mum, and thanks Dad. Thanks also go to my wonderful sister, Carol, and brother, Sammie.

My gratitude also extends to the many close friends that I have made within the IAM Group and beyond. Cheers!

Finally, I would like to thank and acknowledge the EPSRC for my funding, and the Department of Electronics and Computer Science for providing the necessary facilities for this research.

Chapter 1

Introduction

The holy grail of Computer Vision is a quest to give computers sight and understanding of visual information. Although the field of Computer Vision has advanced remarkably from early experiments in content-based analysis, it is far from the perceived performance of the human visual system.

In recent years the amount of digital video has grown at a rapid rate, and it is likely to continue in this way into the foreseeable future. The majority of videos are stored on tape and film, and digital formats such as Video CDs (VCD's) and Digital Versatile Disks (DVD's) are becoming a standard for domestic use. Developments in digital storage technology have made it possible to store entire movies on computers, and it is possible to stream digitised video over the Internet, and broadcast digital signals to televisions. Advances in areas of digital video such as storage media, data formats, transmission, and compression technologies are relatively mature when compared to techniques that analyse video content.

This thesis is concerned with the development of processes for the extraction of high-level information in video sequences. The research has resulted in the development of a platform to detect visual content integrated with symbolic

methods for analysis. This chapter will begin by describing the motivation for this work.

1.1 Motivation

Humans rely on their visual senses for a vast majority of tasks. A computer with visual capabilities will present significant advantages for a large variety of applications. For example, the automatic detection of defects in the manufacturing of goods, or assisting the retrieval of visual information in content-based applications. One of the common ways to register visual information into a computer is through a video camera. Electronic signals generated by the video camera are converted into a digital format suitable for computer analysis; the digital recording is also known as digital video. To set the scene, I will begin by briefly describing a tentative view on the way humans perceive video and the way it is represented in a computer. I will then highlight the emerging problems of content-based video analysis.

Broadly speaking, video consists of two parallel data streams: an audio component, and a visual component. To humans, video is perceived as a close visual analogue of what an actual observer in the scene sees, where objects, colour, texture, shape and other visual phenomenon are experienced. Digital video can be viewed as a set of images (frames) displayed in a sequence. Each frame is composed of elementary building blocks called pixels. Variations in the colour of pixels form what we perceive as perhaps recognisable content. However, to a computer a collection of pixels does not directly yield any notion of high-level content.

Unlike textual documents whose contents are in a form suitable for operations such as content-based searching and location of keywords, video frames and their pixel composition do not present this luxury. The advantages of 'human-level' content such as text may be combined with video to overcome some problems of video. Video formats such as MPEG 4 and MPEG 7 store textual descriptions

which relate to their contents. However, much of the textual content is manually derived. Providing methods which automatically create 'meaningful' descriptions directly from pixel data is still a profound problem in computer vision.

Automatic visual analysis of video presents problems for video (and image) archives as they are increasingly being stored and processed digitally, raising a need for efficient and effective indexing and retrieval in large volumes of video information. Interacting with video by high-level means will benefit users, thus, pursuing the development of automatic annotation techniques is the direction of this research.

To give the reader a better insight into the way video is processed for content a brief introduction to video processing techniques is given in the next section. However, a more in-depth description will be given in the literature review in Chapter 2.

1.2 An Introduction to Video Analysis

A stream of images displayed one after another on a computer screen creates an intended perceivable effect of 'animated' visual content. The 'interpretation' of this content using a computer is one aspect of video analysis. This section aims to give a brief introduction to video analysis and to introduce some common terminology used throughout this thesis. Caliani *et al.'s* (9) paper on commercial videos will be used as an example to represent the broad view of recent video analysis.

So far, video has been described as being composed of frames, and frames composed of pixels. In the opposite direction of abstraction, consecutive frames are grouped together to form *shots*, and contextually related shots are merged into *scenes*. A shot is a single sequence of a motion picture or a television program recorded by one camera without interruption, and it is regarded as a basic unit for a majority of video analysis applications.

One of the primary tasks of video analysis is identifying the beginnings and

ends of shots. Identifying shot boundaries is complicated by different types of transitions that connect shots together. The basic shot transition is a *cut* and is generally classed as an abrupt transition e.g. one view instantly changing to another. Another type of transition called a *dissolve* (gradual transition) merges two sequences by partly overlapping them.

Once a video is broken into shots, each shot can then be analysed for its content. Segmenting video thus produces more manageable units of video. Caliani *et al.* extract color and line properties from the *key-frame* which is a representative frame for the shot. In their case they use the first frame of the shot as the key-frame. The extracted colour and line contents represent information that is derived directly from the pixels of each frame. Information formulated in this manner are known as *low-level* features.

Another level of content abstraction is now introduced. From the low-level features *semantics* are deduced to give a better description of the overall meaning of each shot. In this case semantics that describe the commercials in categories such as 'Practical', 'Utopic', 'Critical' and 'Playful' are derived from shot properties (e.g. number of cuts) and content attributes.

To summarise, the first phase of video analysis is the isolation of shots. Each shot can then be analysed for low-level features. Once the abstraction of pixel information is quantified, semantics are inferred to give a high-level interpretation of content. This summary highlights the typical stages of processing video see Figure 1.1 for an illustrative overview. The next section will follow on to discuss the types of content that can be extracted from video and from which high-level information may be deduced.

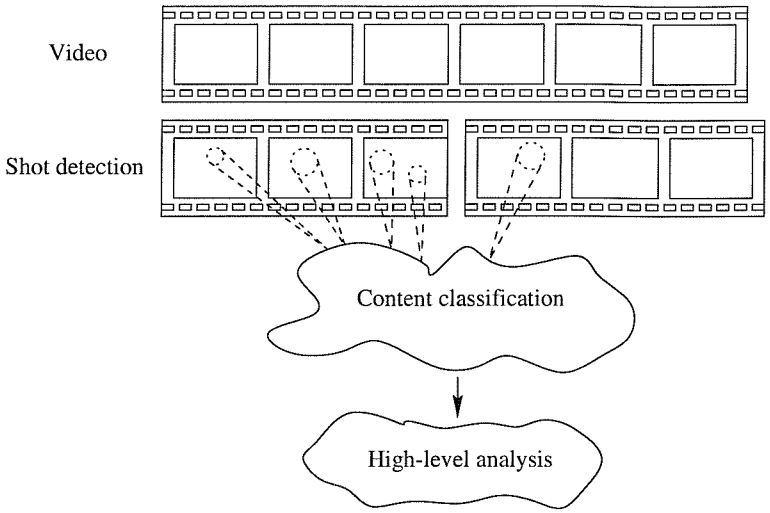


Figure 1.1: An illustration to reinforce concepts in video analysis.

1.3 Representations of Video Content

Invariably, abstracting video into what we call its content will vary according to the nature of the application. Dimitrova *et al.*(20) state that *"...video sequences must be classified based on the semantics of their contents"*. Recently, there is evidence of a trend towards a desire to process visual information for semantics (14) (8) (64). This section will briefly introduce the typical types of content extracted from video and discuss their potential for semantic analysis. The types of content the discussion will concentrate on are: shot boundaries, features across shots, and motion within shots.

A unique aspect of video compared to images is that it can convey time varying (or temporal) information. The content types discussed are related to temporal features or aggregated low-level features extracted from video. The detection of shot boundaries is a common step in the initial stages of video analysis. Although shot boundaries do not explicitly express actual content information, measuring the style and frequency of detected shot boundaries can be used to infer the type of content. For example, the impression of "action" can be portrayed by certain editing operations such as frequent hard cuts which may indicate fast ac-

tion content (42). Another example of shot detection to infer content is reported by McGee *et al.* (44) where the frequency of cuts was used as a characteristic feature to isolate commercial breaks from program segments.

The analysis of features across an entire shot can be used to detect semantic content and this is evident from research in the domain of news video. Separating anchor-person shots from news items is desirable for video classification purposes. Zhang *et al.* (78) identify potential anchor-person shots by calculating the average and variance of histogram and pixel differences between frames over an entire shot, and they are verified by testing the metrics against a set of predefined constraints. Candidate shots are then classified into anchor-person shot types e.g. single anchor-person and double anchor-persons. This involves searching for variations within regions (templates) characteristic of anchor-person scenes. The variations are modelled by the same metrics used to find potential anchor-person shots.

Motion is an integral part of a shot and has received much attention over the years. Patel *et al.* (49) generated a qualitative description of camera motion present in a shot by searching for specific patterns of motion vectors. Motion classes include stationary, panning and zooming. Recently, attention has shifted towards tracking specific regions within video (69) (5) (11) (19). These regions are often called *video-objects* which may or may not encompass actual objects within their respective shots. Generally, motion information of video-objects is calculated by matching features of their regions to subsequent frames to extract displacement values i.e. motion vectors, the results of which are used to indicate 'object' trajectories.

Content descriptors, like the ones described, can be separated into two classes: *syntactic* and *semantic*. It is important to highlight the differences between the two classes because both represent content but on different levels of intuition. Syntactic features can be extracted from video without any background knowledge and therefore describe the physical properties of the underlying

content. Examples include colour, edges and texture. Features derived from syntactical features are called semantics. The term 'semantic' has been used in the literature to convey information which vary at different levels of abstractness such as shots-breaks, objects, and events.

Fischer *et al.* (22) describe video content analysis with respect to syntactic features as *"A direct recognition of the content without further transformation is in general impossible"* which reinforces Dimitrova's statement at the beginning of this section. The implication of this is that high-level content-based access to video demands the use of semantic features.

The content types introduced in this section allow a general insight into video content at a semantic level. Semantics which describe visual content can have varying levels of conciseness from very narrow terms, e.g. a shot-break, to very broad terms, e.g. general scene descriptions. Narrow terms can be too vague for general intuition, and broad terms can convey too much meaning for it to be used effectively in some high-level content-based applications. Access to content, say via a query, is limited to the *expressiveness* of the underlying representation (57). The question that begs to be asked is what level of granularity should visual content be resolved to?

It is envisaged that the requirements of sophisticated multimedia applications will demand higher level means of access to content to aid in effective 'understanding' and retrieval of visual information. This will help answer queries such as one requesting all shots containing Mr X to be retrieved. A step in this direction will require new techniques to interpret visual content on a semantic level. Courtney (17) proposed resolving video into objects, which is probably the most fundamental *high-level* building block of content. It is fairly intuitive to visualise that objects convey an instinctive handle with which it is convenient for humans to express content and interaction.

1.3.1 Object-level Content

Representing the content of general video in terms of object-level entities is highly appropriate for many video applications such as surveillance. However, this approach is not without its problems. The main barrier is the initial step of recognising objects in video. Unfortunately, the general recognition of objects in video sequences is beyond the current state-of-the-art of machine vision, and it is not as well researched when compared to efforts in recognising objects in images. Video is inherently noisy and the resolution of digitised frames is low when compared to images used for object recognition. Thus, current object-recognition technology may not be as effective when applied to videos. Although it may be worth pursuing means of applying object recognition techniques used for images to video, this research investigates an *alternative* route to recognise objects.

Some success in classifying video for its semantic content is possible by constraining the video domain (78) (30) (62) (44) as with image object recognition techniques that use images in controlled conditions. The context in which these domains operate allows assumptions that enable a more manageable environment to tackle vision problems. One such problem is object recognition which can be reduced to satisfying parameters such as width and height of a region e.g. newly detected regions within a video footage of a carpark that conform to a range of aspect ratios may indicate people.

To summarise, techniques for the general recognition of objects in video are unavailable. However, by exploiting the context of a given domain, objects can be 'recognised' in a non-explicit way. In this research, binding video to a particular context is an integral part in the extraction of semantic information. An in-depth review of systems attempting to extract object-level information from video is given in Chapter 2.1.2.

1.4 The Research Objectives

The overall aim of this research is to develop new techniques to analyse digital video for high-level content. The level of analysis proposed aims to extract information as closely as possible to the level of objects for improving video access and interaction. The results of content analysis are to be represented as data-objects containing 'slots' that store object-level attributes such as colour, and position. A demonstration of how this level of abstraction will contribute towards machine-level 'understanding' of visual input will be presented.

The outcome of this research should be usable by people who wish to obtain high-level descriptions of video. Possible applications include: surveillance, monitoring system design, indexing and retrieval of video databases, and perhaps sports commentary. Examples provided to demonstrate the concepts introduced by this research will give a strong indication of the immediate possible applications.

1.5 Thesis Overview

The remainder of this thesis is organised into five chapters. In Chapter 2 a literature review of video analysis techniques is presented. It begins with a review of shot boundary detection techniques to complement this introduction to video analysis. A review of techniques used by vision systems to resolve video into video-objects and semantics is also given.

Chapter 3 introduces a multi-moduled video analysis architecture called the Video Content Recognition System (VCR System). The VCR System is a platform developed for integrating and testing video analysis techniques.

Chapter 4 introduces a set of modules called Specific Content Detectors (SCDs) which are common image video operations integrated into the VCR System. Implementation and algorithm details are described.

Chapter 5 demonstrates a sophisticated implementation of video decompo-

sition, specifically top-views of snooker shots. Knowledge and rules are used to extract high-level semantics.

Chapter 6 gives the conclusion of this research, a summary is given and avenues of future work are presented.

Chapter 2

Review of Video Analysis Techniques

The objective of this chapter is to review and discuss the state-of-the-art techniques used in the extraction of high-level content in multimedia systems.

The chapter begins with a survey that covers research on the extraction of semantic information from video. After the presentation of the survey a discussion follows which raises questions and identifies areas that require further research, and finally, the conclusions are presented.

2.1 High-level Video Analysis

In Chapter 1 it was stated that there is evidence of a trend towards processing visual information for semantics, and that semantics that convey too much information are seen as a limitation for multimedia applications. To overcome this limitation it was proposed that by resolving video into objects, explicit handles are then available for applications to gain access to the underlying content.

The Oxford Dictionary defines the word semantic as: *of or relating to meaning in language.* Labels representative of video content which have significance in human language vary in granularity where visual descriptions can span from specific to vague. The granularity of the labels is determined by the techniques used to extract semantic information from visual data.

Figure 2.1 gives an example of granularity in the semantic spectrum of outdoor scenes. The semantic spectrum has two extremes ranging between *coarse* and *fine.* However, some researchers will refer to these extremes as high-level and low-level semantics. The definition of 'semantic' already implies a 'high-level' significance but it does not define the level of intuition or abstractness, and hence, the *granularity.*

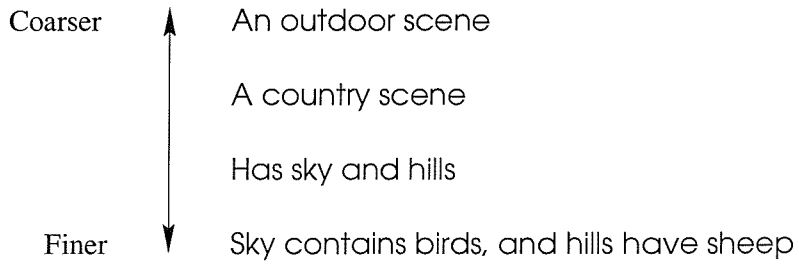


Figure 2.1: An example of how semantics can be expressed at different resolutions of intuition in a *semantic spectrum.*

The general definition of coarse semantics is semantics which convey broad meanings, such as "a picture of an outdoor scene", and conversely, fine semantics convey narrow meanings, such as "the outdoor scene has sky and hills". Towards the fine end of the semantic spectrum, semantics become finer, which describe ever more specific portions of content which make up the overall picture. This implies that a number of fine semantics may be required to give a fuller description of the content. However, as semantics becomer finer, the overall semantic meaning diminishes. Semantics which are too fine may not be able to convey the underlying semantic meaning at the appropriate level of abstractness for them to be useful.

Grouping related fine semantics into a single semantic entity gives a broader definition, thus it is coarser in the terminology of the thesis. The granularity of

semantics is appropriated by the application concerned. In Section 1.3, it was proposed that the level of granularity for video analysis are object-level entities, which was reasoned to be the most appropriate level for content-based video analysis.

In Chapter 1 it was identified that applications constrained to a particular video domain resulted in some success in the extraction of semantics such as objects where the context of the video allowed for specialised analysis. Although the objectives of this research specifically stated that the content extracted from video is to be on a similar level to objects it must be noted that object recognition is not always possible. This situation may be improved by processing video for coarse-grain semantics which may aid the processes that locate objects by narrowing the context.

Often, it may be heard that multimedia systems are criticised for being too specific in their chosen video domain. An approach towards general video analysis may be seen by linking domain specific applications together to form a larger network of specialised domains. The context of video may be used to aid the connective infrastructure to channel video to the appropriate specialised domain. The derivation of coarse-grain semantics, e.g. analysing the frequency of shot-breaks, may provide the necessary clues for context identification. Identifying the visual context of video narrows the scope which in turn allows for more specialised visual analysis. As a potential aid for general video analysis, it is conceivable that the visual context may be used to build a scaleable infrastructure by linking domains of domains which channel the appropriate visual analysis tools.

The objective of this section is to report on the trends in recent video analysis by surveying the work related to processing video for semantic content. I have defined two distinct areas of review that reflect the extremes of the semantic spectrum. I will begin with a review of papers that are classed in the category of coarse-grain semantic analysis.

2.1.1 High-level Video Analysis for Coarse Semantics

In this thesis the term coarse semantics is used to describe video semantics that do not give explicit details about the underlying composition of the overall content. The techniques which map visual primitives to coarse semantic categories will be the primary focus of this section. In the Introduction we used Caliani as our example on Video Analysis, and this theme is continued by reviewing their work on the study of video commercials.

Alberto Del Bimbo cited as a common author in (9) (16) (14) (3) (15) addresses the problem of retrieval of commercial videos based on the semiotic categories: Practical, Critical, Utopic and Playful. In their system, a user interface allows users to specify the strength of each semiotic element to influence the selection of video retrieval.

The initial stage of translating video into semantic categories involves two processes: video segmentation and shot content description. The role of video segmentation is the identification of the start and end point of each shot. Del Bimbo *et al.* calculate metrics based on editing characteristics such as: *cuts*, *dissolves*, and *rhythm*. The rhythm of a video is related to the shot duration and to the use of cuts and dissolves, and these editing procedures are used by film-makers to induce tension by using a sequence of gradually shorter shots, for example. Metrics based on these editing features are formally expressed in Equations 2.1, 2.2, and 2.3 respectively.

$$F_{cuts} = \frac{\#cuts}{\#frames} \tag{2.1}$$

$$F_{dissolves} = \frac{\#dissolves}{\#frames} \tag{2.2}$$

$$r(i_1, i_2) = \frac{\#cuts + \#dissolves}{i_2 - i_1 + 1} \tag{2.3}$$

Where F_{cuts} is an overall feature related to the number of cuts in a video, and likewise $F_{dissolves}$ is in relation to the number of dissolves. $r(i_1, i_2)$ takes frame

numbers over a frame interval of $[i_1, i_2]$ and gives a measure of the rhythm. From these features other feature descriptors are derived, see Equations 2.4 and 2.5.

$$F_{breaks} = r(1, \#frames) = F_{cuts} + F_{dissolves} \tag{2.4}$$

$$F_{continuous} = 1 - F_{breaks} \tag{2.5}$$

Similarly, the second part of the initial stage, shot content description, generate measures from low-level visual features. Colour and line properties are extracted from the keyframe of each shot. Four measures are generated for colour and these are expressed in Equations 2.6, and 2.7, where $F_{recurrent}$ expresses the ratio between colours which recur in a high percentage of keyframes and the overall number of significant colours in the video sequence, and $F_{saturated}$ expresses the relative presence of saturated colours in a scene.

$$F_{sporadic} = 1 - F_{recurrent} \tag{2.6}$$

$$F_{unsaturated} = 1 - F_{saturated} \tag{2.7}$$

The measures for lines is given in Equation 2.8, where $F_{slanted}$ measures the ratio of slanted lines with respect to the overall number of lines in a sequence.

$$F_{hor/vert} = 1 - F_{slanted} \tag{2.8}$$

The visual features are further classified into categories of semantic significance. The derived measures are combined in a way such that the semantic categories: practical, utopic, critical, and playful are expressed, see Equations 2.9 2.10, 2.11 and 2.12.

$$S_{practical} = F_{dissolves} + F_{hor/vert} + F_{unsaturated} \tag{2.9}$$

$$S_{playful} = F_{cuts} + F_{slanted} + F_{saturated} \tag{2.10}$$

$$S_{utopic} = F_{cuts} + F_{dissolves} + F_{recurrent} \tag{2.11}$$

$$S_{critical} = F_{continuous} + F_{sporadic} + F_{hor/vert} \tag{2.12}$$

Lienhart *et al.* (42) analysed German television broadcasts for commercials and attempted to recognise the commercials from a known pre-processed set. Commercials are often characterised by large amounts of motion, being highly animated and full of action. The sensation of action is supported by high frequency of shot-cuts and rapid changes in colour. Other visually perceivable features commonly associated with commercials are text, still frames at the end of commercials, and editing habits. Although these features are measurable, German television commercials are separated by a short break of 5 to 12 dark frames which can be reliably detected and used as a commercial broadcast indicator. The recognition of commercials is performed by matching a 'finger-print' of each commercial to a database of preprocessed commercials. In this case colour coherence vectors (48) proved to be a superior finger-print.

Related work by McGee *et al.* (44) on the identification of commercials uses a similar approach to that of Lienhart *et al.* . Their research analysed over thirteen hours of US television broadcasts for the cut-rate, static sequences, the presence of black frames and the presence of text. Like German television broadcasts the transmission of US commercials are usually preceded and terminated by a series of black frames. Their experimental results revealed that the cut-rate alone was able to detect commercials and the combination of cut-rate and black frames improved the reliability.

In the context of this thesis distinguishing commercials from programs using domain dependent video cues generate very broad semantic terms e.g. a commercial block or a non-commercial block. It is often sufficient to search for specific video signatures that infer high-level video content but the resulting semantic

labels are typified by coarse-grained categories. The identification of broad categories may have an advantage in that it allows the context to be bounded for more sophisticated analysis where the underlying content may be inferred.

It is evident from the literature that video editing techniques are exploited for semantic information (1). In particular, the genre of news video has received much attention over the years with respect to content-based video analysis. Much of the early work in news video analysis involved segmenting the video into shots and classifying them as anchor-person and non-anchorperson shots (78). In this case certain regions (model templates) of the video image are measured for inter-frame activity which are characteristic of anchor and non-anchorperson shots. The relatively significant success of news video analysis may be due to a priori knowledge of the presentation format (the semantic structure) and its characteristic visual content. In the terminology of this thesis this conforms to a known context.

It appears finer semantic categories are obtainable if the semantic categories are known. Hanjalic *et al.* (28) designed a semi-automatic news analysis system where news bulletins are classified according to pre-specified categories. News video is analysed for anchorperson shots which are located by matching the key-frame of each shot with all other shots generating a set of dissimilarity values. The lowest values are then assumed anchor-person shots. Candidate anchor-person shots are verified by examining the inter-frame activity of each shot. News bulletins which are assumed to be non-anchor person shots are classified into finer semantic categories, e.g. News about the weather. Instead of focussing on a visual solution their efforts concentrated on analysing the audio stream for spoken words. Spoken words are used as an audio-key that match the user's pre-specified categories. It appears that audio cues are a powerful feature for resolving video into semantic labels (21) and is proving to be popular for news video analysis (46) (30).

In summary, it can be said that a prior knowledge of the structure of video

allows the type of content to be inferred without explicit visual analysis. The coarse semantics generated by the visual analysis systems tend to be related to visual concepts rather than visual content. The granularity of semantic items generated by the news video systems is limited due to the impractical number of genres. The next section reviews the literature on video analysis applications which focus on finer-grain semantics.

2.1.2 Constrained Video Analysis

The previous section reviewed work on video analysis intended for general video or video with a large variation in content. This section focuses on literature where video analysis is constrained to a single video genre, and where object-level entities are extracted.

Saur *et al.* (62) analyse basketball video for fine-grain semantic basketball content such as fast break, steal or loose ball, and potential shots. The events fast break and steal or loose ball are strongly correlated with the motion within basketball scenes and the computed motion of the camera. Identifying probable shots relies on a set of heuristics which is a combination of camera motion and knowledge of basketball video structure. The motion of this particular genre allows events to be inferred because the events have consistent motion characteristics. Although video-objects were not explicitly detected the a priori knowledge of the constrained video genre offered the possibility of detecting high-level events that may be labelled with fine-grained semantic significance.

Within the literature of video analysis one of the most frequent object-level entities in video analysed for is the human face. Satoh's work (60) analyses drama videos for face sequences and associates each face sequence with its role-name. Faces are automatically (56) extracted from each frame and a skin colour model is computed for each face which is used to track faces in subsequent frames. The next stage of processing evaluates each face-shot for similarity (70) and similar face-shots are aggregated to form a larger face-shot of the particular actor/

actress. The aggregated face-shots are classified to their corresponding role names by clustering face-shots to named face sets. The performance of the system was evaluated using a test set of two episodes of a drama series to yield an accuracy of 70%.

The process of associating face objects with a semantic label has seen success by evaluating the co-occurrence of detected faces with transcripts of news video. Satoh *et al.* (61) labels faces detected in news video which have similar filming characteristics as drama videos. The general process of associating faces and names in news video begins with face extraction which results in face locations and are evaluated for similarity across frames. Then names are extracted from news transcripts and video caption recognition is performed. The face-name association is determined by a co-occurrence factor that evaluates extracted face sequences, text from the video captions and recognised names from corresponding text transcripts.

The papers currently reviewed indicate that fine-grain semantic information may be extracted with relative success when videos are constrained. The reduction of video into objects imposes greater constraints as demonstrated by Courtney (17) where automatic content-based video indexing is based on object motion in stationary background scenes serving surveillance applications. Courtney's contribution processed video in three stages: 1) motion segmentation, 2) object tracking, and 3) motion analysis.

The first stage segments moving foreground objects from their backgrounds by comparing frames of activity with a designated static scene. The comparison involves subtracting the active frames from the static scene and then thresholding the absolute differences to yield binary areas where objects might have appeared. Morphological operations are applied to the binary images that produce larger and smoother regions. The resulting regions exceeding a pre-determined size are assigned labels.

The second stage tracks objects by linking object regions between frames.

Detected objects are stored in a structure called a V-object (video object) which contains place-holders for an object label, the object centroid, object velocity, and trajectory information. V-objects are linked to their corresponding V-objects in the next frame using position and estimated velocity information. The process eventually forms a motion graph of tracked objects and this is analysed in the next stage.

The final stage annotates the motion graph which describe events of interest. The tracked objects in the motion graph are grouped into structures that are representative of the paths of the objects in the video. The motion groups are described below:

- A *Stem*, represents a simple trajectory of an object in two or more frames. This grouping is used to determine the state of motion of objects, e.g. if tracked object centroids are different then the object is likely to be moving, otherwise it is stationary.

- A *Branch*, represents a highly reliable trajectory estimate of an object through a series of frames. A branch is classified as stationary if the whole branch is a stationary stem, otherwise it is moving.

- A *Trail*, represents the object tracking stage's best estimate of an object trajectory using a mutual nearest neighbour criterion. It is classified as stationary if all branches within a trail are stationary. Conversely it is moving if all branches within it are moving.

- A *Track*, represents the trajectory estimate of an object that may cause or undergo occlusion one or more times in a sequence. V-objects within a track are classified as stationary if all its trails are stationary, and are classified moving if all trails are moving.

- A *Trace*, represents the complete trajectory of an object and all the objects with which it intersects.

After motion analysis it is possible to designate the following motion events to each detected object in a video sequence. Various rules are used to define the motion events from the motion graph, e.g. Rest, is the tail of a moving stem, and the head of a stationary stem. Each event is recorded in an index table for future reference. Using these events, it is comprehensive enough to assist the analysis of video sequences, e.g. a removal event, to detect theft.

- *Appearance*: An object emerges in the scene.
- *Disappearance*: An object disappears from the scene.
- *Entrance*: A moving object enters in the scene.
- *Exit*: A moving object exits from the scene.
- *Deposit*: An inanimate object is added to the scene.
- *Removal*: An inanimate object is removed from the scene.
- *Motion*: An object at rest begins to move.
- *Rest*: A moving object comes to a stop.

An automatic surveillance system which labels events and interactions by Ivanov *et al.* (32) also analyses video using similar processing stages as (17). Their system is specifically designed to monitor activities in car parks which consists of three components: an adaptive tracker, an event generator and a stochastic parser.

The tracker processes video input from a camera which isolates object regions and records the movements of the objects. An adaptive mixture of Gaussians technique is used to determine foreground pixels from the background which are formed into object regions using a two-pass connected components algorithm. The correspondence of object regions between frames is accomplished by a tracking algorithm which incorporates both the region size and position.

The event generator maps object trajectories produced by the tracker onto a set of pre-determined events. In total nine events (car-enter, person-enter, car-found, person-found, car-exit, person-exit, car-lost, person-lost, object-stopped) are generated according to a set of rules. As an example, the car-exit and person-exit events are produced if an object-trajectory ended in one of the designated "exit" areas.

The stochastic parser maintains a set of parser states which are driven by the events from the event generator. The parser states are rules which interpret the events for further semantics, and thus, generating high-level automatic interpretations of video.

Parallels at the system-level of the vision analysis and the extraction of semantic content may be drawn in (17) and (32) where both systems extract semantic details from motion trajectories without performing object recognition. Other examples of multimedia/ vision systems that locate and track objects in video in a similar manner where objects are delineated, the object motion extracted, and analysed or stored may be found in (19)(40)(11)(5)(69).

Research on object location, tracking, and identification has been reported in the analysis of the tennis broadcasts (50)(51) and (67). Objects such as players and the tennis court are located and tracked which are used to facilitate content-based retrieval and sports broadcast applications. Reliable tracking and 'recognition' of player objects addressed in the the papers may be attributed to the a priori knowledge about the game of tennis.

2.2 Summary

The literature review supports the general idea of analysing video for semantic information in the following steps: 1) feature extraction, 2) feature correspondence across frames, 3) the extraction of semantic information, and 4) further stages of semantic analysis and semantic interpretation.

The level of semantic data that can be gained from analysing visual editing

effects is generally limited to the coarse category of the semantic spectrum. Although efforts such as those in commmercial video analysis attempt to improve on the granularity of semantic content by integrating domain dependent knowledge, the actual underlying visual content is not used.

The extraction of fine-grain semantics in constrained video analysis hinges on the examination of regions which may or may not belong to actual objects. Video analysis benefits from using a limited/ sub-genre of video content and allows the use of domain specific knowledge to derive (near) object-level semantics.

A common theme in the literature review is that once semantics are extracted from video, they are further processed for more semantics that give an interpretation of the video. The types of visual features sometimes require post-semantic processing to gain access to the underlying content. The literature review highlights several methods of post-semantic analysis and these include mappings of 'pre-semantic' terms, and rule-based techniques.

The next section of this chapter examines the limitations of the current work in video analysis and discusses ways to aid the processing of video for fine-semantic features.

2.3 Context Engineering

It is evident from the literature review that the field of video analysis is aiming towards the extraction of semantic information. The level of granularity with which extracted semantics is conveyed varies with the application. However, there is a limit to the extent to which current machine vision technology can extract semantic information from video.

The literature review points to works that detect shot breaks for the extraction of semantics, and works that detect signals embedded in video to infer content. Work using these approaches do not make full use of the physical temporal visual content, and their application base may be limited. A recent survey on automatic video indexing, Brunelli *et al.* (8), indicates that semantic video

analysis is largely based on the classification of features such as shots, motion, colour, and captions. However, there is little reported on analysing video for real objects. Another recent survey on content-based retrieval of images, Smeulders *et al.* (64), suggested that the approach for semantics to be associated with image data would entail the recognition of objects.

It may be argued that object-level semantics provide a better intuitive handle for humans to grasp visual content. The decomposition of video into objects covers a large base of visual applications such as indexing and retrieval for example. Unfortunately, the recognition of objects in general video is currently beyond machine vision and alternative steps and strategies for the extraction of fine-grain semantics must be used. The literature suggests that to make relatively successful steps towards fine-grain/ object-level semantics video analysis at this level is best approached by analysing a small sub-set of visual genres.

The recognition of objects may be improved if the context of a video is known. Since general object recognition is not possible, it may be possible to use an engineered alternative that is based on a known context of video(s) and domain specific knowledge. Although this may be seen as too specific, however, the combination of specific vision analysis domains builds on the generality aspect.

Figure 2.2, illustrates the arguments in the context of news video. The diagram illustrates how a news genre might be partitioned into domains for the purposes of semantic extraction of news video. The structure of the diagram implicitly embeds domain knowledge in the framework, and it is possible to see how the framework might scale to other genres by linking the news domain to a higher context domain.

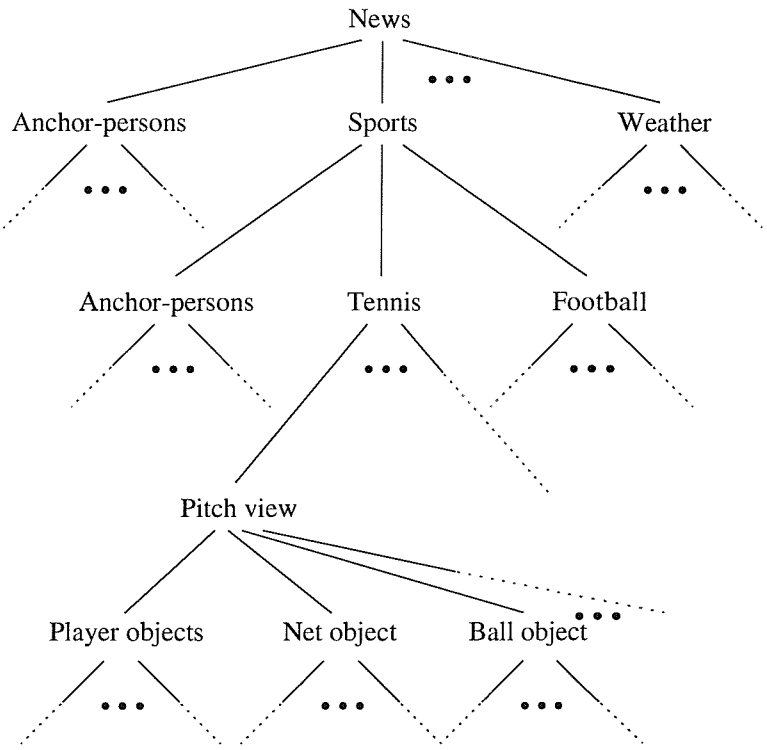


Figure 2.2: A diagram illustrating how a genre may be divided into different contexts which benefits and supports the extraction of semantics from video.

The two anchor-person domains demonstrate the ideas further. If the domain is capable of handling general anchor-person shots then the two anchor-person domains may be linked. However, if the anchor-person domain is not general enough to handle anchor-person shots from their respective domains then different anchor-person domains must be defined which use features in their local context to generate semantics. The leaves of the diagram represent object-level entities where lower-levels may refer to other object-level domains or image features.

Further support for the access to semantic level video content using domain knowledge is presented by Hampapur (27). The paper is focussed on indexing and retrieval but the issues for the generation of semantics applies to this thesis. Three types of knowledge are defined in his paper:

- Physical Knowledge: describes the constraints derived from the physical environment. For example, in basketball this includes colour of the ball, structure of the backboard and basket.

- Cinematic Knowledge: details the way video is filmed and produced. This includes camera motion, editing and camera locations used in producing the video.

- Semantic Knowledge: includes the knowledge about the video context, the temporal structure, and other high-level information. Referring back to the basketball example, a game has four quarters, with a 130 second break between Q1, Q2, Q3, Q4, etc.

The study has revealed that most of the techniques used in video analysis (specifically towards indexing) rely on the physical and cinematic knowledge to extract features and event labels. Access to video data at the concept level requires the use of semantic knowledge models.

In this thesis, object-level entities are considered to be the foundation for concept level access, and the recognition of these objects mandates the use of context (domain knowledge). The manual identification of features of interest which may belong to an object is gained as a result of experience and skill in processing visual data (35). The use of context with which to process visual content whereby bounded vision processing modules preserve parameters that define the result of experience appears to be the most natural alternative for object based recognition.

2.4 Conclusion

In conclusion, the area of video analysis is aiming towards the extraction of semantics. The level of semantic granularity varies, and is determined and limited by the features extracted from video, e.g. motion vectors and shot breaks. A distinction between coarse-grain and fine-grain semantics has been presented, and the prominent desire to interpret video in terms of objects manifests itself in general video analysis.

Analysing video for objects is difficult, but the situation may be improved if the context of a video is known. A known context of video allows assumptions to be made where they may be used to infer a more detailed level of semantics in video, e.g. objects. The next chapter presents a framework that uses the ideas of a small video context and domain specific knowledge to extract fine-grained semantics from video.

Chapter 3

The VCR System

In the previous chapter, a number of problems and limitations have been identified with the state-of-the-art video analysis. They may be summarised as follows:

- Semantics that convey too much information for multimedia applications is seen as a limitation.
- Objects may provide the best compromise for video semantic analysis but object recognition video is difficult.
- Techniques used in constrained video analysis may not be compatible with other domains.
- Content-based applications are moving towards semantic-level access.

Steps toward solving some of these problems are possible. This chapter introduces the Video Content Recognition (VCR) System as a new approach to video analysis. The VCR System assumes a context with which to locate and identify objects contained within a video shot. Video is decomposed into video-objects by a process called *hierarchical visual decomposition*, and this will

be explained in detail later in this chapter. Extracted video-objects are used to derive semantics using a rule-based system. This approach is proposed as an attempt to address the problem of high-level interaction with video content using objects as the primary handles. The limitation of semantics conveying too much information and the desire for applications to have semantic-level access to content are addressed in the approach.

This chapter begins with an overview of the system in Section 3.1.

3.1 System Overview

This section presents an overview of the VCR System, and gives the necessary background for the reader to formulate the concepts used in this new approach to video analysis. I will first introduce the system as a whole, and then give a general explanation of the major components.

Initially, the VCR System can be viewed as a combination of two systems: a visual processing system, and a rule-based system. The visual processing system resolves video clips for high-level content such as video-objects and temporal visual effects, e.g. shot-breaks. Video-objects can be actual objects or regions that convey content, and temporal visual effects may be used to assist this process. Once video is decomposed into video-objects and visual semantics, knowledge and rules are applied to them to derive higher level semantics (including temporal effects) within the rule-based system.

It has yet to be proven that a single theoretical umbrella can tackle all aspects of machine vision, and while the development of a different and better science of machine vision is ongoing, the VCR System looks at an engineered alternative to an all embracing solution. The vision processing system of the VCR System packages together coordinated multiple vision techniques to process a specific category of video. The approach of the vision processing system encapsulates existing vision techniques into 'blackboxes' to create a highly modular system. The development of video analysis using this approach creates a scaleable system

that can be extended to incorporate other specialised vision processing modules. This potentially allows the framework to scale to different domains that are best processed with different techniques.

The VCR System, illustrated in Figure 3.1, is composed of four discrete systems: *Sensors*, *Specific Content Detectors (SCDs)*, an *SCD Correlator*, and an *SCD Manipulator*. Each component is introduced by presenting the steps showing how the VCR System processes video for content.

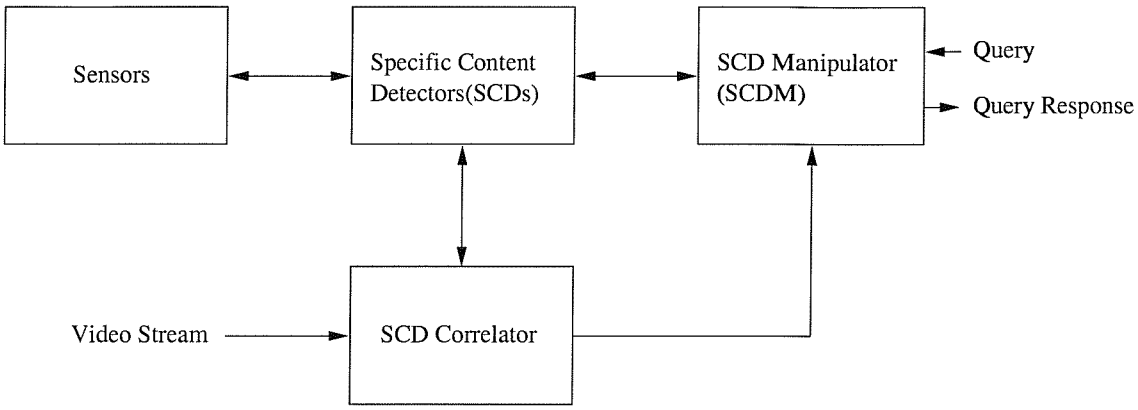


Figure 3.1: The framework illustrating the main components of the Video Content Recognition System.

The system is instantiated by presenting video to the SCD Correlator. The SCD Correlator functions as a buffer to the rest of the system, storing raw video images and the results of video analysis. The SCDs process video images for visual entities, the results of which are stored in the SCD Correlator. The SCDs use the Sensors to extract low-level features from the video images, and are then used to generate the visual entities. The Sensors are a shared pool of resources for the SCDs, and are responsible for extracting salient features from video. The main task of the SCD Correlator is to create video-objects by correlating visual entities across frames. The SCD Manipulator is a container for the implementation of a rule-base which filters and manipulates results from the SCD Correlator to derive *meaningful outputs*.

The VCR System tackles the problem of 'object recognition' using context-level assumptions that benefit classification techniques used within the SCDs, e.g. in the context of a tennis broadcast, a large green coloured patch may indicate a pitch, and players may be non-green segments restricted to pre-defined areas. The assumptions *model* a particular *context* that make the problem of *object recognition* more tractable. These assumptions may in turn reduce the need for sophisticated and computationally intensive techniques. The implementation of these model assumptions is built into the structure of the SCDs

The remainder of this chapter describes the components of the VCR System in detail. Section 3.2, describes the data structure used to represent content; Section 3.3, describes the SCD Correlator; Section 3.4, describes the Specific Content Detectors; Section 3.5, describes the Sensors, and Section 3.6, describes the SCD Manipulator.

3.2 The Structure of an SCD-Object

SCD-Objects represent video content, and are the outputs of SCDs. They are analogous to storage containers, and hold information in a consistent fashion. SCD-Objects are pervasive within the VCR System, and for clarity this section will cover them in detail.

SCDs produce SCD-Objects to represent visual entities or the indication of events, e.g. objects, or even points at which a scene-break occurs in video. SCDs maybe constructed from other SCDs, and inorder to provide compatible communication between these vision 'experts', SCD-Objects are used. The structure of an SCD-Object contains an extensible list of content attributes. SCDs assign the attributes of SCD-Objects with values that represent detected content in video. By standardising the output across all SCDs, a coherent operation of the vision processing framework (the SCDs) is maintained.

An SCD-Object is implemented as a C++ class, and contains the fields with its definitions as shown in Table 3.1. Objects may be represented in terms

of regions that may be assigned with attributes such as color, texture, shape, and motion (11). The SCD-Object implementation allows flexible extensions to include other attributes, and methods, e.g. functions to link to other SCD-Objects.

• Name	A semantic label
• Orientation	[0..359]
• Position	(x, y), where x,y $\in \mathbb{N}$
• Coordinates	A list of coordinates indicating shape on video image: $(x_0, y_0), (x_1, y_1)...(x_n, y_n)$, where x,y $\in \mathbb{N}$
• Colour	$(Colour, ColourStrength)$, where Colour is a text definition, e.g. "red", and ColourStrength, [0.0..1.0], is a value to indicate the
• Confidence	Probability that the object actually exists : [0.0..1.0]

Table 3.1: The content fields within an SCD-Object.

SCDs assign values to the fields of an SCD-Object that are representative of its detected content. To give an example of how SCD-Objects are used and created by SCDs, the following SCDs are assumed: Find Circles, and Compute Colour. Find Circles locates circles in a given video image and creates SCD-Objects for each circle. The fields that are assigned with values within SCD-Objects generated by Find Circles are: Name, Position, and Coordinates which form a circle. ComputeColour receives an SCD-Object as input and calculates the average colour within its defined polygon area. The SCD-Object returned from ComputeColour is the same SCD-Object used for input except for its Colour field assigned with an RGB (Red/Green/Blue) tuple.

The example of Find Circles and Compute Colour is illustrated in Figure 3.2. Find Circles is presented with a video image containing two coloured circles, and two SCD-Objects are produced. Suppose a query from the SCD Manipulator, or a request from another SCD requires the colours of the two SCD-Objects, the Compute Colour SCD receives the SCD-Objects, and 'fills-in' the Colour fields.

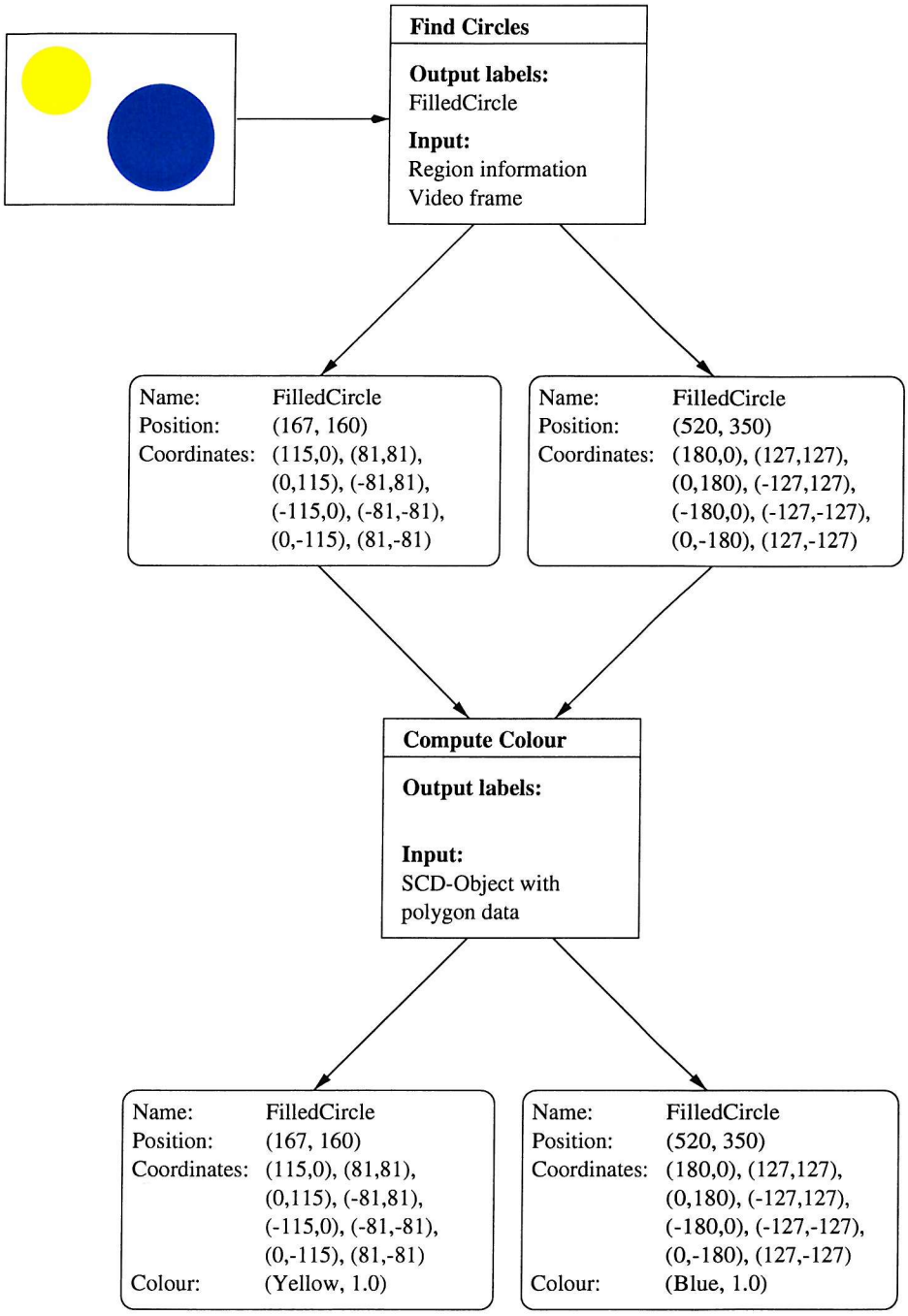


Figure 3.2: An example of two SCDs (Find Circles and Compute Colour) generating SCD-Objects.

Given that video is a high-bandwidth format, the speed of processing operations must be taken into account. Structuring SCDs to 'fill-in' fields of SCD-Objects as and when it is required are steps towards efficiency. In the given

example, Find Circles could have been implemented to automatically calculate the colour, but it is an inefficient use of computer power if the field is never used. The beauty of the use of SCD-Objects is that it attempts to standardise the representation of content such that 'general' SCDs (e.g. ComputeColour) can receive compatible SCD-Objects and produce results.

To summarise, a common content representation format has been introduced that is central to the functioning of SCDs. The steps in the example in Figure 3.2 illustrate the general flow of SCD-Objects between SCDs. However, the coordination between the modules of the VCR System needs to be explained, and the next section begins with a description of the SCD Correlator.

3.3 SCD Correlator

This section presents the *Specific Content Detector Correlator* (SCD Correlator). Firstly, this section will cover the concepts behind the SCD Correlator, and then the details of its internal mechanisms will follow.

The SCD Correlator is an entry point at which the VCR System receives and instantiates the processing of video. It functions as a buffer for an incoming video stream by storing a set of video frames as raw video images. The buffered set of raw video images are processed for visual entities. The primary task of the SCD correlator is to correlate visual entities from one frame to another, thus creating video-objects.

Figure 3.3 illustrates the schematics of the SCD Correlator and its external connections to the rest of the system. It consists of two pipes, a Video Frame $\text{Pipeline}_{0..m}$ and an SCD Results $\text{Pipeline}_{0..m}$, both of which are implemented as first-in-first-out queues. The Video Frame Pipeline and the SCD Results Pipeline are arrays of $m+1$ connected storage areas (buffers), and in this thesis these buffers will be referenced as B_j^{VFP} and B_j^{SRP} respectively, where $0 \leq j \leq m$. The Video Frame Pipeline stores raw video images, and the SCD Results Pipeline stores SCD-Objects from the SCDs for corresponding video frames, e.g. A video

image of a ball stored in B_0^{VFP} will have an SCD-Object representation of a ball stored in B_0^{SRP}). It is possible for more than one SCD-Object to be associated within a single SCD Results buffer.

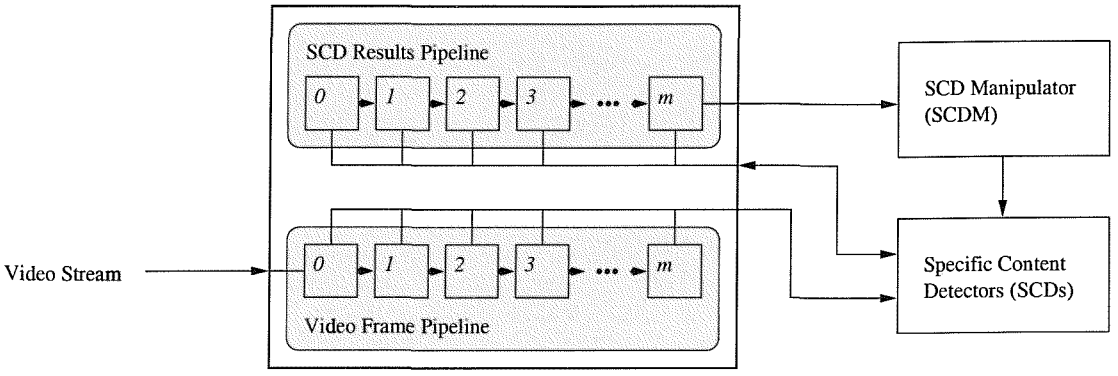


Figure 3.3: The box in bold represents the composition of the SCD Correlator. It contains two pipes where the Video Frame Pipeline stores raw video images, and the SCD Results Pipeline stores a high-level representation of the corresponding video image.

As a video stream enters the first buffer of the Video Frame Pipeline, it is converted and stored as a raw video image. Before a video image is stored in a buffer, the current contents of the buffer are shifted to the next buffer in line, and so on. This means that newer video frames are stored at the beginning of the pipe, and older frames are located towards the end of the pipe.

For each video frame that enters the Video Frame Pipeline, the SCDs are called to process the stored set of video images. The output from the SCDs are stored in the SCD Results Pipeline. Processed video images are stored as object representations that resemble the video images. As object representations of video frames are created, a process to link the objects along the pipeline is performed. The process of associating objects from one frame to the next creates temporally invariant 'objects' (video-objects). The algorithm that models the mechanics of the SCD Correlator is given in Table 3.2.

Repeat
Receive a video image, f, from video stream
Shift contents to next buffer within Video Frame Pipeline and SCD Results Pipeline
Write f to the first buffer in Video Frame Pipeline
Call SCDs
Correlate visual entities in Video Frame Pipeline

Table 3.2: The algorithm of the SCD Correlator.

To give a worked example, Table 3.3 shows a tabulated view of both pipelines where $m = 3$ and seven video images ($f_{0...6}$) are entering the pipeline over time-step t_i. At t_1, video image f_0 enters the Video Frame Pipeline, and at this point initiates the SCDs to process the current contents of the pipe. The output from the SCDs are placed in the SCD Results Pipe, and in this case r_0 represents the output. At t_2, the Video Frame Pipeline is signalled to receive video image f_1, but before f_1 is written to buffer B_0^{VFP} the contents of the buffers are shifted along the pipe, e.g. B_2^{VFP} to B_3^{VFP}, B_1^{VFP} to B_2^{VFP} and B_0^{VFP} to B_1^{VFP}. The whole process of calling the SCDs to process the contents is repeated.

Tabulated Video Frame Pipeline

t_i \ B_j^{VFP}	0	1	2	3
0				
1	f_0			
2	f_1	f_0		
3	f_2	f_1	f_0	
4	f_3	f_2	f_1	f_0
5	f_4	f_3	f_2	f_1
6	f_5	f_4	f_3	f_2
7	f_6	f_5	f_4	f_3

SCDs

Tabulated SCD Results Pipeline

t_i \ B_j^{SRP}	0	1	2	3
0				
1	r_0			
2	r_1	r_0		
3	r_2	r_1	r_0	
4	r_3	r_2	r_1	r_0
5	r_4	r_3	r_2	r_1
6	r_5	r_4	r_3	r_2
7	r_6	r_5	r_4	r_3

Table 3.3: A worked example of video images shifting in the Video Frame Pipeline and processed by SCDs where results are stored in the SCD Results Pipeline.

Each time the SCDs are called in response to new video input, a process to link detected visual entities in the SCD Results Pipeline is performed. The creation of video-objects in systems such as VideoQ (11) and Netra-V (19) track segmented regions across video frames, but they do not perform recognition of the content. The VCR System employs a new strategy to create labelled (recognised) video-objects, and the procedure is described in the next section.

3.3.1 The SCD Correlation Algorithm

When SCD-Objects are created by SCDs, they are independent entities that do not have any relation to any other SCD-Object in the SCD Results Pipeline. SCD-Objects that represent 'real' objects require continuity across frames in-order for them to be referenced as individual time invariant entities (video-objects), thus, improving the prospect of knowledge-based manipulation on video. The creation of video-objects is the primary task of the SCD Correlator.

This section presents algorithms that allow the SCD Correlator to create video-objects. The process involves associating a newer SCD-Object to a similar older SCD-Object (above some threshold). Figure 3.4 illustrates the process of linking SCD-Objects from one buffer to the next.

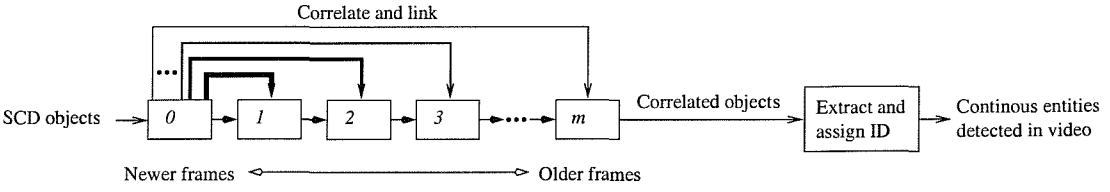


Figure 3.4: Correlation and linking of SCD objects in a series of buffers[0, m] in the SCD Results Pipeline.

The arrows emanating from the first buffer to the other buffers represent the sequence in which the contents within the buffers are compared. Given a correlation function $C(B_i^{SRP}, B_j^{SRP})$ that takes two buffers B^{SRP}, the sequence of comparing the contents of the buffers is shown in Table 3.4.

$n = 1$
Repeat
$C(B_0, B_n)$
$n = n + 1$
Until $n == m$

Table 3.4: Sequence of comparing buffers for an SCD-Object.

The combined effect of shifting frames along the SCD Results Pipeline and the order in which buffers are passed to the correlation function, results in buffers compared at every level, see Figure 3.5. The figure shows frame numbers, and indicates correlated frames for a SCD Results Pipeline of three buffers.

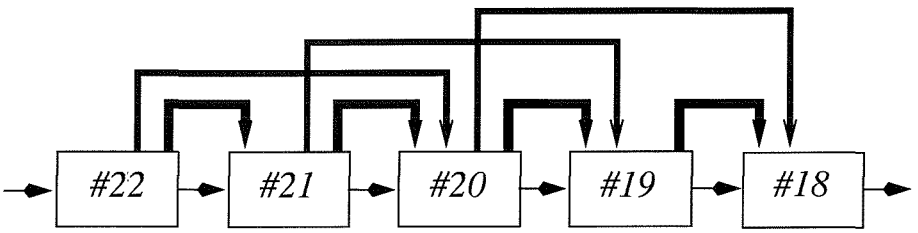


Figure 3.5: An illustration of frames compared by the SCD Correlator where $m = 2$, e.g. three buffers.

Given a case where an SCD-Object in B_0^{SRP} fails to find a link in $B_{1..2}^{SRP}$ but succeeds in B_3^{SRP}, then the 'missing' SCD-Objects are predicted using interpolation. The current version of the VCR System assumes a linear velocity motion model to predict missing SCD-Objects. However, a motion model that includes acceleration would not be difficult to adopt, and may be more faithful to the actual positions of objects in video. It is assumed that an object in a video frame corresponds to the closest object in the next frame (though this may not be entirely true in cases where motion of objects has a large acceleration).

The correlation of objects across frames is performed by matching semantically equivalent (identified by their labels) SCD-Objects from one frame to another. In the current implementation SCD-Objects are compared for 'similarity' based on the Euclidean distance of:

- Difference in size
- Relative distance
- Difference of orientation

The *difference in size* between two SCD-Objects is indicated by summing the differences of their corresponding shape coordinates. The *relative Distance* is the distance between two SCD-Object centres, and the *orientation* is the smallest difference of angular alignment. Equation 3.1 expresses the 'similarity' between two SCD-Objects as an unbounded dissimilarity metric.

$$dissimilarity(O^f, O^{f+n}) = \sqrt{\sum_{i=1}^{m}(O_i^f - O_i^{f+n})^2} \tag{3.1}$$

Where O_i^f is an SCD-Object in B_f^{SRP} referencing feature i, and $m = 3$.

The dissimilarity measure is based on the assumption that an object will have very little change from one frame to the next and that the corresponding object must be the closest in terms of feature space distance. It is quite conceivable some objects are better matched with more emphasis on particular feature(s). The current implementation of similarity does not support *weighted* Euclidean metrics where the i^{th} feature is biased to influence the similarity. Finding appropriate weights will require a series of exemplars to act as a training set where a function of the variance of visual features is used (33). For the implementation of a prototype, linear weightings are sufficient to demonstrate the creation of video-objects. Support for weighted meterics will be a future extension to the prototype.

The algorithms for SCD-Object correlation are as follows:

Add results of object detection SCDs to Buffer
for bufferIndex = $B_{max} - 1$ to 0   copy contents of Buffer[bufferIndex] to Buffer[bufferIndex+1] copy latest results to Buffer[0]

```
Correlate objects across buffers
for IFD = 0 to IFD_max (where IFD_max < B_max)
    bufferIndexIFD = IFD+1
    if( no shot-break in Buffer[0] to Buffer[bufferIndexIFD-1] )
        Link objects between buffers( 0, bufferIndexIFD )
```

```
Link objects between buffers( startIndex, endIndex )
clear object stack S
push all objects in Buffer[startIndex] on to S
while( S is not empty )
    O_newer = pop S
    O_older = Find object in Buffer[endIndex] that best matches O_newer
    if( O_older is valid )
        if( O_older is already connected to an object O_Xnew )
            push object O_Xnew to S
            disassociate O_Xnew with O_older
        connect objects O_newer and O_older
```

```
Find object in Buffer[endIndex] that best matches O_newer
O_best = invalid
Similarity_best = very large number
for all objects O_old in Buffer[endIndex]
    if( O_old is already connected to an object O_Xnew )
        if( IFD == IFD of O_old to O_Xnew )
            if( Similarity(O_newer, O_old) < Similarity(O_Xnew, O_old) )
                Similarity_best = Similarity(O_newer, O_old)
                O_best = O_old
    else
        if( Similarity( O_newer, O_old ) < Similarity_best &&
            Similarity( O_newer, O_old ) < Similarity^IFD_Threshold )
            Similarity_best = Similarity( O_newer, O_old )
            O_best = O_old
return O_best
```

The algorithms describe the processes to link an SCD-Object in one frame to the best matching SCD-Object in another frame. Correlation is instantiated by calling Correlate objects across buffers, where each SCD-Object in Buffer[0] is linked to an SCD-Object in Buffer[X]. The linking operation attempts to connect SCD-Objects together such that each SCD-Object satisfies the condition: an SCD-Object is connected to the most similar available object in another frame. This requires linking and unlinking of SCD-Objects as the process of minimising the dissimilarity iterates.

The process of linking SCD-Objects across frames connects continuous objects together, and can recover missed detections (e.g. from object occlusion) by interpolating the position, shape coordinates, and orientation. The results of correlation at this stage produce motion vectors from one object in one frame to the next.

Invariably, there is a trade-off between speed and accuracy. Experiments have shown that in some frames, missed detections, false positives and spurious

detections can occur. Linking objects over time increases the likelihood that an object really exists for a particular frame. After the objects have been correlated, a filtering process eliminates false positives and extracts objects that appear in at least L_{min} frames. The results represent continuous semantically labelled entities which are submitted to the next stage (the SCD Manipulator) for further processing. The rules of the filtering stage are executed in the following order:

1. If object, O, has no connection with an older object but it exists in a connected state in at least L_{min} frames then declare, O, as a new instance of semantic type labelled by SCD, and submit for further processing.

2. If object, O, is connected to an older object that is connected to a new instance then submit for further processing.

Since the correlation process submits objects to the next stage provided that it is connected to at least L_{min} frames, any object which is connected by a frame count of less than L_{min} frames is 'automatically' rejected.

To summarise this section, the SCD Correlator has three main functions:

- A buffer to store a snapshot of video
- Store the results of the snapshot
- Create video-objects

The results of transforming video images into a computer representation are stored as SCD-Objects. The creation of SCD-Objects is discussed in the next section.

3.4 Specific Content Detectors (SCDs)

Steven Pinker (52), a cognitive psychologist, describes the brain as a collection of modules, each of which specialises in a specific task. This section introduces

a similar idea where video is processed for fine-grain semantics using specialised modules called *Specific Content Detectors*(SCDs). Visual analysis in the VCR System is initiated by calling a single SCD that is constructed from a set of SCDs each of which is tailored to detect object-level entities in video. The SCD makes appropriate calls to other SCDs to retrieve the information it needs to complete a task.

Firstly, the concepts behind the SCDs must be established before a technical description of how they operate within the VCR System is given. An SCD is simply a blackbox that takes input from the Sensors and other SCDs, and generates an output that has semantic significance, e.g. object-level entities, and indication of shot breaks, in the form of SCD-Objects. On a grander scale, an SCD is a building block and a processing element of a larger framework constructed from other SCDs. The construction of an SCD is tailored for a particular video context, e.g. an SCD may detect anchor-persons in news video, and the VCR System assumes the given video is of the correct domain.

The video context constrains the system such that the visual processing operations within the SCD can make assumptions to 'simplify' the 'recognition'. Detectable visual features known to exist for a particular video domain can be labelled with a semantic definition. A hypothetical example of detecting single anchor persons in news video using an SCD framework is shown in Figure 3.6.

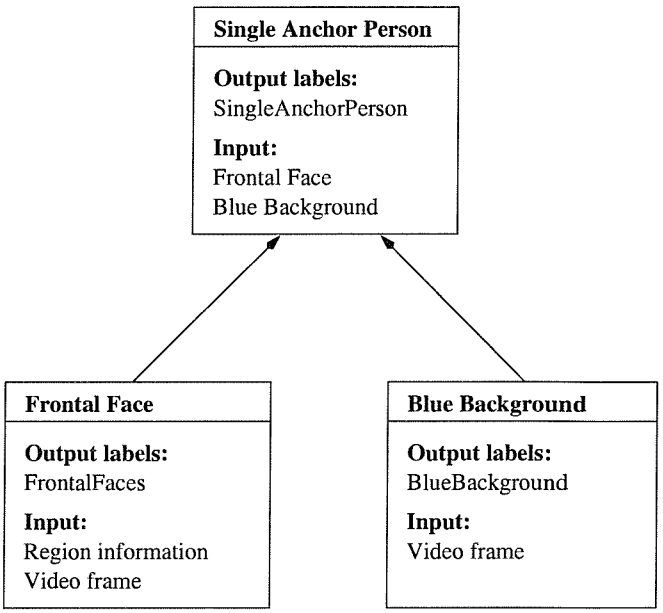


Figure 3.6: An example construction of a *Single Anchor Person* SCD.

The example may seem tentative but given the video context is news video then the SCD can function usefully. The context of the SCD construction operates within a 'sub-context' of the grand genre of news video. The power of SCDs is realised when multiple sub-contexts are connected to form a more complete picture of news video. Figure 3.7 extends the current example to include double anchor-persons as a demonstration of the idea to scale to multiple sub-contexts.

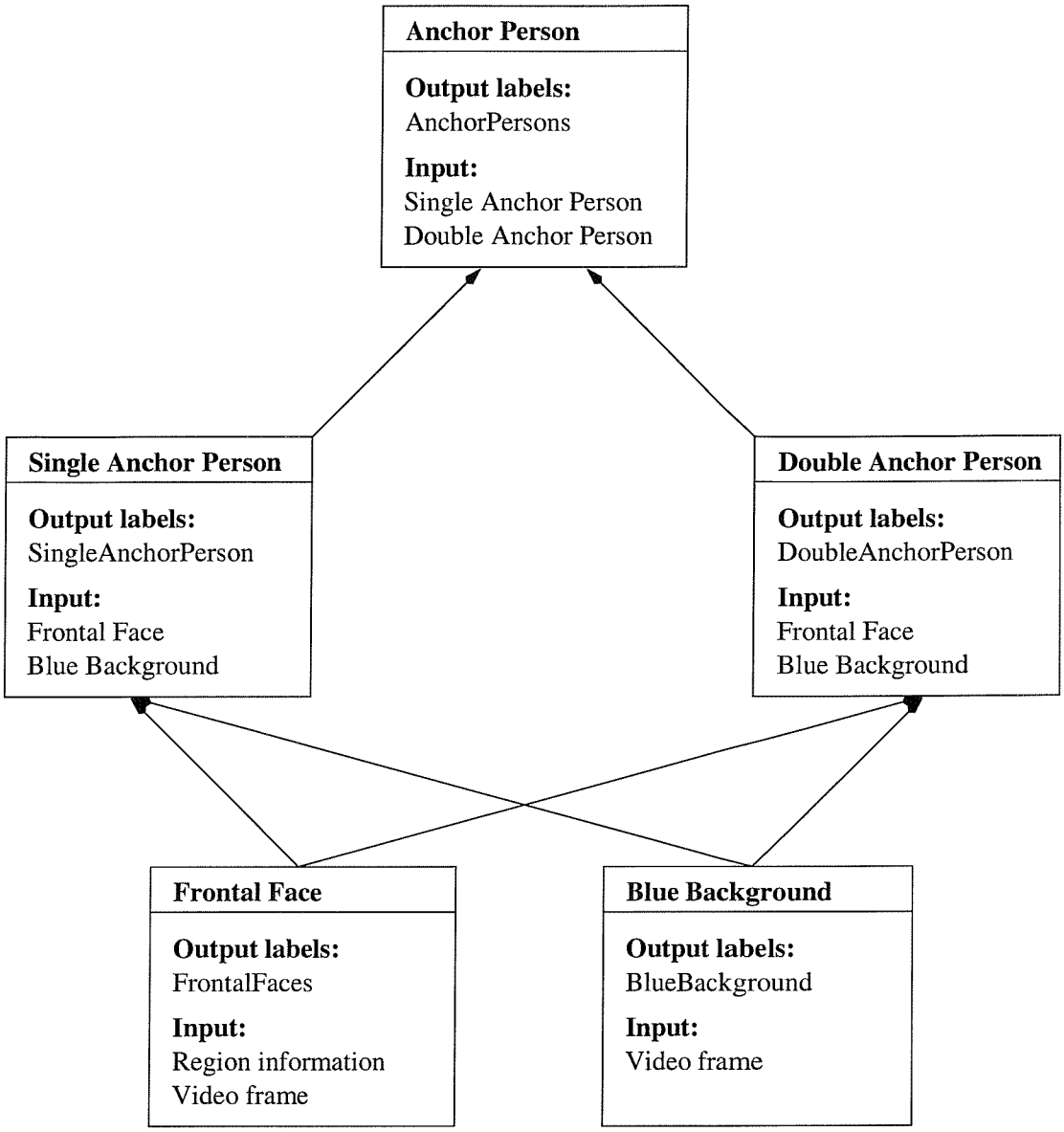


Figure 3.7: An example construction of an anchor person detector using two SCDs where one detects single anchor persons and another detects two anchor persons.

The newly introduced SCD Double Anchor Persons uses the same SCDs (Frontal Face and Blue Background) as the Single Anchor Person SCD. The advantage of this method when compared to other methods that detect anchor-person shots (78), is that we have semantic handles to the actual content that gives more

power for video applications such as querying. Suppose the current anchor-person SCD construction is extended to include a face recognition and a face location SCD, the query: "Retrieve all anchor-person shots with the face of Miss X" is possible. The face detection SCD locates the faces, and the detected face SCD-Object is passed to the face recognition SCD for analysis. One of the main tasks of intelligent interpretation of sensory data is object classification and labelling. This is usually accomplished using statistical pattern recognition techniques or neural networks.

The structure of a SCD element is shown in Figure 3.8. It is composed of three sub-systems: a *Vision Processing Unit*, a *Semantic Packaging Unit*, and (optionally) more SCDs.

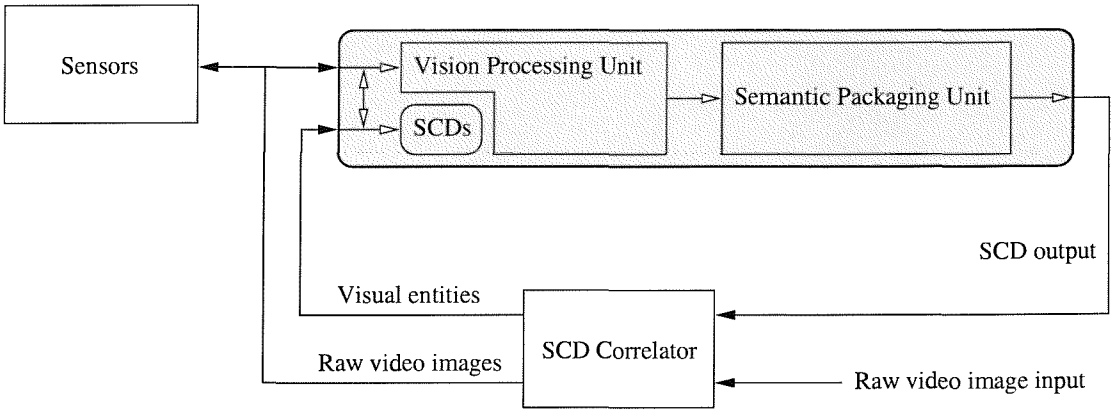


Figure 3.8: The shaded area indicates the structure of a Specific Content Detector. It is composed of a Vision Processing Unit (VPU), a Semantic Packaging Unit(SPU), and a recursive *sub*-network of SCDs. Communication between SCDs is routed through the *SCD Correlator*.

The Vision Processing Unit (VPU) processes data generated by the Sensors and SCDs for semantic-level entities, e.g. objects, and the results are passed to a Semantic Packaging Unit (SPU). The SPU encapsulates the results from the VPU in a wrapper such that all SCDs outputs have a common data format (the SCD-Object), and thus, enabling other SCDs to be compatible with each

other. The outputs of all SCDs are routed to the SCD Correlator where it acts as communication conduit for SCDs and storage for SCD-Objects.

The figure shows an SCD encapsulates a set of SCDs making the definition of SCDs recursive and can be expanded to form a tree-like structure. The recursive definition of an SCD binds its 'internal SCDs' to a context, see example in Figure 3.7, such that the vision analysis in the internal SCDs can make assumptions to simplify content recognition.

In Sections 3.4.1, 3.4.2, and 3.4.3, the VPU, SPU, and the internal SCDs will be presented in detail respectively.

3.4.1 The Vision Processing Unit

At the heart of an SCD is a hand-crafted Vision Processing Unit (VPU). The nature of a VPU processes SCD and Sensor outputs to detect higher-level visual entities than the basic features detected by the Sensors. The higher-level visual entities may be real world objects or parts of real world objects such as faces. Image content detectors are commonly represented as feature vectors, whose elements correspond to significant parameters that model image attributes. Techniques used in a VPU to detect content are not restricted to a single *general* method, but rather multiple or established techniques can be used, e.g. neural, analytical, algorithmic, ad hoc.

The focus of the VPU is to use simple and relatively fast vision techniques. The implementation of a VPU to recognise objects does not impose a requirement of 'explicit' recognition, but rather the detection of context invariant features or signatures that can be used to infer objects. For example, given that the context is a top-view of a snooker table, then an SCD can be implemented to assume all green areas that satisify a certain geometric criteria must be snooker tables.

The operation of a VPU reads the Video Frame Pipeline in the SCD Correlator for video images, and analyses them for content. The VPU is tailored to detect specific types of content that can be given a semantic label, e.g. objects,

or indications of shot breaks. To aid the implementation of a VPU, the Sensors provide a set of general purpose image processing algorithms where video frames can be processed for syntactic content.

For all 'object-detecting' VPUs a mathematical certainty factor is assigned to each detected object which indicate the confidence that the objects exists in a frame. Objects are only declared to the next stage of processing if and only if the confidence of existence exceeds a pre-determined threshold. For objects that are passed on to the next stage their calculated probabilities are stored should it be required by methods that analyse the detected objects. The calculation of probabilities for detected objects varies with the video analysis method and is likely to result in non-correlated probabilities with other VPUs that do not use the same probability scheme. However, the confidence of detected objects is reinforced by the SCD Correlation, where false positives are eliminated.

Given that a VPU succeeds in detecting content, packaging the results in terms of an SCD-Object is the next stage of processing. The next section presents the Semantic Packaging Unit (SPU).

3.4.2 Semantic Packaging Unit

The Semantic Packaging Unit (SPU) represents the last stage of processing within an SCD. Modularising video analysis techniques into SCDs requires a common communication protocol such that SCDs can be developed independently from one another. The job of the SPU creates SCD-Objects that are best representative of the output from a VPU.

The implementation of a SPU calls a list of methods that calculate values to fill-in the attributes of a newly created SCD-Object. The methods are tailored to the video analysis techniques used in the previous stage. To complete the processing cycle of an SCD, the resulting SCD-Object(s) are stored in the SCD Results Pipeline of the SCD Correlator.

3.4.3 More SCDs

This section explains the significance of SCDs within another SCD as shown in Figure 3.8. SCDs are constructed to form a tree-like structure that implements *hierarchical decomposition* of video, where root-node parent SCDs represent more abstract content than their child SCD nodes. The aim of this scheme where child SCDs are called based on the content detected by parent SCD nodes is to resolve video into visual-entities effectively.

The beginning of this chapter stated *"...while the development of a different and better science of machine vision is ongoing, the VCR System looks at an engineered alternative"*. The construction of an SCD using other SCDs is the engineered alternative. The Visual Processing Unit of an SCD aims to find visual entities that can be semantically labelled, and since general object recognition is not possible, inferring from visual features is an available route.

On the general level of video and image analysis, visual features do not have any semantic relevance unless bounded by a context. The context can be determined in two ways: 1) Automatically, i.e. video genre detection (21), or 2) Assumed, i.e. constrained video analysis systems. To give a clearer explanation of how multiple SCDs process video for content, another example will be used.

The example is a system that detects snooker balls from top-views of snooker tables. An SCD, LocateSnookerBalls, searches for snooker balls by locating circular shaped regions. However, to reinforce the confidence that SCD-Objects generated from LocateSnookerBalls are snooker balls, an SCD to detect the presence of a snooker table (LocateSnookerTable) is required. LocateSnookerTable will be the parent node of LocateSnookerBalls, and thus, LocateSnookerBalls will be executed if and only if snooker tables are present. A more general analysis of snooker would of course include SCDs to detect other views of snooker action.

In this particular example, the advantage of the hierarchical scheme is that it will save computation time in two ways: 1) LocateSnookerBalls starts processing if and only if LocateSnookerTable detects a table, and it is assumed that

the detection of a snooker table is simpler than the detection of balls, 2) Domain knowledge of snooker dictates that snooker balls are found on snooker tables, and if the VPU of LocateSnookerTable is implemented to find snooker table boundaries, then LocateSnookerBalls can search the bounded area instead of the whole video-image.

The execution order of events in an SCD is shown in Table 3.5. This section completes the explanation of SCDs. However, in this chapter I have mentioned Sensors without covering them in detail. SCDs have two resources for video analysis, their VPU's and data from the Sensors. The Sensors are described in the next section.

```
if( parent SCD satisfies criteria )
    Execute VPU
    Execute SPU
    Call child SCDs
```

Table 3.5: Execution sequence of constituient SCD components.

3.5 Sensors

It is said that the world is the way it is because of the way we perceive it through our senses. Raw video is fundamentally a sequence of images each of which is a sandwich of planes of ordered pixel intensities. The conversion of pixels into a higher-level form provides the initial handles for more complex analysis. The creation of these handles usually involves an intermediate step, where the physical properties of an image are extracted, i.e. its syntax.

In the VCR System, the Sensors are a shared pool of video analysis resources that is accessible through the Specific Content Detectors (SCDs). The Sensors receive video images or parts of video images from the SCDs and processes them for syntactic properties. Although SCDs contain resources for vision processing, a distinction is made between the Sensors and the SCDs. Frequently used or

generalised video operations are accessible through the Sensors, providing an efficient system where SCDs access a single copy of code for a particular operation.

The Sensors currently include the following functions to process images:

- Segmentation of images into regions.
- Extraction of regions.
- Image scaling capabilities.
- Edge detection.
- Histogram functions.
- Functions to calculate mean and variance.
- Thresholding.
- Colour operations, e.g. greyscaling, and colour reduction.

Usually, the list of functions in the Sensors are increased by developing code in the VPU before it is transplanted to the Sensors. One could say that the Sensors can be implemented as SCDs, but there are advantages from segregating 'low-level' video analysis code (the sensors) from code which extract semantics that is contained in the SCDs . It is viewed that the separate containment of image analysis functions (collectively called the Sensors) will provide efficient development of SCDs. Tested image analysis functions are made available for all SCDs via the Sensors, thus reducing redundant code which may reduce the size of SCDs. More specialised code is thus contained within the SCDs which enables proper management of code for the implementor of SCDs.

3.6 SCD Manipulator

The beginning of this chapter visualised the VCR System as a combination of two systems: a vision processing system and a rule-based system. The vision

processing aspect of the VCR System is represented by the SCD Correlator, the SCDs, and the Sensors. The rule-based component is the Specific Content Detector Manipulator (SCD Manipulator). This section will describe the purpose of the SCD Manipulator, and the current implementation in the VCR System.

The SCD Manipulator is a term given to the encapsulation of the procedures that process SCD-Objects for semantic information. In the VCR System the SCD Manipulator is viewed as a black-box that provides 'hooks' for a *Manipulator Core* to access the SCDs and to receive SCD-Objects, see Figure 3.9. The Manipulator Core analyses SCD-Objects for semantics which in turn can access 'other' SCDs to complete its task. The output of the Manipulator Core is currently limited to conclusion statements about the video, e.g."Face detected at location (x, y).". The content of the text messages is dependent on the application. In an ideal situation an embedded logic engine is used to implement the Manipulator Core where a knowledge base with the associated rules may be loaded. The use of an embedded logic engine is covered in this chapter.

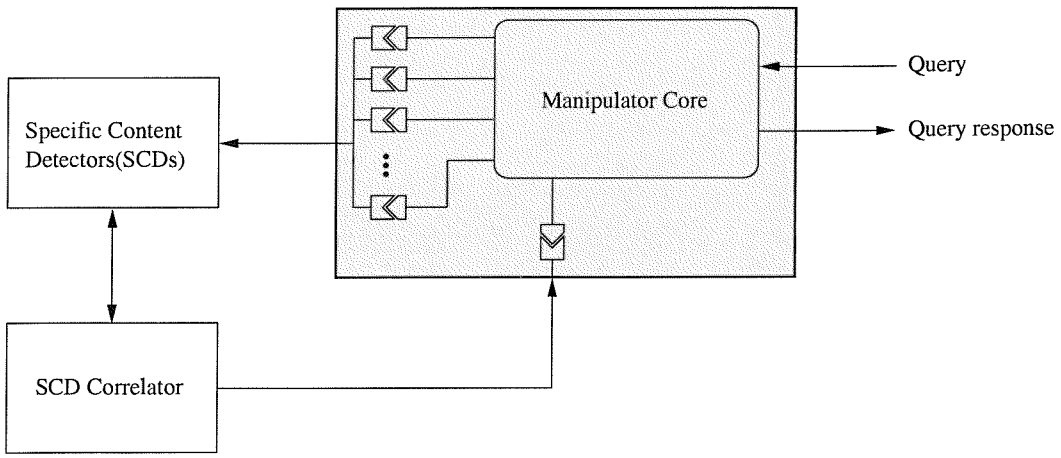


Figure 3.9: The shaded area is the SCD Manipulator, and provides an interface for the Manipulator Core. The Manipulator Core can be implemented by various means, e.g. using a logic engine, schemas etc.

The design philosophy of the VCR System allows the Manipulator Core to be implemented using any method that conforms to the protocol of SCD Manipulator, e.g. if-then-else contructions, or an embedded logic engine. The semantic

identities of SCD-Objects enables the Manipulator Core to process for higher-level semantics, e.g. the interaction of objects. The Manipulator Core is also responsible for input and output to the user. The SCD Manipulator provides interfaces that impose implementation constraints on the Manipulator Core, where it is restricted to the following operations that allow it to communicate within the VCR System:

- Turn an SCD on or off.
- Ask an SCD to analyse given SCD-Object(s).
- Receive SCD-Object(s).

The current version of the Manipulator Core is hardcoded for applications using a set of hand-crafted if-then-else constructions. The initial point at which a construction is invoked to generate a response is analoguous to a query. Hard-coding queries may appear to be a limitation. However, systems that process specific types of video for content detect a limited number of events (62), and do not have a need for dynamic querying. From a different perspective, the SCD Manipulator can be viewed as a meta-data generator for dedicated video indexing and retrieval systems, this is demonstrated in Chapter 5.5.4.

The advantages of an embedded logic engine cannot be ignored, and some work carried out to conceptualise the procedures is demonstrated as an example of how a Manipulator Core might be implemented, e.g. using Prolog. Prolog is a programming language for symbolic computation, and is well suited for solving problems that involve objects and relations between objects (7). Prolog enables a convenient mechanism for dynamic queries and expressing rules. Jones and Batchelor (35) use symbolic based vision analysis for the interpretation of images. Their work is centred around an evolved Prolog interpreter called PIP (Prolog Image Processing). The PIP system differs from the Manipulator Core in several ways:

- Does not support video-objects.

- The PIP system manages the complete pipe-line of vision analysis from grabbing the image to interpretation via a program.

The example Manipulator Core implemented in Prolog assumes the following SCDs: Frontal Faces and Frontal Face Recognition. Frontal Faces analyses video full-frontal faces, and Frontal Face Recognition performs face recognition on SCD-Objects generated by Frontal Faces. A Prolog program loaded in to the Manipulator Core communicates with the VCR System using the following predicates:

- scdswitch(+Scdname, Bool), This switches an SCD on or off.
- scdlook(+Scdname, +ScdObject), Asks an SCD to analyse an SCD-Object.
- scdget(-ScdObject), Retrieves an SCD-Object from the SCD Correlator.

The following clauses represent the program that uses the above predicates:

```
isaface(X) :- scdtitle(X, "FaceLocate").
face(Y) :- scdget(X), isaface(X), scdgetname(X, Y).
```

When the VCR System is running the example query face("Johnnie Cage"), an action is generated when a face is found and is recognised as "Johnnie Cage". This example gives pointers for future work, and elaborate examples are demonstrated in the next Chapters 4 and 5.

3.7 Summary

This chapter presented the VCR System which converts video into object-level entities for the purposes of providing concept level access to video data. The VCR System is composed of four main components: the SCD Correlator, the SCDs, the Sensors, and the SCD Manipulator.

The conversion of video into objects begins at the SCD Correlator which contains two queues, one to store video frames and one to store detected object

entities. Video frames are shifted through the SCD Correlator and each frame is analysed by the SCDs which use a common pool of image analysis functions collectively called the Sensors to aid in the extraction of object-level content. The results of the SCDs are called SCD-Objects and are stored in the SCD Correlator.

The SCD Correlator links 'object-level' SCD-objects across the stack and assigns a unique value to each string of SCD-Objects. The unique identifiers are temporally invariant variables with which the SCD Manipulator can reference detected objects. The SCD Manipulator is a hub that routes communication between a Manipulator Core and the SCDs. The Manipulator Core is responsible for user input and output, and is used to 'interpret' the detected SCD-Objects. The Manipulator Core may be implemented in a variety of ways, for example such as an embedded Prolog engine or as if-then-else contructions.

The discussion in Chapter 2 supports the decomposition of video into objects if the type of video input is a known a-priori. The SCDs are constructed to suit a particular type of video which may be inter-changed with other SCDs that are suited to other types of video within the VCR System. An SCD is a scaleable component, and different domains are handled by partitioning them via a 'root-node' SCD whereby methods to deduce the context are used to switch on the appropriate 'sub-SCDs'.

The next Chapter introduces the processes involved in the implementation of certain types of SCDs which may be used for general purpose video analysis. Experimental results are presented and the avenues for possible applications are stated.

Chapter 4

Object Oriented Video Analysis

The previous chapter introduced the idea of the VCR System as a new platform for video analysis. The basic ideas of the VCR System are that video is analysed for semantic content by a hierarchical set of 'visual experts' called Specific Content Detectors (SCDs). Semantic content is represented in terms of SCD-Objects which are temporally resolved to form video-objects. SCD-Objects are also used to indicate occurrences of events. Using a rule-base or symbolic schema, video-objects may be manipulated and processed for higher-level semantics.

The purpose of this chapter is to present the stages in the development and implementation of general purpose vision SCDs. To demonstrate the working principles of the VCR System the following general purpose SCDs are introduced:

- A face locating SCD.
- A scene break detector SCD.
- A colour classification SCD.

The organisation of this chapter begins with the construction of the face location SCD, and this is then followed by the scene break SCD, and finally the

colour classification SCD in Sections 4.1 4.2 and 4.3 respectively. Following the presentation of the SCDs, a demonstration of how the SCDs function within the VCR System is presented in Section 4.4. This chapter is then summarised in Section 4.5.

To begin, the face location SCD is described in the next section and is organised by first introducing the problem, reviewing the published techniques of face detection, and its implementation in the template of an SCD.

4.1 Face Detection SCD

Detecting the presence of human faces can provide important cues for many image and video analysis tasks (12). Face detection and location is used as a prerequisite for face analysis tasks such as recognition (70) (45) and expression interpretation (6). The capability to generate high-level semantic information from isolated faces may be applied in many applications such as face authentication, indexing and retrieval based on face objects, and video analysis in the context of the VCR System.

Implementing an SCD to detect faces is non-trivial when faces encoded in visual data can appear in any pose, position, orientation, and scale. The task is further compounded by problems associated with illumination variation and noise which is inherent in video streams. However, the purpose of this section is to give the reader an insight into the implementation of *object-detecting* SCDs by presenting the developmental stages of an SCD which analyses video for faces.

This section begins with some background in face detection (Section 4.1.1). Some of the most robust available techniques for face detection are computationally intensive, and are ill-suited for some applications of video analysis because of the undesirable length of time it takes to process a single frame of video. The aim of Section 4.1.2 is to present a pre-filtering technique which can identify, relatively quickly, regions in an image or video frame likely to contain human faces. The process of integrating the pre-filter into an SCD is presented in Section 4.1.3,

and finally Section 4.1.4 gives the summary.

4.1.1 A Survey Of Face Detection Techniques

The approaches to detect faces generally fall into these categories: Face appearance models, Facial colour detection, and Neural methods. A survey of these classifications are given in Sections 4.1.1.1, 4.1.1.2, and 4.1.1.3, respectively. A summary of these techniques is given in Section 4.1.1.4.

4.1.1.1 Face Appearance Models

The use of the term *face appearance models* covers profile matching, shape analysis, and statistical modelling of features. These techniques generally process images with the prior knowledge about faces. The integration of face information which allows adequate modelling of facial features may give clues to face locations.

Chow *et al.* (13) attempt to locate faces by using simple heuristics to locate the spatial arrangement that resembles a face from a segmented face image. Their method classifies facial features such as the eyes, eye brows and mouth. These features are usually darker relative to their respective surroundings and can be located such that an approximate location of the face can be hypothesised. A face is confirmed by detecting the facial features within the hypothesised location. Attributes such as shape and position of the features are taken into account for an accurate assessment. The technique assumes faces are on a simple background. This limits the use on video sequences where typical real-world face scenes are present with complex backgrounds. But in spite of this, the advantage with this method is its speed in processing face images and this makes it a favourable candidate for video processing. Despite its detection success rate of 96%, the conditions that must be met are not typical of videos.

Samal *et al.* (59) derive sets of face silhouettes using principal component analysis and then uses a modified Hough-like technique to detect faces. Images

are pre-processed before face detection is performed. The steps that are involved in the pre-processing stage are edge detection, edge thinning, and thresholding to remove unwanted edges. Selected face silhouettes are used to locate faces in images. They report a success rate of 95%, however the use of silhouettes limits the application to the same order as Chow *et al.* . It is conceivable that this may be inappropriate for complex scenes. Video sequences may not have the necessary definition that is required for this technique. Kwon *et al.* (38), locates faces for age classification from facial images by fitting ovals as an approximation to face silhouettes. Both techniques are not suitable for video sequences.

Luo *et al.* (43) presents a human face location technique based on contour extraction within the framework of a wavelet-based video compression scheme for video conferencing applications. Video conferencing applications transmit video and audio information over a network. These applications demand a continuous high frame rate and good quality pictures. Currently, most networks are unable to provide the bandwidth necessary for such transmissions. A solution is to compress the video before transmission. Their motivation to locate faces for video conferencing applications is to improve the perceptual quality of images especially where facial expressions are concerned. By locating the face, a better coding scheme can be applied to that area whilst the *background* can use a higher compression coding scheme. A face is located by edge detecting a low resolution representation of the original image, and searching for contours that *fit* the shape of a head. Feature points, e.g. joining edges of the jaw and neck, are located along these contours and a minimum enclosing rectangle encompassing these points represent the location of the face.

Dai *et al.* (18) models a face as a texture and can detect them in cluttered colour scenes where the size of faces is comparatively small. Hypotheses of locations of faces are generated from colour information so that each location can be evaluated for a face texture. They found that the I component of YIQ colour representations of regular RGB colour images represented enhanced facial regions. A

face texture is represented as a set of inequalities derived from a space grey-level dependence (SGLD) matrix. A SGLD matrix is used in textural feature analysis, and a brief overview of this method is presented in their paper. Their method appears to be very effective. However this relied heavily on the fact that most East Asians have yellow skin colour. Work in detection of faces based on colour is presented in the next section.

4.1.1.2 Face Colour

Lee *et al.* (39), use motion and colour as their primary visual cues to segment a face from a complex background. Predefined colour information is used to locate facial features. Their system uses natural head movement to isolate a large section of a persons face and maps an elliptical cropping area over a thresholded motion field. The segmented image containing the head of the subject is projected into HSI colour space where facial skin, eyes, eyebrows, and mouth form localised clusters and can be easily distinguished. Using a heuristic algorithm that relies on prior knowledge of the human facial structure the system can locate eyes, eyebrows, and mouth.

Chen *et al.* (12) use colour characteristics to detect human faces in colour images against complex backgrounds. A neural network is used to isolate areas of trained skin colour in a given colour image. These form candidate face regions where filtering and segmentation procedures are used to *improve* the quality of these regions. They use lips to verify each candidate region to confirm the existence of a face. A CIE-xy colour distribution graph of the lips produces a fairly distinct cluster. This is used to colour segment lips from the candidate face region. A region growing technique using pixel aggregation is applied to segment a more accurate lip region. They use an *energy-thresholding method*, where the characteristics of lips, including their shape, edge and colour are formulated into three different energy terms. Thresholding the sum of these energy terms determine whether pixels belong to a lip. The new probable lip region is further

scrutinised by three shape descriptors: the length/width ratio of the region, the ratio of the area of the lip and the minimum enclosing rectangle, and the curvature of the lip. A lip is confirmed when the empirically determined descriptors fall within certain numerical ranges. The system detected between 76 and 96 percent of faces in their test sets. They report an average processing time of 4 seconds per image (378 x 252 pixels).

4.1.1.3 Neural Methods of Face Detection

A different approach to face detection explored by Rowley *et al.* (56), uses neural techniques. The neural network searches for face images by visiting every location of the input image. Their system handles faces that are larger than the input window of the neural network by repeatedly sub-sampling the input image, and repeating the process. This scheme is computationally expensive and is not suited for real-time applications even though the detection rate is between 78 and 90 percent.

Juell *et al.* (36) used four neural networks in a hierarchical structure to determine whether a face exists at a particular location. Three of the neural networks were trained to recognise either a mouth, a nose or eyes, and are called *child* level neural networks. The fourth neural network, the *parent*, recognises a face structure from the spatial outputs of the child neural networks. Images were edge enhanced and copies were fed to each of the child neural networks. The neural networks scan the entire edge images and produce binary outputs at specific locations, indicating detected features specific to the neural networks. The parent neural network convolves over the child level neural outputs to locate faces. They report that their neural networks exhibit a tendency towards false positives and this was particularly severe when noisy images were processed.

4.1.1.4 Summary

The papers reviewed present a mixed variety of techniques each of which have their advantages and drawbacks. The techniques that model face colour tend to have reasonable processing times. However, neural methods have provided consistent and accurate results, but those presented are too slow to be applied in near real-time video analysis. In the context of the VCR System, an SCD that detects faces as objects must satisfy the following requirements:

- Be able to detect more than one face.
- Find faces on complex backgrounds.
- Provide isolation and confirmation of faces at a reasonable speed.

In accomplishing the stated conditions, a hybrid approach where the best face verification technique is combined with a filter which rapidly identifies potential face locations is presented in the next section.

4.1.2 A Prefilter Enabling Fast Frontal Face Detection

One of the common problems of processing video is speed. Video is a high-band width format, and the detection of faces in video sequences must be achieved with reasonable performance. To this end, our contribution to the speed up of face detection is described in Chan *et al.* (10). A pre-filtering technique rapidly identifies locations in video frames where preliminary evidence suggests a face may be sited. A more elaborate and established technique is then used to confirm or deny the existence of faces at these locations. The pre-filtering technique is based on the fact that, for frontal illuminated faces, the eyes are usually a prominent feature of the face (47) (53) (75). They have a spatial distribution that is roughly related to other facial features such as the nose and mouth. The distance between a pair of eyes gives an indication of the size of the face, and the positions of the eyes can be used to estimate the orientation.

Using this premise, the technique generates regions that are most likely to contain faces. These regions are then verified, in turn, to test whether a face actually exists. Generating these regions relies on detecting possible pairs of eyes (eye-pairs) that may or may not belong to a face. The eye-pairs inherently provide information about the location, orientation and scale of potential faces. Square regions around the eye-pairs are used to establish the area that may contain the rest of the face. These areas are rotationally normalised so that they represent possible upright faces. A suitable face verification technique can then be used to verify that the captured areas actually contain images of faces. The current system uses a neural network for the final face verification stage and is based on the approach of Rowley *et al.* (56). Figure 4.1, illustrates the individual stages of the pre-filtering process and face verification.

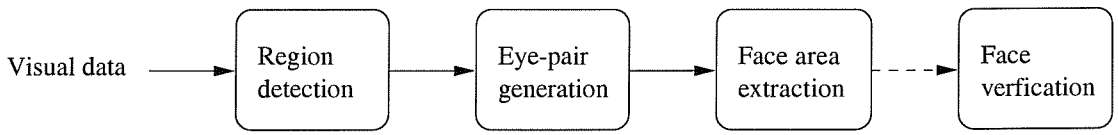


Figure 4.1: The stages in isolating regions that may contain faces.

Before the technique of face detection is described, an introduction to neural networks is given in Section 4.1.2.1. The details of the technique are presented in the following order: Sections 4.1.2.2, 4.1.2.3, 4.1.2.4, and 4.1.2.5 describe region detection, eye-pair generation, face area extraction and face verification respectively; Section 4.1.2.6 reports results of some experimental work; and Section 4.1.2.7 gives the conclusions and future work.

4.1.2.1 Neural Networks in the Context of Visual Analysis

This section introduces the basic concepts of neural networks in the context of visual analysis. Neural networks are based on the parallel architecture of biological brain cells and are simulated on a computer to 'solve' problems. They are useful when algorithmic solutions cannot be formulated and they can be used to

find structure within existing data sets.

In the case of visual analysis neural networks are commonly used as pattern classifiers where patterns or images are processed for 'recognisable' features. Figure 4.2 illustrates a typical neural network that recognises visual stimuli. The architecture of neural networks generally has an input layer, a hidden layer and an output layer, and is fully interconnected in-between each layer by a set of connections. The strength of each connection is determined by a weighting value.

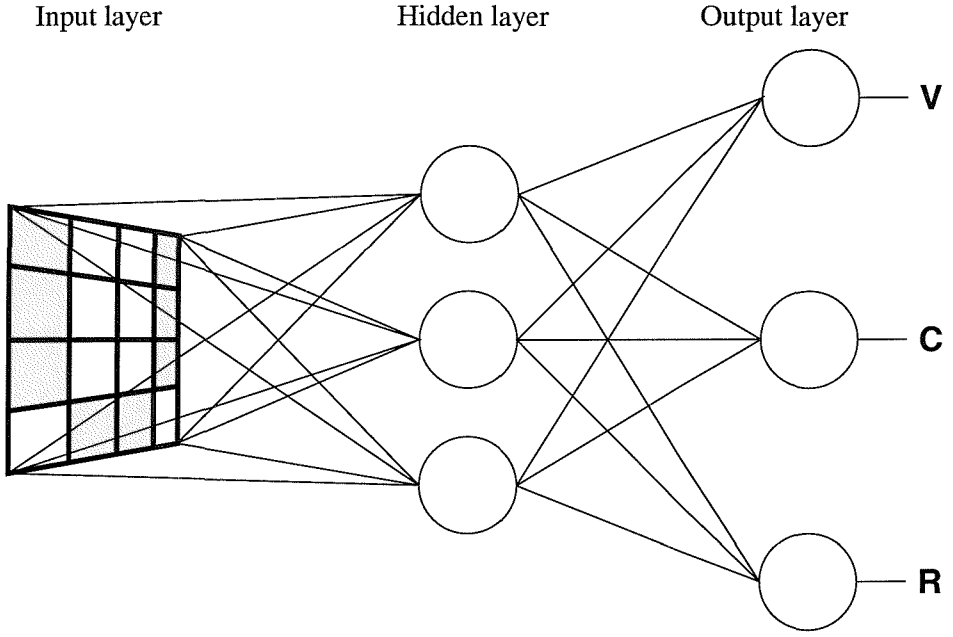


Figure 4.2: An example of a neural network architecture that performs optical character recognition. The input layer in this example is arranged in a matrix of sixteen nodes fully interconnected to the three node hidden layer. The hidden nodes are fully interconnected with the three node output layer which is 'semantically' labelled with characters.

Each layer consists of a set of nodes (neurons) which only have connections to every neuron in the next layer. A neuron receives input from connections in the previous layer and outputs a response that is weighted by its output connections to other neurons. Determining the proper weightings for the neural connections is achieved by training. A training algorithm called the Back Propagation Learning Rule is commonly used to train neural networks. References to this may be found

in (58).

Before neural networks can be used they must be trained from a set of representative examples together with their corresponding desired outputs. In reference to Figure 4.2, training the neural network to associate an image of a 'V' to the output node V requires examples of 'V' patterns. Input values range between 0.0 and 1.0. Correspondingly, the output nodes are 'clamped' with predetermined values (V= 1.0, C= 0.0, and R= 0.0).

Once an exemplar input pattern is clamped with its corresponding outputs the training begins. The training is executed by a training algorithm where it alters the weightings of the interconnections to influence the input values to generate the given outputs. The whole procedure of preparing an input pattern, clamping its intended output, and applying the training algorithm is repeated for other patterns belonging to a carefully chosen training set.

A properly trained neural network pattern classifier should in theory be able to map similar images to a representative 'semantic' output node, e.g. a noisy image of a 'V' should map to the V node in the output layer. Classification typically begins by seeding each node in the input layer with a function of the input image pixel intensity values. The input layer values are weighted and propagated through the neural network to generate an output. If an input pattern is similar to a pattern used during training then a correct response should be reflected in the output layer.

This brief introduction serves to provide a basic insight into neural networks without the introduction of equations. More information on the subject of neural networks may be found in (2). The method of the face detection begins in the next section.

4.1.2.2 Region Detection

The initial stage of the face detection technique receives an image and segments it into regions. Each of these regions are evaluated in turn to see whether they

satisfy certain criteria pertaining to eyes. The visual input is segmented by remapping pixel values to a voxelised RGB colour-space. Mapping colours of an image to its representative voxel produces homogeneous regions. The segmentation process has complexity of $O(n)$ where n is the number of image pixels, and it is ideal for applications where speed is a concern. It can be efficiently implemented by reducing colour bits in the red, green, blue colour channels for each pixel using bit masks. The implemented system uses only the first significant bit. Dark regions are extracted from the segmented image by visiting every dark pixel (seed pixels) and flood filling surrounding areas that have the same voxel number as the current seed pixel. During the flood filling process, the number of flooded pixels are counted and the extreme coordinates of the fill are preserved.

To reduce the computational complexity in the next stage, each region is evaluated with a set of heuristics that determine whether it could be a potential eye region. The heuristics based on a very broad interpretation of the geometry of human faces and some initial experimentation are as follows, but it should be noted that parameter values are not critical. They are used to eliminate candidate regions which have properties sufficiently different from those of a potential eye region that they may be eliminated.

Definitions:

w and h are the width and height of the segmented image in pixel units.

R_n, where n is the region number in the set of regions R.

$R_n.width$ is the width in pixels.

$R_n.height$ is the height in pixels.

$R_n.aspect$ is the aspect, defined as $\frac{R_n.width}{R_n.height}$.

$R_n.numberofpixels$ is the number of pixels the region occupies.

$R_n.homogeneity$ is a measure of homogeneity, defined as $\frac{R_n.numberofpixels}{R_n.width*R_n.height}$.

1. Elimination of regions that are too small and too large.

   $1 < R_n.width < 0.5w$ and $1 < R_n.height < 0.5h$

2. Regions associate with eyes have a limited range of aspect ratio.
   $\frac{1}{7} < R_n.aspect < 7.0$

3. This criterion determines how much the region covers its minimum enclosing rectangle.
   $R_n.homogeneity > 0.5$

Through experimentation it was found that smoothing the video images to reduce noise produced better results. Smoothing of the video images is achieved by using a 3x3 averaging mask before the region detection process begins. Segmentation of the filtered input produced smoother regions and a reduction of false positive eye regions was recorded in preliminary experiments. Figure 4.3 illustrates an image passing through the stages described in this section.

Figure 4.3: Images displaying the results of each sub-stage during region detection. The filtered regions are indicated by the rectangles in the *Region filtering* image.

4.1.2.3 Eye-pair Generation

The eye-pair generation process attempts to pair regions together that may potentially belong to a face. Given that there are n regions after the region detection stage, the number of statistically possible eye-pairs is $\frac{n^2-n}{2}$. It is desirable to reduce the number of eye-pairs by comparing regions with other regions and using a set of eye-pair heuristics loosely based on the broad geometry of faces. Again, parameters are not critical and were obtained from observations of a wide variety

of images containing faces. The algorithm is as follows:

Definitions:

$distance_x(R'_j, R'_k)$, horizontal distance between the centres of regions R'_j and R'_k.

$distance_y(R'_j, R'_k)$, vertical distance between the centres of regions R'_j and R'_k.

For all possible eye-pairs (R'_j, R'_k):

`if` $distance_x(R'_j, R'_k) > distance_y(R'_j, R'_k)$ `then`

$relative\_width = \frac{R'_j.width}{R'_k.width}$, $region\_aspect1 = \frac{R'_j.width}{R'_j.height}$,

$region\_aspect2 = \frac{R'_k.width}{R'_k.height}$, $sum\_of\_widths = R'_j.width + R'_k.width$

`else`

$relative\_width = \frac{R'_j.height}{R'_k.height}$, $region\_aspect1 = \frac{R'_j.height}{R'_j.width}$,

$region\_aspect2 = \frac{R'_k.height}{R'_k.width}$, $sum\_of\_widths = R'_j.height + R'_k.height$

`endif`

`if` $0.2 < relative\_width < 5.0$ `and`

$k_1 * sum\_of\_widths < region\_distance < k_2 * sum\_of\_widths$ `and`

$0.8 < region\_aspect1 < 7.0$ `and` $0.8 < region\_aspect2 < 7.0$ `then`

`Store` eye-pair (R'_j, R'_k)

The condition $distance_x(R'_j, R'_k) > distance_y(R'_j, R'_k)$ determines if the eye-pair (R'_j, R'_k) is more horizontal or vertical. The reason for having such a condition is that the aspect ratios can be calculated roughly relative to the vertical position of a face, where the width of a region relates to the width of an eye region of an upright face. An input image with a face on its side will have the eye regions with the width being the actual height of the eyes in the image. The term *relative_width* ensures that no two regions have greatly exaggerated size differences, since regions belonging to the same face should not vary by orders of magnitude. Illumination will affect the size of eye regions in the segmentation

process, thus, a range is considered. $region\_aspect1$ and $region\_aspect2$ ensures that the eye regions are approximately in-line with each other. This eliminates eye-pairs with one eye region in a horizontal position and an eye region in a vertical position. The $k_1 * sum\_of\_widths < region\_distance < k_2 * sum\_of\_widths$, where $k_1 < k_2$, ensures that the distance between an eye-pair is not exaggerated relative to the size of eye regions. In this case the sum of widths relative to the upright face position is used to give a measure of the size of eye regions.

4.1.2.4 Face Area Extraction

The resulting eye-pairs possess information that allows rotation and scale invariance of faces. This stage takes each eye-pair and extracts a square region which covers the main facial features (eyes, nose, mouth). Figure 4.4a presents the definition of the square region. Two square regions must be extracted to achieve full rotation invariance. The eye-pairs define an imaginary line between the two squares and both areas on either side must be taken into account. Figure 4.4(b...g), shows a face image and all the captured areas on both sides of the generated eye-pairs. The captured face candidate areas are rotationally normalised.

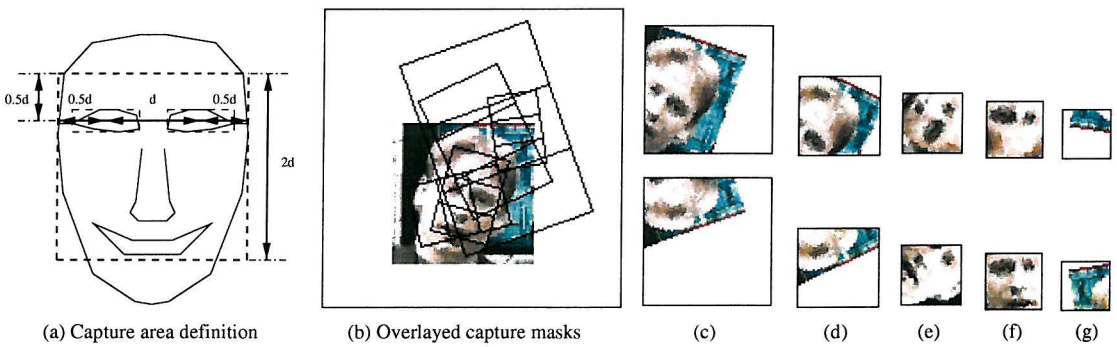


Figure 4.4: An image with captured face candidates based on eye-pairs. The columns of images(c)...(g), show two images captured for each eye-pair.

Our implementation captures face candidates which are rotationally normalised on the fly. This is achieved by scanning pixels parallel to the eye-pairs

and remapping them to an orthogonal grid that is of the same pixel dimensions as the predetermined square capture area.

4.1.2.5 Face Verification

Face candidate images, such as the ones shown in Figure 4.4(c...g), captured in the previous stage present a pattern classification problem for upright frontal faces. A neural network, based on the work by Rowley *et al.* (56), is used to classify each face candidate sub-image. They use a 20 x 20 pixel window moved over the entire image and perform filtering functions to enhance the image viewed by the window, before it is passed to a neural classifier. They pre-process the input image by correcting the lighting and then performing histogram equalisation to improve the contrast. The current system only needs to perform histogram equalisation on the face candidate images because frontally illuminated faces are assumed.

Video stills and scanned images were used to generate a set of training patterns. Training the network used visual data generated from the pre-filtering process where 1375 *representative faces* were manually selected. False positives generated by the neural network were augmented to the non-faces training set, and the network retrained. The number of non-faces used in the training was nearly 1100.

Face candidates are resized to 20 x 20 pixel dimensions, grey-scaled, and histogram equalised, before mapping to the trained neural network and the output is thresholded to give a binary decision, face or non-face.

4.1.2.6 Experimental Results

Our system uses 24 bit colour images or video frames and was developed on a Pentium 133Mhz machine running Linux. Each frame is mapped to a 300 x 300 pixel frame buffer before any processing takes place.

Figure 4.5, shows various frontal illuminated face views, where located faces are signified with a box that also indicates the orientation. When a general

database containing over 240 faces was used the pre-filtering algorithm detected eye-pairs for 92% of the total faces and of these 90% were correctly confirmed as faces by the neural net verifier.

Figure 4.5: Representative examples of faces found by the system. Each image shows 4 numbers: the number of faces in the image, the number of detected faces, the number of false positives, and the number of eye-pairs.

This result was obtained with our initial implementation of the verifier and it is expected that the proportion correctly verified will improve with more careful

training. On average about 21 eye-pair candidates were found per image, which is an important statistic since it is this number which determines the number of applications of the computationally intensive neural net verification algorithm in our approach. The benefits of the approach are clear when it is recalled that, in Rowley *et al.*'s original approach (55), neural nets are applied 193737 times per image with a processing time of 590 seconds on a Sparc 20 although they describe modifications which give a small degradation and a processing time of 24 seconds. Currently our approach is averaging about one image per second on a Pentium 133. Rowley reports detection rates of between 78.9% and 90.5%, and for our test set a detection rate of 83% was achieved. Although a high detection rate has been achieved, the necessary conditions for successful face location may constrain the scope of the pre-filter in general applications. However, one of the important factors for processing video is speed, and we believe that the speed improvement in our approach shows substantial promise when working towards real-time applications.

4.1.2.7 Conclusions

A pre-filtering technique developed for face detection provides an order of magnitude improvement in processing time on the method described by Rowley *et al.* The pre-filtering technique can currently detect 83% of eye-pairs belonging to faces in a test database containing full frontal faces of reasonable size. It is believed that, although parameters in the algorithm are not critical, it will be possible to extend the cases considered in order to improve the robustness of the technique.

4.1.3 SCD Encapsulation

Chapter 3 presented the structure of an SCD which may be composed of a Vision Processing Unit (VPU), a Semantic Packaging Unit (SPU), and further embedded SCDs. This section describes the process of integrating the new technique of

detecting faces into an SCD template, to create an SCD called Face Detection.

Figure 4.6 illustrates the face detection algorithm encapsulated within the VPU of Face Detection. Typically, the VPU of the Face Detection SCD 'listens' for video images from the SCD Correlator and applies the face detection algorithm. However, it is possible for other SCDs to invoke Face Detection with their own 'video images'. The face detection algorithm computes the location, orientation, and an approximate size for each face from a video image, and the data is delivered to the SPU for SCD-Object generation.

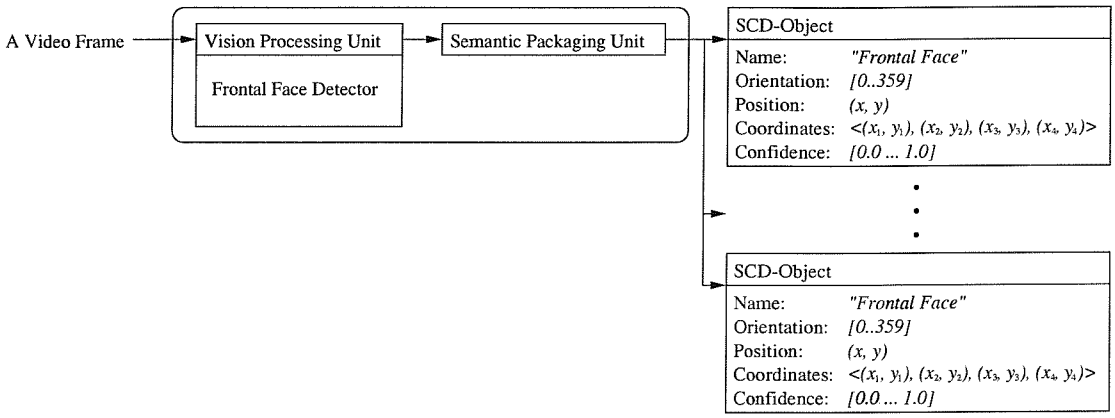


Figure 4.6: Face Detection: An SCD template integrated with a face detector.

Within the SPU, SCD-Objects are created that store information of detected face objects. The orientation and the indicative size are used to calculate four-sided polygons that are relative to their respective face locations, and the positions of the vertices reflect the orientation and face scale. SCD-Objects with the attributes: Name, Orientation, Position, Polygon Coordinates, and Confidence Factor, are assigned with values to represent faces. The values of the Confidence Factors are particular to the techniques from which they are generated; in this case it is the output value of the face neural network. The confidence factor gives a means of comparing confidences for a particular type of SCD-Object, but the current implementation is not appropriate for comparisons from different SCDs.

A single frontal face SCD-Object represents a single face, and for a video

image containing N faces, N SCD-Objects are the expected results. The SCD-Objects propagate from Face Detection to the SCD Correlator where they are stored.

Generated SCD-Objects are assigned with a semantic label that is descriptive of the content and in this case the label, *"Frontal Face"*, is used. However, a video frame may contain more than one face and a semantic label common for all detected face SCD-Objects is inadequate for unique identification, e.g. this is face A and this is face B. The ability to uniquely identify semantically equivalent objects in video is necessary for higher-level analysis. Recalling Section 3.3.1, the VCR System assigns object-type SCD-Objects with unique identities by linking SCD-Objects within the correlator, thus generating temporally invariant objects.

4.1.4 Summary

This section presented a face location SCD that locates faces in video images. The face detection algorithm extracts the location, orientation, and indicative size of faces. These are then used to construct SCD-Objects that form the output.

The Face Detection SCD generates SCD-Objects known as *object-type*. This is where regions within video images are given semantic labels. The next section continues with the development of SCDs, where a different type of SCD-Object classed as an *event-type* is generated. The event-type SCD-Objects described in the next section represent shot-break events.

4.2 Shot Boundary Detection SCD

In video analysis shot boundaries (as defined in the Introduction) indicate the start or end of a shot, and they usually imply a change of content. Shot boundaries not only indicate a possible change of content, but also the frequency of shot boundaries can be used to infer semantic-level content. The computer vision community often regard the detection of shot boundaries as a prerequisite for

video analysis applications. It is generally used to isolate sections of video to create smaller and manageable clips of video.

In the VCR System shot boundary detection is implemented as an indicator of a change in content. An SCD may use shot boundaries as indicators for starting/ stopping and resetting it's internal processes. The main use of shot boundaries in the VCR System is for regulating the correlation process. The next section begins with a brief literature review of shot boundary detection. This is then followed by the details of implementation.

4.2.1 The Detection of Shot Boundaries

Video is generally produced by joining a set of discrete video shots to form a contiguous recording. The observation at the joins of video shots are visual transitions of one shot changing to another shot. These joins are called shot boundaries, and they mark the start or end of a video shot. The transition from one shot to another can be blended with special effects such as fades, and dissolves. The detection of these joins falls under the umbrella of 'shot boundary detection'.

The detection of shot boundaries is fundamental to most kinds of video analysis, e.g. the analysis of television broadcasts, and it is a common prerequisite for applications requiring content-based access to video. The isolation of shot boundaries identifies locations in video allowing it to be partitioned into smaller clips (shots). These shots generally portray a continuous scene in a fixed context. The partitioning of shots into segments of *sub-genres* allows applications to manage, process, and organise, video in an efficient and structured manner. A large proportion of transitions generally fall into the categories (41): hard-cuts, fades, dissolves, and wipes.

The hard-cut as described in the introduction is an instantaneous transition from one shot to the next, and it is the most basic type of production edit. The next section will begin by elaborating on the techniques used to detect hard-cuts.

4.2.1.1 Pairwise Comparison

The Pairwise Comparison is a fairly rapid and simple to implement algorithm for the detection of instantaneous transitions of one frame to the next, e.g. hard-cuts. The general principle of this method is to compute the difference between consecutive frames by comparing corresponding pixels in one frame and the successive frame. If the difference exceeds a threshold then a hard-cut is declared.

The pair-wise comparision is formulated in Equation 4.1, where the frame-to-frame difference of pixel grey-levels exceeding a threshold indicates a scene change.

$$d(i, i+1) = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} \mid g(i, x, y) - g(i+1, x, y) \mid \tag{4.1}$$

Where $g(i, x, y)$ denotes the grey-level of the pixel at (x, y) in frame i, W and H are the width and height of a frame respectively, and if $d(i, i+1)$ is greater than some threshold, T, a scene change is declared.

This method tends to be sensitive to minute changes because digital video may contain scenes that have rapid motion, e.g. camera movements, and may be falsely interpreted as a large change. A method to improve the pairwise method is to compare the average activity of a group of pixels (blocks) (66). Comparing the average activity of each block, as opposed to pixels, dampens the effects of noise and local movements of individual pixels.

Xiong *et al.* (74) describes a method called Net Comparison, where only a part of an image of a sequence is used to determine a scene change, and consequently out performs standard Pairwise and Histogram methods in terms of speed. The initial stage divides each frame into small non-overlapping square areas. A uniform sample of these designated areas are consecutively compared to the corresponding regions of successive frames. The comparison computes the difference, D_1, between the mean values of the grey-level or colour model, and a threshold value, T_1, determines whether regions are changed from one frame

to the next. Hard-cuts are identified when the proportion of regions exceed a threshold T_2. However, the detection of gradual transitions proved to be less robust (73).

Another class of the Pairwise Comparison metric that is documented in the literature (74) (77) is known as the *Likelihood Ratio*. The general methodology of this technique divides a frame into uniform regions, and the mean and variance of each region are calculated. The likelihood ratio of each of the corresponding regions are then computed using the values of the mean and variance. To determine whether the pattern in the regions are associated with the pattern in the successive regions, the corresponding likelihood ratios are thresholded. A frame-to-frame difference is obtained and thresholded to determine a camera break. Equations 4.2, 4.3 and 4.4 expresses this mathematically. This method is more robust for the detection of camera breaks (77), but it is relatively computationally expensive.

$$d(i, i+1) = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} I(i, i+1) \tag{4.2}$$

$$I(i, i+1) = \begin{cases} 1 & \text{if L(i,i+1)} > \text{T} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

$$L(i, i+1) = \frac{[\frac{v_i + v_{i+1}}{2} + (\frac{m_i - m_{i+1}}{2})^2]^2}{v_i v_{i+1}} \tag{4.4}$$

where $L(i, i+1)$ is the likelihood ratio, v and m are the mean and variance and PQ denotes the number of regions in a frame.

4.2.1.2 Histogram Based Methods

The parsing of video based on histograms makes the assumption that a frame is similar to the successive frame within a shot, and the difference gives an indication

of similarity. The detection of hard-cuts based on this premise is almost ideal, however, the metrics used to provide a qualitative measurement must be chosen carefully, and in many cases scene cut detection algorithms usually require fine tuning for optimal performance. A popular and effective metric is histogram differences - the difference between two histograms. The observation that the colour content and intensity within shots do not change as rapidly when changing to a different scene makes this possible in a large majority of cases. Thus, for shots that contain hard-cuts or short-lasting transitions, they can be detected as a peak in the continuous signal of differences between histograms.

There are two basic algorithms in the category of histogram differencing: Global and Local histogram comparison. The global histogram comparison method is expressed in Equation 4.5, where a hard-cut is declared between two consecutive frames if the frame-to-frame histogram difference exceeds a threshold.

$$d(i, i+1) = \sum_{l=0}^{G-1} \mid h(i,l) - h(i+1,l) \mid \tag{4.5}$$

Where $h(i,l)$ denotes the histogram at grey-level l for frame i, and G is the number of possible gray-levels.

Local histogram techniques generate histograms of defined regions within an image. Conventionally, a frame is divided into uniform and non-overlapping regions where a histogram of each region is computed and compared to the corresponding region of the successive frame. Camera breaks are declared if the grey-level difference exceeds a threshold, see Equations 4.6 and 4.7.

$$d(i, i+1) = \sum_{P-1}^{p=0} \sum_{Q-1}^{q=0} H_{dif}(i, i+1, p, q) \tag{4.6}$$

$$H_{dif}(i, i+1, p, q) = \sum_{G-1}^{l=0} \mid h(i,p,q,l) - h(i+1,p,q,l) \mid \tag{4.7}$$

Where $h(i, p, q, l)$ denotes the histogram value at grey-level l for region (p, q) of frame i, and G is the maximum number of grey-levels.

Histograms are commonly used in the process of video segmentation (77) (23) (78), although they are simple to implement, however, more elaborate histogram based techniques are necessary to detect other types of cuts, e.g. fades and wipes. Zhang *et al.* (77) developed a technique called the *Twin-comparison* approach to solve the problem of special effects transitions. The approach adapts a difference metric, e.g. histogram difference, to accommodate gradual transitions. Their experiments have shown that the difference values during a dissolve are slightly higher than those in preceding and following segments of a video sequence. Two thresholds are used where T_b accommodates hard-cuts, and a lower threshold T_s for detecting gradual transitions. If the histogram difference of two consecutive frames, $SD(i, i+1)$, satisfy the condition $T_s < SD(i, i+1) < T_b$, then the ith frame is marked as the potential start position of a gradual change. The end of a gradual change is detected when histogram differences of the potential frame and subsequent frames exceed the upper threshold limit, T_b. Thus satisfying the condition $SD(i, i+n) > T_b$, where n is the number of frames relative to the potential start frame of a gradual transition. Zhang *et al.*, termed this as an accumulated comparison. The values for the thresholds are difficult to set for videos in general, and it would be more reliable if the thresholds varied for a long sequence of frames (37). In addition to these methods, camera motion is classified in an attempt to suppress false positives that could possibly result. A camera zoom, for example, can cause a large number of pixels to change and register as a positive break.

4.2.1.3 Feature Based Methods

The approaches that have been outlined are directly based on the change of intensity metrics and histogramming. This contrasts with a feature based approach where intrinsic properties such as edges are used to determine shot transitions.

Zabih *et al.* (76), observed that edges during a cut or a dissolve disappear at a distance from new emerging edges. The characteristics of transitions are resolved through a metric of dissimilarity that is based on the Hausdorff distance (31). The measurement represents the fraction of changed edges is termed the edge change fraction, see Equation 4.8.

$$\rho = max(\rho_{in}, \rho_{out}) \tag{4.8}$$

The parameter, ρ_{in}, represents the fraction of edge pixels entering the scene, and similarly, ρ_{out}, represents exiting edge pixels. An edge change fraction graph plot of a video sequence reveals signatures that are characteristic of cuts, and they are identifiable as peaks, where cut classification is possible through spatial analysis of the plot. Zabih's *et al.* approach is not robust to rapid changes in global illumination, or shots which are too dark or too bright. Another serious problem is that they use a single global edge threshold to find edges and could result in scenes that have too many or no edges.

The problem of global illumination appears to be a contributory factor for degraded performance benchmarks of scene cut detector tests. Kong *et al.* (37), addresses the problem of illumination variation, where improvement in the isolation of shots is achieved through Colour Ratio histograms (24).

Camera operations produce successive differences of the same order as gradual transitions. Common operations such as zooming, panning, and tilting need to be detected to differentiate between them and distinguish them from a scene transition. Toller *et al.* (66), reports a 95% success rate for the detection of shot transitions that consists of camera breaks, fades, dissolves, and wipes, on MPEG video. Their technique is a combined approach that uses heuristics and metrics generated from motion vectors, histogram comparisons and edge information. The computation of optical flow to classify camera operations on a frame-by-frame basis is infeasible, "Unfortunately, determining an entire frame of motion vectors with high spatial resolution is a very time consuming process..." (77).

However, MPEG uses motion information to achieve high compression rates, and Toller *et al.* exploits this fact for the feasible practice of camera motion classification.

There are two problems associated with motion vectors in MPEG videos. The first, motion information is not indicative of the ground truth representation, but the best matching blocks of pixels, where MPEG encoders optimise for compression and not necessarily for accuracy. Toller *et al.*, relies on edge information to filter out motion vectors that best represent the actual motion. This relies on the premise that detail, and hence edge information, in a video frame tend to produce accurate vectors. The second problem is, not all MPEG videos contain motion vectors (76).

4.2.1.4 Summary

Shot boundary detection has been researched extensively and this review covers the principle ideas behind the techniques used. A more substantial review may be found in Toller *et al.* (66). In the next section the implementation of the Shot-Break Detection SCD is described.

4.2.2 SCD Encapsulation

The current implementation of the VPU for the Shot-Break Detection SCD uses the Net Comparison (NC) method proposed by Xiong *et al.* (74). Principally the NC method was chosen as the algorithm for the shot-break detector SCD because it was simple to implement and it is very effective to detect the most common type of scene transition, the hard-cut.

A VCR System containing large numbers of SCDs will inevitably be detrimental to performance and rapid techniques used in SCDs are favorable. The NC method has better performance when compared to pair-wise, likelihood, and histogram methods where they are applied on whole video-images. In comparison, the NC method can use parts of video-images to test for changes in video

sequences. The algorithm for Net Comparison is shown in Table 4.1.

1) Uniformly take m rows$(h_1, h_2, ..., h_m)$ and n columns $(v_1, v_2, ..., v_n)$ on the image.
2) Along h_i and v_j $(i = 1, 2, ..., m; j = 1, 2, ..., n)$, consecutively take non-overlapping square regions which are denoted as $ha_{i1}, ha_{i2}, ..., ha_{ip}; va_{j1}, va_{j2}, ..., va_{jq}$.
3) Compare the corresponding regions (ha_{ik} or va_{jl}) of the successive images by computing the difference D_1 between the mean values of the region's RGB intensities.
4) If D_1 is greater than threshold T_1, we shall say that the region has changed between the two frames, otherwise it is unchanged.
5) If the number of changed regions is greater than threshold T_2, a camera break is declared.

Table 4.1: The Net Comparison algorithm.

Step 3 of the algorithm analyses successive images by accessing video images stored in buffers B_0^{SRP} and B_1^{SRP} in the SCD Correlator. Through experimentation on a large proportion of videos it was found that appropriate values for T_1 and T_2 were at 100 and 0.5 respectively. Note, in the implementation of the SCD, $T2$ is compared to the percentage of changed regions to account for the varying dimensions of video frames.

The output of the Shot Break Detection is expressed in the SPU section of Figure 4.7. The SCD simply outputs an event-type SCD-Object to indicate the detection of a shot-break. The "*" in the figure denotes an event-type SCD-Object where linking across frames performed by the SCD Correlator is not required. The event-type status is implemented by adding a flag to the SCD-Object attribute list.

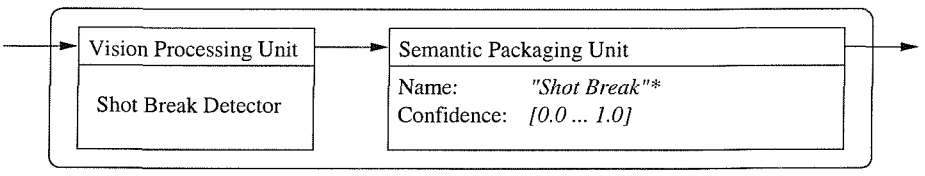


Figure 4.7: An SCD template integrated with a shot-break detector.

To demonstrate the shot-break detection SCD, Figure 4.8 presents frames of a sample news clip with their frame-to-frame comparisons. The changed regions between successive frames are shown as white squares. A camera break, as indicated in the figure, is declared when the percentage of changed regions exceeds a threshold T_2 (the activity).

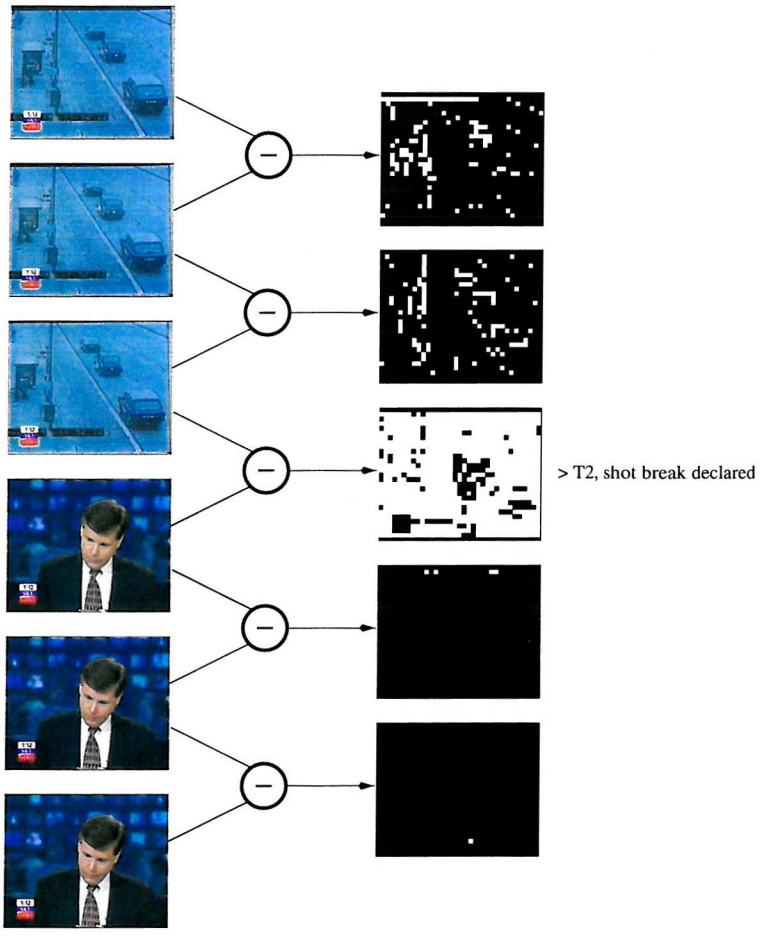


Figure 4.8: A video sequence and the activity of changed regions.

Video clips can have arbitrary numbers of shot-breaks. For the NC algorithm, hard-cuts are evident as peak signals in the total activity of frame comparisons. Figure 4.9 shows plots of activity of the total-changed-regions for three news video clips. The sharp spikes above the activity threshold of 0.5 correspond to genuine hard-cuts, and no hard-cuts were missed. Subsection 4.4.2 will present the shot-break SCD in the context of the VCR System.

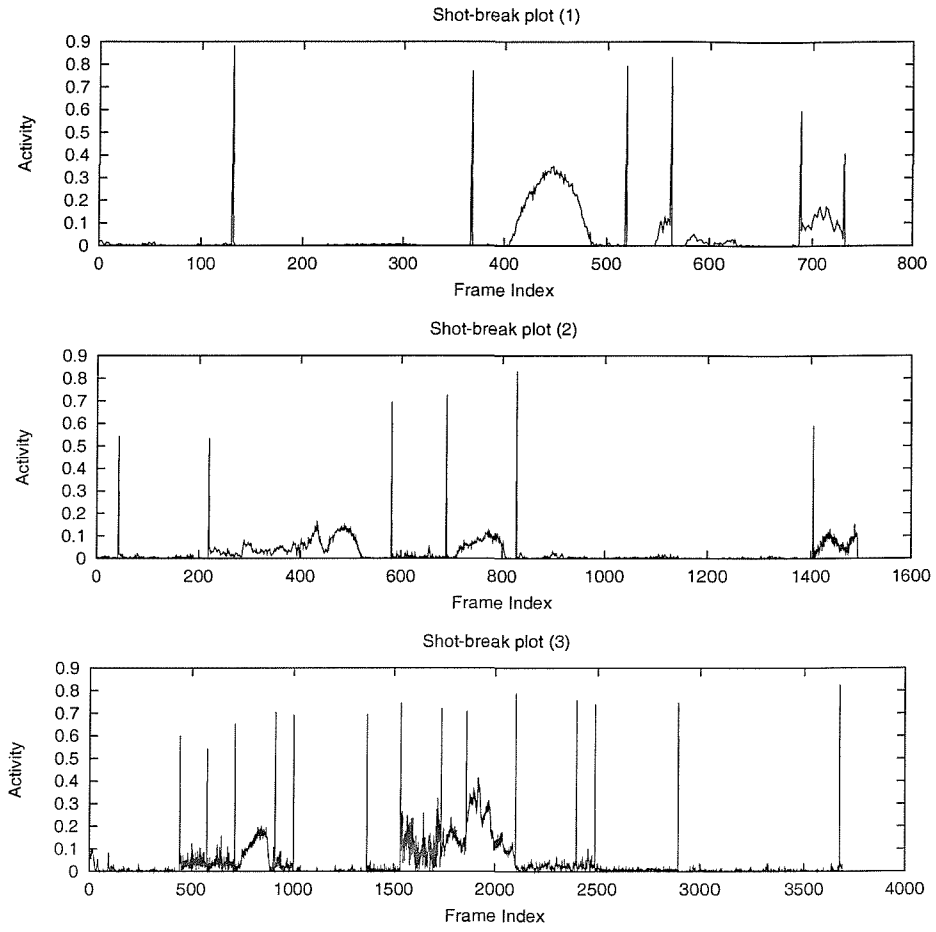


Figure 4.9: Samples of shot-break activity.

4.2.3 Summary

The stages in the development of a shot-break SCD detector is presented in this section. The primary method for the detection of shot-breaks is based on the Net Comparison method where parts of consecutive video images are used. This method allows real-time shot-break detection which is suitable for the implementation of the SCD.

4.3 Colour Recognition SCD

The visual description of objects often include attributes such as shape, texture and colour. Swain and Ballard (68) demonstrated that some objects may be

recognised entirely on the basis of their colour alone by matching colour-space histograms. Colour has a high-degree of robustness to image variations such as orientation, shift in viewing position, and change of object shape, and it is unsurprising that it is used in many visual applications ranging from object recognition to image similarity retrieval (63).

In the VCR System, one of the attributes associated with an SCD-Object is colour. When object-type SCD-Objects are created they are initially furnished with a semantic label, position, orientation, and a list of coordinates which represent the object's shape. However, attributes such as colour are not computed during the creation of SCD-Objects because the philosophy of the VCR System attempts to maintain computational efficiency by calculating attribute fields on demand. Although, the methods used to classify colour are rapid, the VCR System does offer the flexibility to have the colour of objects included in the creation of SCD-Objects.

In this section the process of classifying colour is presented in Subsection 4.3.1, and the method for recognising named colours is given in Subsection 4.3.2. Finally a summary is given in Subsection 4.3.3.

4.3.1 Labelling of Colours

The colour of an object adds richness and a significant semantic dimension to the object's description. For a computer monitor colour may be specified in terms of three primary colours, red, green, and blue, and by varying these intensities different colours are perceived by humans. There are different standards used to represent colours. Examples include Commission Internationale de l'*Éclairage* (CIE) and YIQ commonly used for colour television signal transmission. These standards are universes which contain all the colours that exist for their domain, and they are given a general term colour-spaces.

A colour-space is a coordinate system for specifying colours. The principle of attaching colours with a semantic label is simply a mapping of colour-space

coordinates to a label. The process of colour recognition begins by projecting given intensities to a 'pre-labelled' compatible colour-space. The regions in which the projections occupy determine the colour label. Batchelor (4) used a similar scheme to recognise colours using RGB values generated from a video camera.

The chosen colour-space candidate for colour classification is RGB. The RGB (Red, Green, and Blue) colour-space is the standard colour model used by computer monitors to display graphics. There are three intensity signals, the R, G, and B, where each intensity typically ranges between [0.0, 1.0]. Colours are made up from combinations of RGB, where (0, 0, 0) is black and (1, 1, 1) is white. The RGB colour-space can be represented as a cube as shown in Figure 4.10. The main diagonal of the cube (0, 0, 0) to (1, 1, 1) is the line of all possible pure greys, and the whole colour-solid represents all the colours that may be specified on a monitor. The main reason for using the RGB colour-space is because the software which decodes video in the VCR System only provides RGB colour video images.

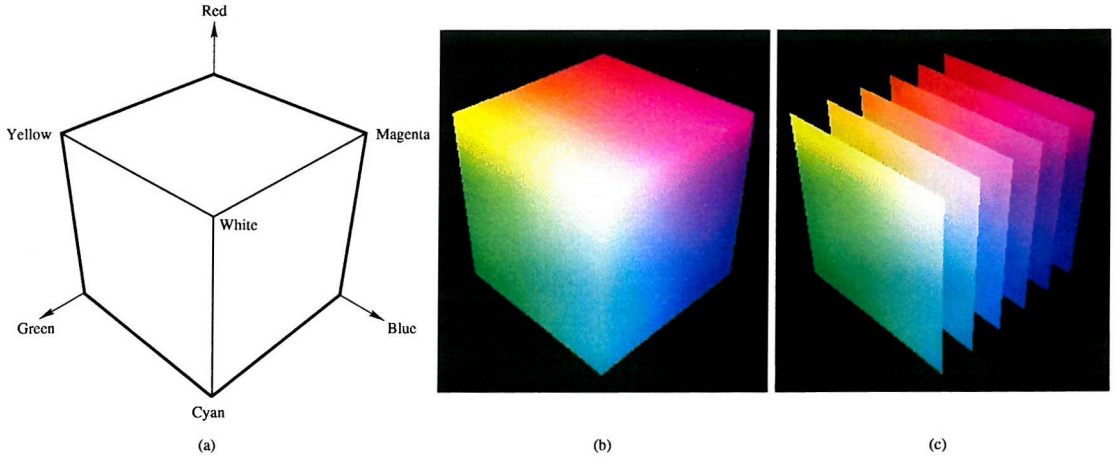


Figure 4.10: (a) The RGB colour cube colour extremes, (b) The RGB colour cube, (b) The RGB colour cube in planes.

From the given definition of RGB, how are RGB tuples for classification assigned with colour labels? Earlier, it was stated that the principle of colour recognition is based on the mapping of colour-space coordinates to pre-defined

labels. Two methods were considered: the first method uses Euclidean distances to find nearest known colour reference points and the second method identifies pre-labelled volumes of colour-space coordinates.

The method based on Euclidean distances initially prepares RGB colour-space by populating it with fixed reference points that correspond to purest hues, e.g. Red = RGB(1.0, 0.0, 0.0), Green = RGB(0.0, 1.0, 0.0), Blue = RGB(0.0, 0.0, 1.0) etc. The first step in classifying the colour of a given RGB tuple, x, calculates the Euclidean distances between x and all the reference points. The colour label of x is the colour label of the closest reference point. The shortcomings of this method are that it does not take into account the fact that the RGB colour-space is not perceptually uniform and less accurate classifications may occur as a result. Although the method may be extended to compensate for the peculiarities of the RGB colour-space, the second method using pre-labelled regions of the RGB colour-space offers a more intuitive and *controllable* solution.

Labelling volumes within colour-space to representative colour labels has the advantage that it may be implemented using a lookup-table providing very fast classification. However, the following points must be considered:

- On modern graphics hardware a total of 16,777,216 or more colours are possible.
- RGB colour-space is not perceptually uniform.

The creation of a lookup-table to account for the huge number of colours on typical computer systems is infeasible. Small uniform intervals along any straight line in RGB colour-space will not produce colours that are perceptually different as far as the human visual system is concerned (71). However, by quantising the colour-space to encapsulate larger regions of colour (thus reducing the colour resolution) a manageable lookup-table of colours is possible.

The second item on the listed points presents a formidable task in identifying boundaries of colour volumes. The non-perceptually uniform characteristic of

RGB colour-space affects the size and shape of voxels where they may intersect colour boundaries. Identifying these boundaries with precision is not helped by the fact that RGB colour-space is device dependent where colours may differ from monitor to monitor depending on the physical characteristics of the screen. In view of these considerations a more convenient model for identifying colours by transforming RGB values into an intuitive specification format is introduced in the next paragraph.

The Hue Saturation Value (HSV) model proposed by Smith (65) offers a way to specify colours in an intuitive fashion by remapping RGB values. Colours are controlled by specifying a pure hue and a tone of that hue is obtained by adding lightness (saturation) or darkness (value). The HSV colour-space is usually represented as a single hexcone as shown in Figure 4.11. Hues are selected by varying the value H (0 to 359 degrees), and tones may be obtained by manipulating S and V ([0.0..1.0]). Decreasing S (saturation) of a hue corresponds to adding white, and decreasing V (value) corresponds to adding black.

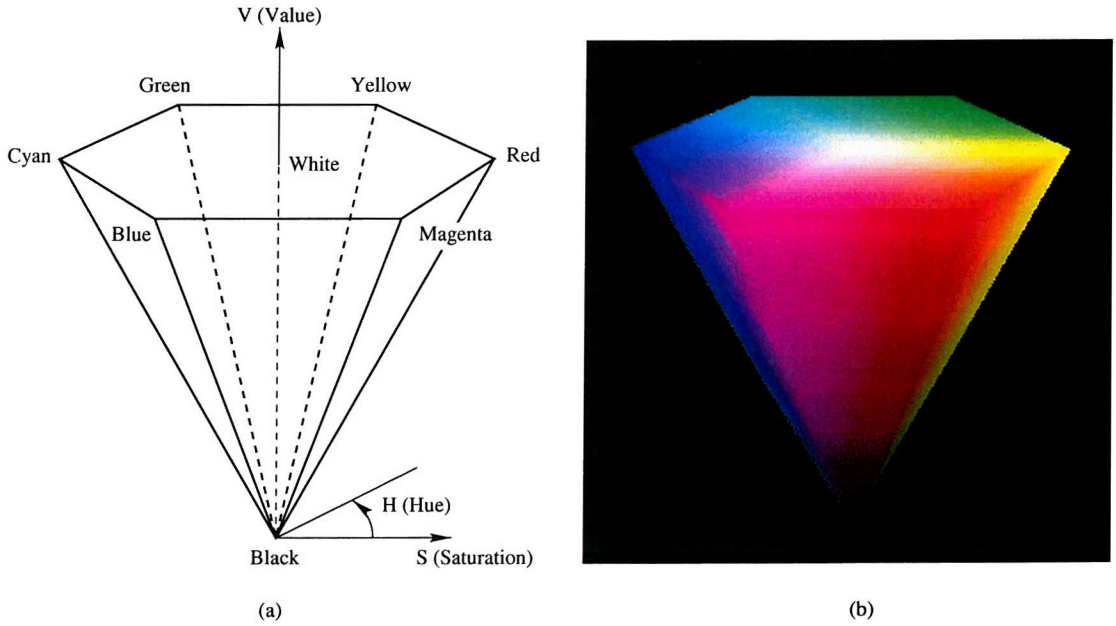


Figure 4.11: (a) Labelled HSV boundaries, (b) The HSV hexcone.

Video images are *generally* converted to RGB before any processing is per-

formed. The hues of RGB tuples may be obtained by converting them to HSV (algorithms are in Smith (65)), thus, the colour of the RGB tuples are determined. Volumes of colour may be conveniently specified by establishing HSV ranges, and this may be easily visualised by converting the hexmodel in Figure 4.11 into a cuboid volume as shown in Figure 4.12.

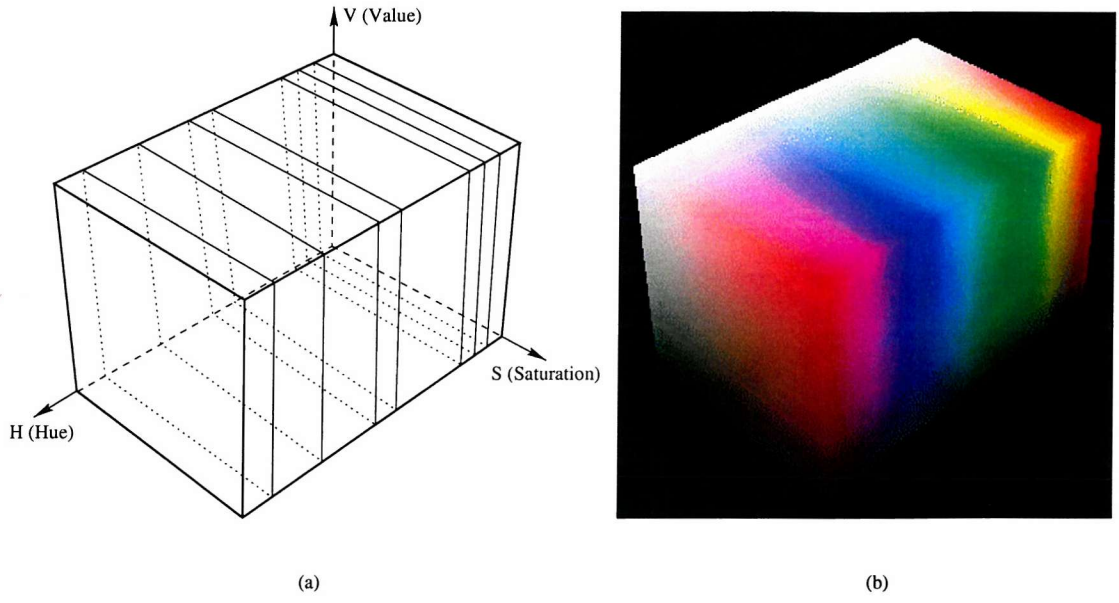


Figure 4.12: (a) HSV wire-frame with hue boundaries marked, (b) The HSV colour cuboid.

In contrast to RGB, the remapped HSV model offers a convenient way to label colours and to implement colour recognition by specifying rectangular volumes which encase blocks of colours. Figure 4.11(a) and Figure 4.12(a) illustrates these volumes. The colour boundaries were determined empirically based on the author's perception of the colours displayed on a colour calibrated monitor. Gong *et al.* (26) specified a table of similar HSV values for colour labelling. The advantage of segmenting the HSV colour-space into rectangular volumes is that it offers fast classification performance. Table 4.2 lists the limits for each of the primary hues in HSV. The table may be extended to incorporate colours such as pink by specifying narrower HSV ranges. The next section describes the encapsulation of

these ideas in an SCD.

Colour	H (degrees)	S	V
Red	[330, 360) ⋃ [360,10)	$[S_T, 1]$	$[V_{T1}, 1]$
Orange	[10, 45)	$[S_T, 1]$	$[V_{T1}, 1]$
Yellow	[45,60)	$[S_T, 1]$	$[V_{T1}, 1]$
Green	[60, 170)	$[S_T, 1]$	$[V_{T1}, 1]$
Cyan	[170, 205)	$[S_T, 1]$	$[V_{T1}, 1]$
Blue	[205, 265)	$[S_T, 1]$	$[V_{T1}, 1]$
Magenta	[265, 330)	$[S_T, 1]$	$[V_{T1}, 1]$
White	-	$[0, S_T]$	$[V_{T2}, 1]$
Grey	-	$[0, S_T]$	$[V_{T1}, V_{T2}]$
Black	-	-	$[0, V_{T1}]$

Table 4.2: The HSV ranges for colour labels. S_T and V_T are predefined ranges.

4.3.2 SCD Encapsulation

The Colour Recognition SCD receives object-type SCD-Object as input and calculates the colour of the associated object. The output from the Colour Recognition SCD is the same as the input SCD-Object but with the *Colour* field assigned with a representative label.

Figure 4.13 illustrates the Colour Recognition SCD receiving an SCD-Object and retrieving the associated video image from the SCD Correlator for colour processing.

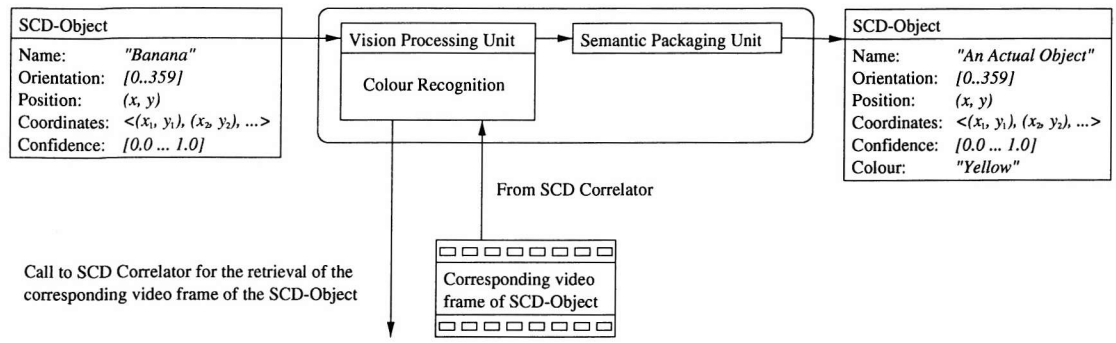


Figure 4.13: The architecture of the Colour Recognition SCD.

Figure 4.14 illustrates on the procedure for the colour recognition of objects. SCD-Objects that represent objects in video contain a set of points which resembles the object's shape, see Chapter 3.2. The coordinates are used to mask off the object's area in the corresponding video frame. The colour values within the object's profile are averaged to a single RGB colour tuple. This RGB colour tuple is mapped to the HSV colour-space and the pre-labelled volume which it occupies signifies the colour.

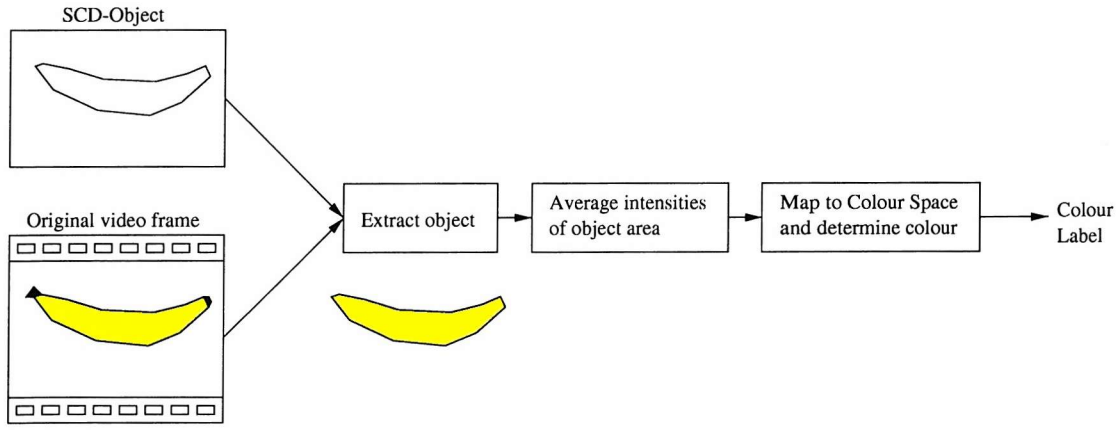


Figure 4.14: The stages in determining the colour of objects.

In reality, objects are generally composed of a multitude of colours. Video clips of objects often exhibit lighting phenomena such as the recorded illumination effects and shadows which introduce extra 'colouring'. Averaging the pixel values

of objects to obtain a colour value for recognition is only one of many techniques which may be considered, e.g. using the dominant colours of the object's region. Experiments on the Colour Recognition SCD are described in Section 4.4.3, but first a summary is presented.

4.3.3 Summary

In summary, the Colour Recognition SCD receives object-type SCD-Objects and classifies the corresponding visual area with the most representative colour label. The average colour of an object's spatial region is used to determine the colour of the whole object. The process of recognition uses a pre-labelled HSV colour-space. RGB representations of objects are remapped into HSV and the labelled volume in which the projection occurs indicates the colour.

4.4 Video Analysis Using SCDs

The previous section presented the typical development stages for some sample SCDs which detailed their functions and algorithms. In this section the Face Location, the Scenebreak and the Colour Recognition SCDs are used to demonstrate the creation of the different SCD-Object types. The ways in which SCD-Objects are created and processed are described in this section and will provide an insight into the way the SCD Manipulator processes SCD-Objects.

The SCDs detailed in this chapter reveal three different ways SCDs process for content and they are:

- The generation of object-type SCD-Objects (Face Detection SCD).
- The generation of event-type SCD-Objects (Shot-Break SCD).
- The use of SCD-Objects for the generation of further SCD-Objects, or the calculation of SCD-Object attributes (Colour Recognition SCD).

SCD-Objects represent the presence of objects and they are also used to indicate events in video. The type of the SCD-Object determines the way it is processed by the SCD-Correlator. Using the SCDs presented in this chapter, a detailed approach will be used to show the relationship between the SCD and the SCD-Correlator. Experiments and results using real video footage is used to demonstrate the above cases.

4.4.1 Face Detection in Video - Object Generation

The purpose of this section is to demonstrate the capabilities of the SCD-Correlator in video analysis by using frontal face SCD-Objects as input. News videos of anchor-person shots are used as the input. Figure 4.15 shows news video detected for faces whilst entering frame-by-frame into the SCD-Correlator. The Face Detection SCD locates and creates frontal face video-objects which are stored in the corresponding buffers within the SCD Correlator. The figure shows that at time t_0 the video frame f_0 is analysed for faces, and a result is found. Similarly, a face is found at t_1 for frame f_1. For the next two frames in the video sequence the Face Detection SCD failed to find faces f_2 and f_3. The missed detections may be explained by a close examination of the video where the anchor-person is in the process of blinking. In the next frame, f_4, a positive result is found, at which point the SCD Correlator detects that there are missing SCD-Objects. The SCD Correlator uses the face SCD-Objects found in frames f_1 and f_4 to recover the 'missing' faces for frames f_2 and f_3 which are indicated as dashed boundary boxes in the figure.

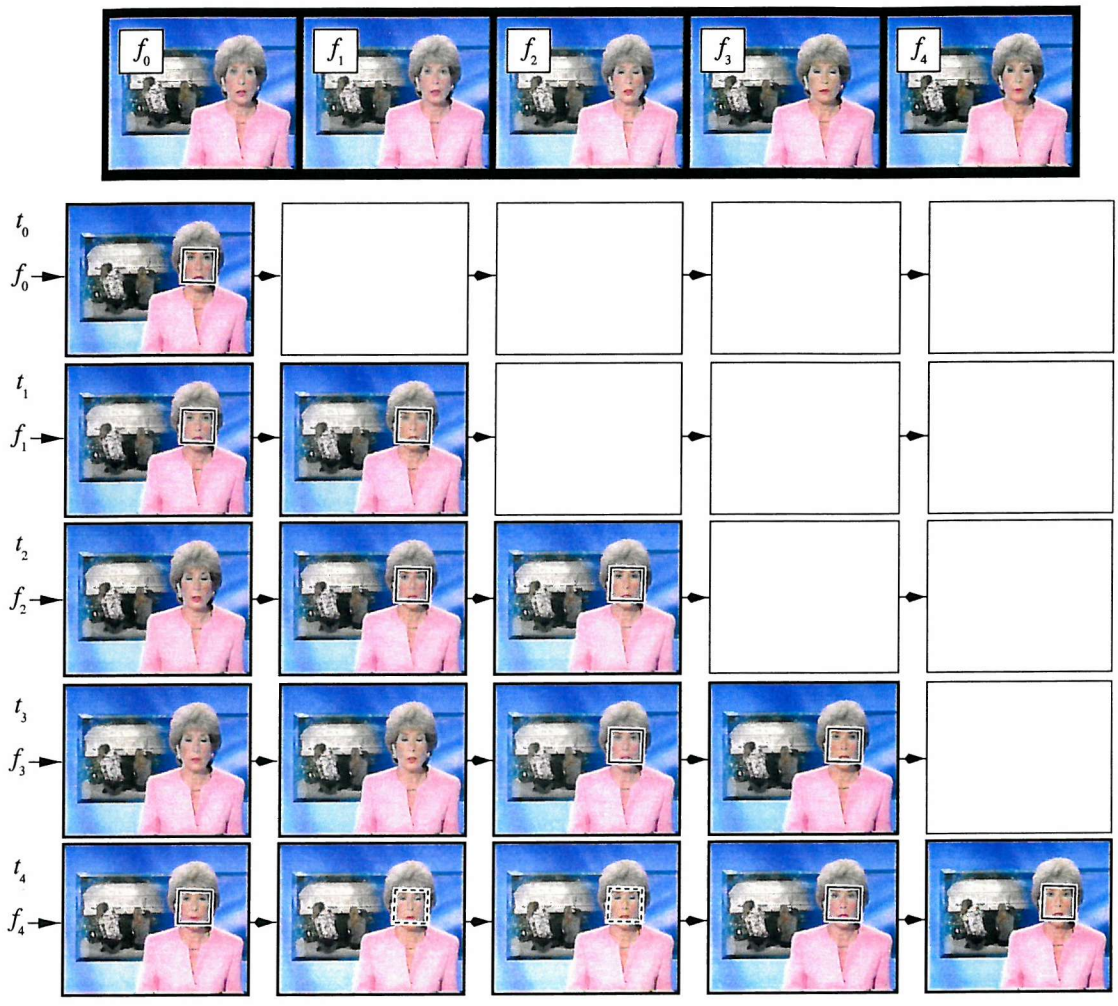


Figure 4.15: A demonstration of faces recovered in the SCD Correlator. The video frames labelled $f_{0..4}$ along the top of the figure represent the input to the SCD Correlator. Faces detected by the Face Detection SCD are represented by solid bounding boxes. Predicted face locations are shown in dashed bounding boxes.

The final results in recovering the missing objects are shown Figure 4.15 where the observed result is a continuous stream of connected objects over time superimposed on video frames. However, the figure only reveals some of the processes within the SCD Correlator for the video clip. Figure 4.16 illustrates a finer view of the processes in which detected Frontal Face SCD-Objects are correlated, predicted, and suppressed. The boxes in the figure represent Frontal

Face SCD-Objects where only the orientation, the centre coordinates, and the confidence are shown respectively. An explanation of the interactions within the SCD Correlator begins in the next paragraph.

Referring to Figure 4.16, the processing for the video clips begin at time t_0 with frame f_0 where a Frontal Face SCD-Object is placed in the first buffer of the SCD Correlator (Buffer(0)). At time t_1 three different operations are performed. Firstly, there is a shift in the SCD Correlator's contents along it's buffer in which the first buffer is emptied. Secondly, two Frontal Face SCD-Objects are entered in to the first buffer, and this indicates that two faces were located for frame f_1. The third step, is the temporal correlation where the contents of Buffer(0) are compared with Buffer(1), and the result is shaded in the figure.

At time t_2, a single face is detected and is automatically linked to a face at t_2 in Buffer(1). No objects are placed in Buffer(0) at time t_3. At time t_4 a face detected in frame f_4, and upon which a Frontal Face SCD-Object is placed in Buffer(0). The SCD Correlator detected a positive correlation for the newly entered SCD-Object to the Frontal Face SCD-Object in Buffer(3) at t_4. The process of creating a temporal link between the two Frontal Face SCD-Objects involves interpolating the two SCD-Objects. The interpolation fills the SCD Correlator with the missing SCD-Objects, and it is used to predict and recover objects 'undetected' by SCDs. The figure represents predicted SCD-Objects as dashed boxes.

To complete the picture of the correlation process, the creation of video-objects is described, see Section 3.3.1 for reference. As shown in Figure 4.16, at time t_4 there are two connected Frontal Face SCD-Object chains (one of the chains is shaded). At time t_5 the two SCD-Object chains shift along the buffer. Although, it is not shown in the figure, only Frontal Face SCD-Objects related to the shaded SCD-Object chain appear in Buffer(0) in subsequent time steps, e.g. for time steps $t_{5..6}$. Given that the SCD Correlator is configured to have five buffers, at time t_5, the 'head' of the shaded Frontal Face SCD-Object chain is

'shifted out' of the SCD Correlator. At this point the SCD Correlator detects the 'overflow' and tests if the SCD-Object chain is connected in at least L_{min} frames. In this case, L_{min} was set to a value of four, thus SCD-Objects belonging to the shaded Frontal Face SCD-Object chain is submitted to the next stage, e.g. the SCD Manipulator. As part of the registration process to submit SCD-Objects to the next stage, a unique ID is assigned to the same chain of Frontal Face SCD-Objects as they leave the SCD Correlator. This allows the SCD Manipulator to identify related object-type SCD-Objects invariant of time, thus video-objects are created.

For the second Frontal Face SCD-Object chain, it begins to exit the buffers at time t_6 at which point the whole chain is composed of two Frontal Face SCD-Objects. Since the length is less than the set value of L_{min}, it is automatically rejected by the SCD Correlator.

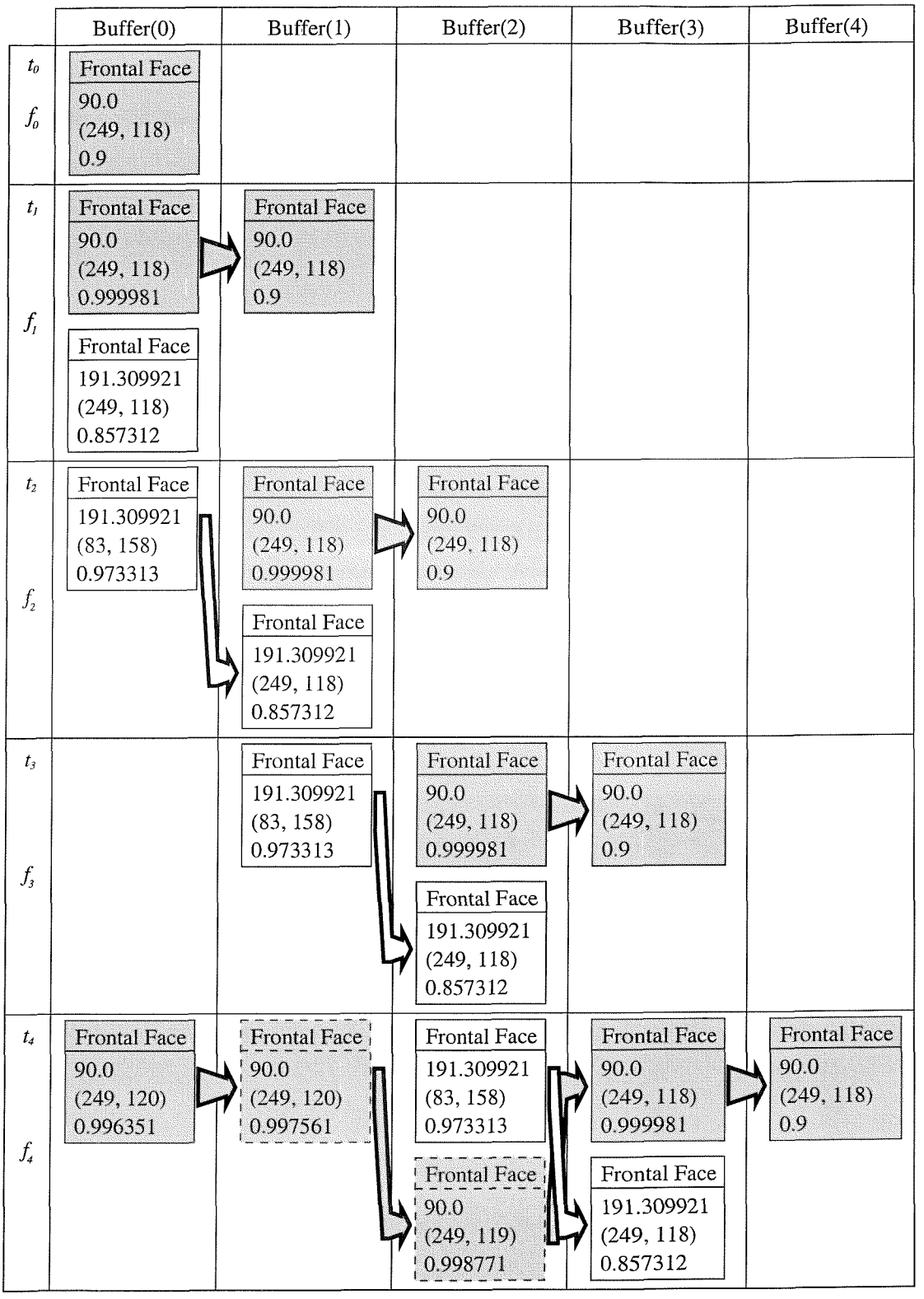


Figure 4.16: A detailed view of the correlation process within the SCD Correlator.

The SCD Correlator has the secondary role of labelling SCD-Objects with unique identification numbers as they enter and leave the last buffer. SCD-Objects may be referenced outside of the SCD Correlator if they are successfully labelled, thus video-objects may be referenced by a single value. In the case of Figures 4.15 and 4.16 the suppression of SCD-Objects is evident for the 'unshaded' Frontal Face SCD-Objects where they are not permitted to continue outside of the SCD Correlator. The SCD Correlator deleted them because of their brief existence. This is a demonstration in which false positives are eliminated, where it is assumed that any sporadic appearances of SCD-Objects will occur without having the characteristics of genuine objects. Figure 4.15 shows the final result of the SCD correlator in which the SCD-Objects that have 'strong evidence' over a temporal domain are visible to the outside of the SCD Correlator.

Although, the SCD Correlator compensated for the missing faces in frames f_2, f_3, an explanation into why the faces were not detected is provided. To answer the question why faces were not detected for frames f_2 and f_3 a detailed examination of the message logs generated by the VCR system was necessary. The logs revealed that in-fact faces were detected. The missing faces have the following SCD-Object attributes(*Orientation, Centre Coordinates, Confidence*):

f_2 $FrontalFace(85.601295, (248, 119), 0.406541)$,

f_3 $FrontalFace(90.0, (249, 120), 0.748379)$.

For the missing face in frame f_2 the low probability is below the threshold in which the Face Detector decides whether a potential face candidate is a face or not a face. The centre coordinates have the expected values, but it does not explain why the probability is low. Further investigations revealed that the low probability is, in this case, related to the orientation of face where it is approximately five degrees off the expected angle. Face candidates are found by detecting eye-pairs, but in this case an 'eye to eye-brow pair' explains the value of the orientation. Since the neural network within the face detector SCD was trained to classify upright frontal faces, a low probability will result for faces with

features that do not match the training set.

The 'missing' face in frame f_3 contains the expected values for orientation and centre coordinates, but the probability is slightly below the set threshold in which the face detector will allow face candidates to be classified as faces. The lower probability corresponded at the peak of the anchorperson's blink in which the eyes were fully closed. It is postulated that the lower probability is related to the training set of the neural network where it was trained on faces with opened eyes. Although experiments have not been performed to confirm the hypothesis, the original references (56) suggested that the neural network focussed heavily on the eyes which adds credibility to the hypothesis. The solution to the problem of blinking eyes may be found by lowering the threshold, but experiments on a series of faces with fully shut eyes must be conducted in-order to determine the lower threshold for the Frontal Face SCD neural network. It can be seen that the recovered face for frame f_3 is highly comparable to the face which would have been allowed if a lower threshold was used within the face detector. Recovered face attributes for frame f_3 are $FrontalFace(90.0, (249, 120), 0.997561)$ which compares with the actual face attributes of $FrontalFace(90.0, (249, 120), 0.748379)$.

The demonstration of the short video clip in this section raises the question of how the SCD Correlator knows when to start and stop correlating for a chain of semantically equivalent SCD-Objects across shots? A revisit to the correlation algorithm in Section 3.3 Correlate objects across buffers shows that it makes use of shot-breaks to control the correlation process. The next section will highlight this in detail, and results of shots breaks detected in video are presented.

4.4.2 Shot Break Detection - Event Generation

SCD-Objects come in two types, one which represents objects and one which indicates events. The type of SCD-Object determines the way it is processed within the SCD Correlator. SCD-Objects which belong to the event-type category shift through the SCD Correlator untouched. The shot-break detection SCD is

an example of an SCD that generates event-type SCD-Objects.

The non-contiguous nature of event-type SCD-Objects renders them unsuitable for temporal linking, thus, the SCD Correlator's *error correction* capabilities cannot be used. Event-type SCD-Objects are ignored by the SCD Correlator, and propagate through to the end of the pipe unchanged. This imposes the necessity of robust technology for the generation of event-type SCD-Objects. Fortunately shot-break detection is well researched, and high detection rates may be achieved.

Shot-breaks play an important role in regulating the assignment of unique IDs for object-type SCD-Objects. Consider, a video sequence with two similar shots, e.g. an anchorperson shot switching to a news bulletin consisting of a scene focussed on a reporter, the detection of face objects will result in the detection of a single face video-object instead of two unique faces because of their similar characteristics. By detecting transitions between shots, the correlation may be controlled by restricting the process between shot boundaries. This has the effect of creating separate *object threads*, and in which each thread is then uniquely labelled.

The current implementation of the Shot-Break Detector only detects hard-cuts, and plans to extend the detection to other varieties of scene transitions is on the agenda of future work. Table 4.3 shows that for a set of news video clips 91% of all hard-cuts were detected. The false positives detected in the news videos were due to recorded camera flashes which are not uncommon with news bulletins. The camera flashes dramatically altered the contrast for the whole image, thus, giving the false appearance of a scene change. In this case each camera flash generated two false positives, one before the flash and one after the flash. The problem of shot-break detection when the incident illumination varies may be addressed by using colour ratio histograms, Kong et al. (37). However, this too is left as a future extension.

Video File	No. of Hard-cuts	Detected	Missed	False Positives
anchorp1.mpg	17	14	3	0
anchorp2.mpg	3	2	1	0
anchorp3.mpg	12	11	1	2
anchorp4.mpg	17	14	3	0
anchorp5.mpg	5	5	0	0
anchorp6.mpg	24	23	1	2
anchorp7.mpg	13	12	1	0
anchorp8.mpg	5	4	1	0
anchorp9.mpg	14	14	0	0
anchorp10.mpg	6	6	0	0

Table 4.3: Results of 10 videos for which hard-cuts were detected.

Currently, two different methods of generating SCD-Objects from actual video data have been presented. The third approach to content generation is the generation of SCD-Objects from other SCD-Objects and this is demonstrated in the next section using the Colour Recognition SCD.

4.4.3 Colour Recognition - Querying of SCD-Objects

In the VCR System, SCD-Objects are analysed by SCDs which may be invoked in two ways: from the SCD Manipulator, and from other SCDs. In this demonstration the Colour Recognition SCD and Face Location SCD are used in conjunction with the SCD Manipulator. The specific function of the SCD Manipulator Core processes Frontal-Face SCD-Objects for colour content using the Colour Recognition SCD, and displays the result. The implementation of the SCD Manipulator Core relies on hard-coded if-then-else constructions.

The Colour Recognition SCD classifies the colour contained within regions specified by the pre-defined shape area of object-type SCD-Objects, and in this case they are face regions. Figure 4.17 illustrates the overall process.

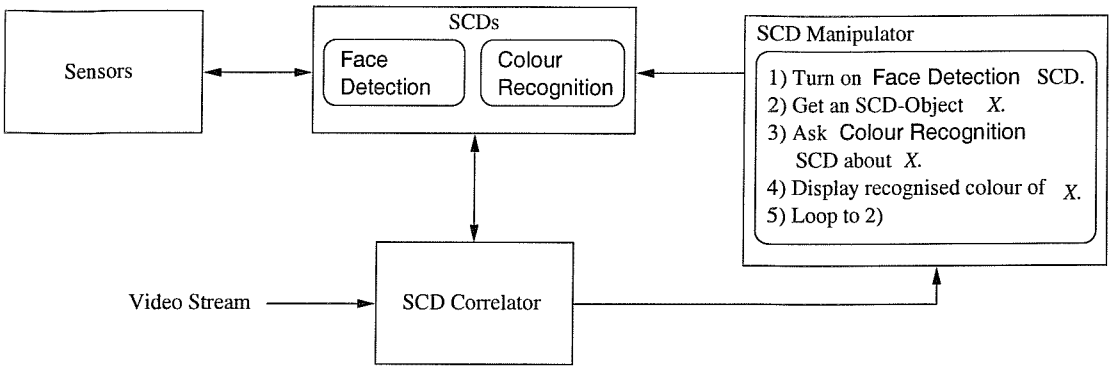


Figure 4.17: An overview of querying face SCD-Objects using the Colour Recognition SCD.

Table 4.4 shows the results of querying faces for colour from the video clip in Section 4.4.1. The 'grey' classification for the face results is perceptually the most accurate label the Colour Recognition SCD is currently able to dispense, but the important demonstration here is that this is delivered in response to a user query. Colours with a nearer representation of faces may be obtained by isolating areas of facial colour (29) (25) (72) within the HSV colour-space. The HSV ranges for colour labels, Table 4.2, is extensible to accommodate finer labellings of colour by specifying new limits.

	f_0	f_1	f_2	f_3	f_4
Colour	Grey	Grey	Grey	Grey	Grey
Average H	4.29	4.09	6.67	5.45	5.45
Average S	62.63	64.86	66.33	64.86	64.86
Average V	171	173	173	173	173

Table 4.4: Results of colour recognition on faces in frames $f_{0..4}$.

4.5 Conclusions

This chapter has presented part of a general architecture for digital video analysis. To illustrate the development of specific content detectors (SCDs) within the

architecture, three SCDs were developed, for face detection, shot-break detection and colour recognition respectively. Brief surveys of these areas were presented before the chosen approaches were introduced.

The SCDs process for content in the following manner:

- The generation of object-type SCD-Objects which represent visual objects (Face Detection SCD).
- The generation of event-type SCD-Objects which represent events (Shot-Break SCD).
- The use of SCD-Objects for the generation of newer/or more elaborate SCD-Objects (Colour Recognition SCD).

The Face Detection SCD clearly demonstrates the generation of object-type SCD-Objects. Frontal Face (object-type) SCD-Objects are linked together within the SCD Correlator to form video-objects. Video-objects are given unique identification numbers with which they may be independently referenced across the temporal domain.

The Shot-Break SCD generates SCD-Objects to indicate the existence of events. 'Shot-Break' SCD-Objects are a special case for the SCD Correlator, and they are used to determine the beginnings and the ends of SCD-Object linking, i.e. the change of context.

The Colour Recognition SCD demonstrates the querying aspects of the VCR System. The interactions between the SCDs during the querying process begins with the generation of Frontal Face SCD-Objects which are eventually sent to the SCD Manipulator. The Manipulator Core invokes the Colour Recognition SCD with the Frontal Face SCD-Objects as the query input, and the results are displayed.

The SCDs presented in this chapter are general video analysis tools and they can be used in any SCD framework. To give an example, the Colour Recognition SCD may be viewed as a 'generic' SCD where any object-type SCD-Objects

may be queried for a colour label, e.g. What colour is this object? Similarly, SCDs may be constructed to process specific SCD-Objects because of certain preconditions, e.g. the prerequisite of located frontal faces. Examples of SCDs which may specifically require Frontal Face SCD-Objects as input include face recognition (70) (45), and face expression classification (6). The next chapter focuses on SCDs that have a specific dependency on a video genre and the content representation of SCD-Objects.

Finally, the SCD Correlator has 1) demonstrated the recovery of 'missing' content by interpolating SCD-Objects, 2) the elimination of SCD-Objects which pertain to false positives, and 3) the creation of video-objects. Processing video in the form of SCD-Objects allows the SCD Correlator to function in a general manner irrespective of the actual video content. The analysis of videos in terms of objects has a significant advantage in that the objects may be manipulated at a semantic-level. The framework of SCDs and the SCD Correlator form the initial key-part of object oriented video analysis.

The next chapter explores and develops a VCR System for snooker videos, which combines context, object recognition, event detection, and higher semantic analysis based on object interaction in a single framework.

Chapter 5

A VCR Framework Application and Evaluation

The previous chapter is concerned with the development of general video analysis SCDs. In this chapter contributions in object recognition technology and vision processing are presented, whereby constraints of the context across genre and video content are imposed. By narrowing the scope of the type of video, it is possible to make assumptions for *simplifying* the vision analysis. The context dependent assumptions may allow for the extraction of semantics such as objects.

The specific aims of this chapter are to show how context assumptions can be used to simplify video analysis, and how high-level events may be inferred from object-level entities. To show these aims effectively, snooker videos are chosen as the video genre. In contrast to the generic SCDs in the previous chapter, SCDs with specialised processing tasks will be presented.

This chapter begins with the identification of assumptions within the domain of the chosen video genre in Section 5.1. This is then followed by Data Collection in Section 5.2, followed by two sections on object detection SCDs in Sections 5.3

and 5.4. Finally, the Experimental Results, Evaluation and Conclusions are presented in Sections 5.5 and 5.6, respectively.

5.1 Context Assumptions

This section presents in detail the types of semantics extracted from top-view scenes of snooker video.

Prior to the actual video analysis, the philosophy of the VCR System is to begin by specifying the objects of interest. This is then followed by an evaluation of the object types to establish the technical feasibility of decomposing video into objects for the chosen video domain.

The results of the evaluation aim to provide foundations for laying down assumptions to aid the problem of object recognition and detection. The rest of this section presents itself as a guideline for designing SCDs.

The video domain presented in this chapter focuses on televised snooker programmes. When snooker video is broken down into shots they can be categorised into different sub-genres which include the following:

- Anchor-person shots.
- Top-view shots.
- Side-view shots.
- Player tracking shots.
- Close-up of balls shots.

Each sub-genre requires its own specially designed SCD(s) which use *built-in-knowledge* for object decomposition. In this chapter SCDs are specifically designed to process shots which are classed as *top-view*. The top-view shot is the best category from the list which can fully demonstrate the ideas of the VCR System.

The first step is to decide which objects are to be detected. A typical top-view scene in a snooker video is shown in Figure 5.1.

Figure 5.1: A typical top-view scene of snooker videos.

The scene consists of the following *objects*: player(s), referee, table, balls, floor etc. The objects that reflect the dominant semantics of the scene are the snooker balls. Observation of top-view shots shows that the snooker balls are generally filmed on the actual table. This observation presents an opportunity for designing an efficient and reliable SCD for detecting snooker balls.

Locating the table is conceivably more reliable than locating the snooker balls. Identifying the table before the balls demonstrates *simplified* vision analysis. The reasoning is as follows, given that the snooker balls are almost always on the table, then the machine vision analysis for the balls may be confined to the visual area of the table. Constraining the problem of snooker ball detection to the confines of the table has the following advantages:

- Improved reliability in detecting snooker balls. This is because processing is confined to an area known to *contain* the balls, thus the chances of false positives are reduced.

- Higher confidence in the results. Prior knowledge assumes only snooker balls are on a table, so detected *ball objects* do visually represent snooker balls.

- Efficient processing. Reduced search space for ball location improves the processing time.

Confining the snooker ball detection process to a visual area which represents the top-view of the table contrasts with *generalised* object recognition algorithms. This demonstrates that there are not only constraints in the visual analysis to a particular shot-type, but also to selected areas dynamically located within a shot.

The implemented SCDs for detecting top-views of snooker tables and snooker balls in top-view shots are SnookerTable and SnookerBall respectively. The SnookerTable SCD offers the chance to maximise the amount of semantic information that may be extracted from top-views of snooker. The semantics in question are the table's pockets. The locations of the pockets are predictable from results of the SnookerTable SCD. A PocketLocator SCD is designed for this purpose.

Implementing the processing hierarchically is a simple case for SCDs. Figure 5.2 illustrates the SCD hierarchy diagram of the top view snooker genre. The SCDs, *Snooker* and *Top-view* are included to generate broader semantics. Note, that they do not directly process visual data, but *built-in* prior knowledge has allowed 'coarser' semantics to be inferred.

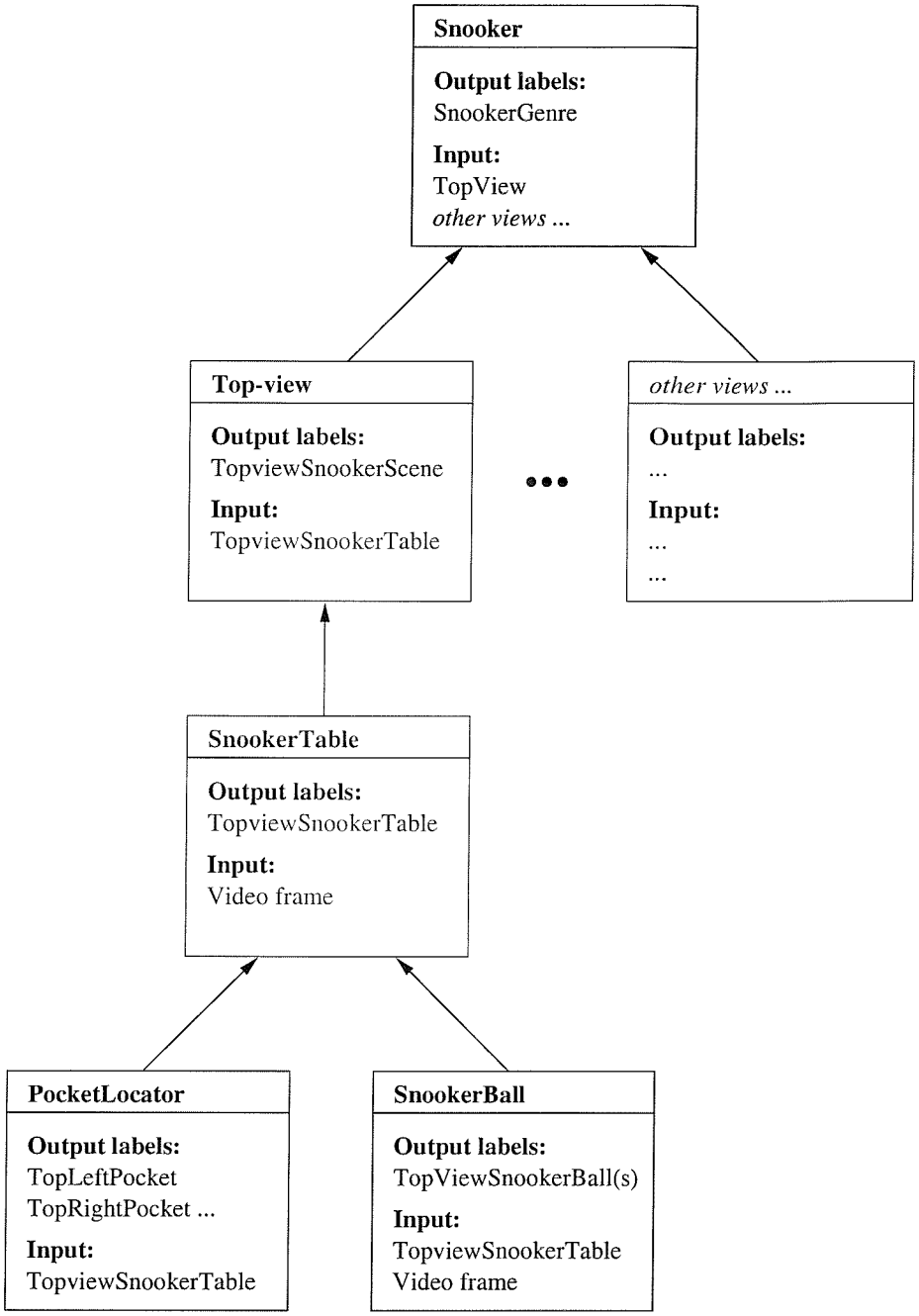


Figure 5.2: The hierarchical structure of SCDs for top-view snooker shots.

The sequence of calls in the SCD tree begins with presenting a video frame to the Snooker SCD, which calls the Top-view SCD and in turn calls the SnookerTable SCD. If a snooker table is detected then a SnookerTable SCD-Object is placed in the SCD Correlator. The successful detection of a snooker table subsequently calls the PocketLocator and SnookerBall SCDs. If a snooker table is not

detected then the 'Top-view' SCD will fail to provide input to the Snooker SCD, thus no snooker scene is present.

The current implementation only considers top-views. However, it is possible to extend the Snooker SCD to handle other sub-genres of snooker. Implementing this, as stated at the beginning of this section, would require SCDs specifically designed for a particular sub-genre which are inserted as child nodes of the Snooker SCD. The child nodes are represented as *other views* in Figure 5.2.

The SnookerBall SCD is a prime example of employing context assumptions which simplifies the processing for robust object recognition. The implementation details of the SCDs starts from Section 5.3, but first the data collection procedures are described.

5.2 Data Collection

The video clips used in the experimentation with the framework are digitised from terrestrial television programmes of snooker matches. The method used to capture video is described.

The first step of creating the video clips is to record the snooker programmes on to VHS tapes. During the recording stage it was found that the transmission signal contained artifacts such as noise, interference and colour bleeding. To counter the effects of a noisy signal, a signal booster used to amplify the incoming transmission signal reduced the effects of noise and improved the quality of the tape recording.

The second and final step in the data collection is the conversion of analogue video tape recordings into a digital equivalent. Using the appropriate software and hardware, digital recordings of the programmes are made by replaying the tapes into a computer which encoded the videos to MPEG I format. For the purposes of experimentation, top-view clips were recorded such that the beginning of each clip showed the cue ball (the white ball) being struck and ending with the cue ball deaccelerating to a stationary position.

Editing and creating visually acceptable video clips is important because the quality affects the performance of vision analysis. An important visual cue is colour. The original video signal contained a lot of colour bleeding, and regardless of the data collection setup (signal booster configuration, and the range of high quality tapes), the colour definition could not be entirely preserved.

Preserving the quality of the original signal in the form of MPEG was the next important step. Tweaking the MPEG encoder to have the most optimal parameters for quality did not improve the situation. A lot of information is lost from the original signal when MPEG videos are created from them. An accumulation of errors is visibly evident for small details such as the balls when comparing the tape and digital recordings. Firstly, the transmission is degraded, secondly, a VHS recorder only records half the resolution of the actual transmission, which results in the loss of definition. Furthermore, encoding a playback signal to the MPEG format introduces further degradation because of 'lossy' algorithms used to achieve compression, and this comes at a price where the introduction of artefacts in the video appears.

Regardless of the low quality and resolution of the MPEG clips, vision analysis is still possible. The possibilities for obtaining better recordings can be made from BetaSp format tapes used in the recording and archiving of broadcast quality video. The next section begins with the techniques used to locate snooker tables.

5.3 Table Location

This section describes the simple model-based technique used to track snooker tables that are typical of top-views. Figure 5.3 shows the top-view scene of Figure 5.1 but with the area of interest outlined.

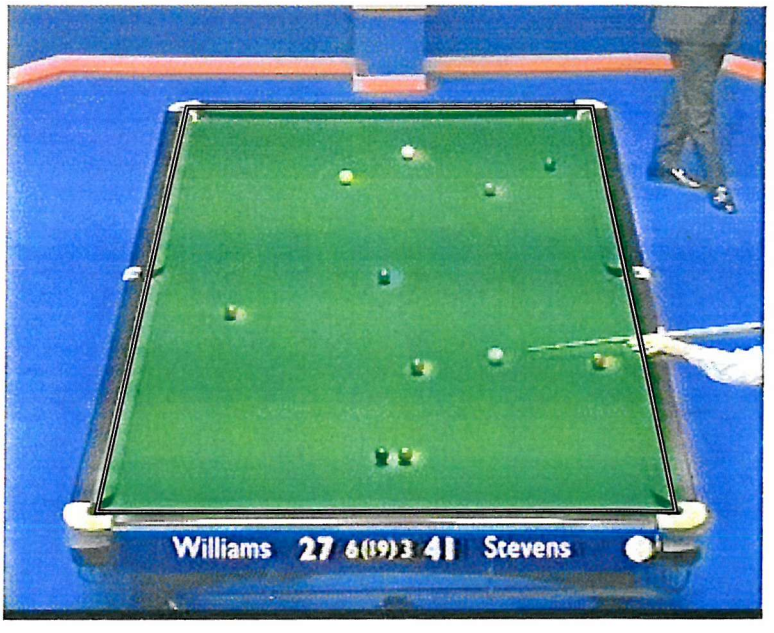


Figure 5.3: The area of interest.

The technique for finding table areas is based on the observation that videos of top-view snooker scenes are filmed with identifiable and consistent characteristics. The main feature which is typical of top-view snooker scenes is the projected profile of the table. The shape of the table resembles a trapezoid, and the top and bottom edges of the table are horizontal to the viewing angle. The algorithm for finding tables is summarised in Figures 5.4, 5.5, and 5.6, and a detailed explanation is given here.

Find top-views of snooker tables
Receive a video image, f, from video stream
Find Table Borders
Verify Table

Figure 5.4: The initial procedure for snooker table location.

Find Table Borders
$T = \frac{f_{maxY}}{4}$ for y = 0 to $\frac{f_{maxY}}{2}$ { if(horizontal green pixel count(y)> T) TopBorderYPos = y; break } for y = f_{maxY} to $\frac{f_{maxY}}{2}$ { if(horizontal green pixel count(y)> T) BottomBorderYPos = y; break } for x = 0 to $\frac{f_{maxX}}{2}$ { if(threepixelstrip(x, TopBorderYPos) == green) $TopLeftCorner_x$ = x $TopLeftCorner_y$ = TopBorderYPos; break } for x = 0 to $\frac{f_{maxX}}{2}$ { if(threepixelstrip(x, BottomBorderYPos) == green) $BottomLeftCorner_x$ = x $BottomLeftCorner_y$ = BottomBorderYPos; break } for x = f_{maxX} to $\frac{f_{maxX}}{2}$ { if(threepixelstrip(x, TopBorderYPos) == green) $TopRightCorner_x$ = x $TopRightCorner_y$ = TopBorderYPos; break } for x = f_{maxX} to $\frac{f_{maxX}}{2}$ { if(threepixelpixel(x, BottomBorderYPos) == green) $BottomRightCorner_x$ = x $BottomRighttCorner_y$ = BottomBorderYPos; break }

Figure 5.5: Algorithm for locating the boundaries of snooker tables.

Verify Table
Adjust the calculated corners to obtain the largest (in pixel area) symmetrical trapezium, t. if(average colour in area defined by t == green && average colour outside of area defined by t != green) t is declared a table

Figure 5.6: Verify candidate areas for snooker tables.

The process of finding a table from a single frame of a top-view snooker scene is a two step procedure, where the first step is a shape location process in which the borders of the table are found, and the second step verifies the detected shape for its resemblance to tables that are typical of top-view snooker video.

The shape of the table is modelled as a trapezoid, and the table location procedure begins by finding the top and bottom borders of the table. Empirical observations indicate that the top and bottom edges appear horizontal, and so, the process of finding the borders is reduced to locating horizontal strips of table coloured pixels. This first step is completed when the horizontal borders are analysed for points which belong to the corners of the table, thus creating the trapezoidal model. This is explained below.

The top and bottom borders are found by counting each scan-line of the given video image for green pixels. The top border is determined by scanning horizontally for green pixels starting from the top of the image to the centre, and similarly, the bottom border is found by scanning from the bottom of the image to the centre. If for a scan-line the number of green pixels exceeds a predetermined threshold then the position at which this happens represents the vertical position of the horizontal border.

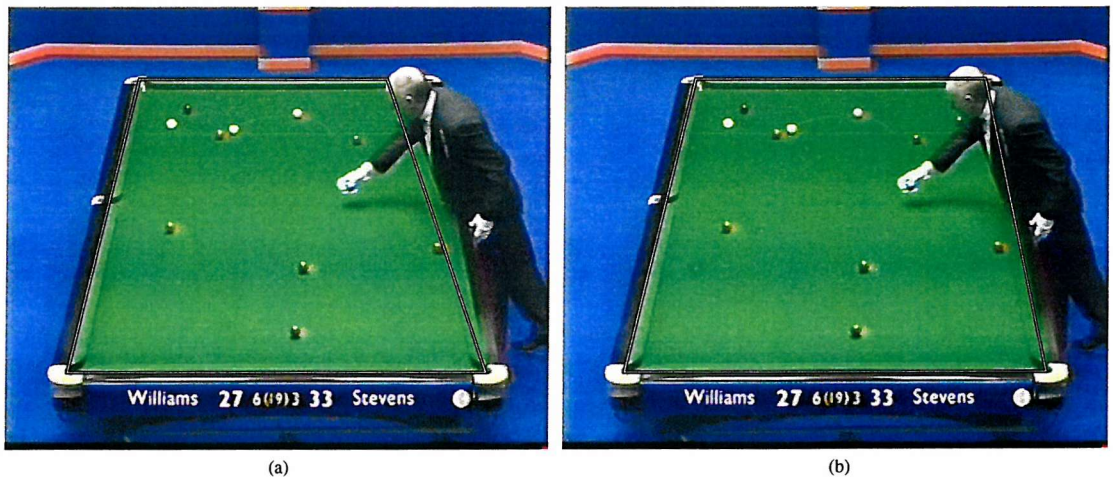
The values of the thresholds are used to determine the presence of a significant green edge, and they are defined to be proportional to the width of the video frame. Experiments using a large set of top-view snooker frames showed that table borders are detectable for all cases when the thresholds are set to a value of 25% of the pixel width of the video frame.

The corners of the table are found by scanning across the detected top and bottom borders. The order in which the scanning is performed across the horizontal borders begins at the edges of the video frame and progresses towards the centre. Scanning is terminated when green pixels are encountered, thus table corner positions are located. To avoid false positives, e.g. spurious green pixels due to noise which may cause the scanning procedure to end prematurely, a three pixel thick scanning area is used along the borders. The leading edge of the scanning line activates if and if only the complete thickness of the line, the leading three pixels, are over green pixels, e.g. `if( threepixelstrip(x, TopBorderYPos) == green )`. At this stage the shape of the table is determined, but further verification proce-

dures are performed before committing a SnookerTable SCD-Object to the SCD Correlator.

The second stage verifies the detected shape for 'trapezoidness' which examines the angles of the trapezoid. Given that the shape matches a trapezoid profile, the coordinates of the shape are adjusted such that its pixel area is maximised. This resolves problems shown in Figure 5.7a where the referee/ player walks into a corner, thus, *damaging* the shape. Figure 5.7b shows the result after shape adjustment. The process of manipulating the coordinates assumes the following:

- The table is positioned central to the viewing angle.
- Not more than one corner is obscured for the top or bottom border.



(a) (b)

Figure 5.7: a) The initial result of finding the table based on green borders. b) The result corrected by maximising the area based on the detected area.

This *dynamic* approach for table location is not the only technique that can be considered. Preliminary experiments with templates were performed. The template approach defines the shape and location typical of top-view tables and it is used to separate video frames into two regions, the table area, and the area outside of the table. A top-view table scene is declared when the colour of the *table region* is green and the area outside is not green, e.g. if(average colour in

area defined by t == green && average colour outside of area defined by t != green) t is declared a table. This approach proved to be successful, but it cannot accurately resolve shapes into actual table regions, e.g. the borders, for general top-view snooker video.

The SnookerTable SCD-Object presents an opportunity for the further extraction of semantics. The semantics in question are the *pockets*. The procedure for deriving the pockets is described in the following subsection.

5.3.1 Pocket Location

The PocketLocator SCD demonstrates the hierarchical processing structure of SCDs, where, in this case, analysis is performed if and only if a table is detected. The derivation of pockets is actually inferred from the information of the detected table, e.g. no image processing is performed to find the pockets.

Pocket locations are derived from the detected table's shape. Prior knowledge about the locations of the pockets of snooker tables is applied to the extracted table representation, where 'pocket' SCD-Objects are created. Generally, top-view scenes of snooker video show that the pockets are located at all four corners and near the centre of the left and right borders of a table. Using this information, the PocketLocator SCD submits six SCD-Objects based on the shape coordinates of a SnookerTable SCD-Object.

The locations of the four corner pockets may be precisely located by using the vertices of the detected table's shape. However, the middle pockets require extra computation. The centre pockets are estimated using the following formula:

- Left Middle Pocket = k * average of top-left and bottom-left table coordinates + $Offset_1$
- Right Middle Pocket = k * average of topright and bottom right table coordinates + $Offset_2$

The constant, k, and the offsets are related to the perspective view of the

table, and they estimate the approximate positions of the centre pockets. The technique for estimating the middle pockets can be made more generic by using the fact that real snooker tables are rectangular, and the centre pockets are positioned in the middle of the sides. The trapezoid extraction of the table's outline represents the 3D perspective projection of the rectangular shape of the table. By reprojecting the rectangular profile to a matching trapezoidal profile, the pocket locations may be precisely calculated. This is left as a future extension.

Figure 5.8 shows the results of locating pockets. Pockets are represented as square areas centred on positions defined by the table's vertices and the formula above. The following SCD-Objects are submitted to the SCD Correlator: TopLeftPocket, TopRightPocket, MiddleRightPocket, MiddleLeftPocket, BottomLeftPocket, and BottomRightPocket.

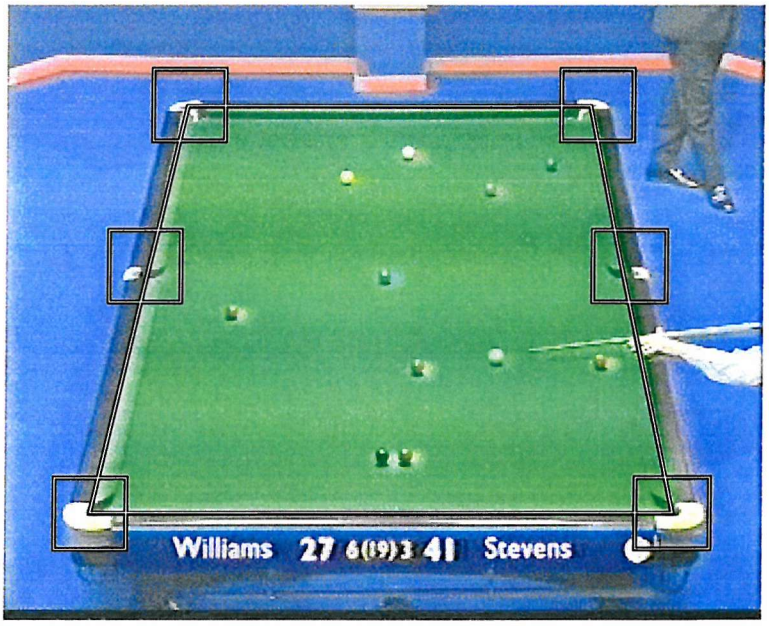


Figure 5.8: Results of locating pockets.

It must be noted that the technique of finding pockets is based purely on the prior knowledge of snooker tables, and that no image processing is performed. The technique relies on the robust detection of snooker tables, and the computational benefits are obvious in the PocketLocator SCD.

Interestingly, when a database of 'pocket images' were generated from the

PocketLocator SCD for performance evaluations, human observers were unable to recognise the images as pockets. Once an image of a snooker table was shown, the recognition by the human subjects improved. This may have implications for a general pocket visual detector which attempts to find pockets based purely on visual appearance. The idea of using *easily* detectable objects (the table) and combining them with prior knowledge to generate extra semantics is clearly demonstrated in this case. Figure 5.9 presents some examples of pocket images whose location is predicted by the PocketLocator SCD from the table model.

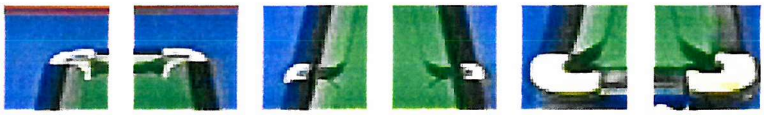


Figure 5.9: Examples of extracted pockets.

5.4 Snooker Ball Location

This section presents the technique for snooker ball detection from top-view scenes. The process has two primary stages of visual analysis. The first stage performs preliminary analysis on video frames in which candidate ball locations are identified. The second stage verifies each location to confirm or deny the presence of balls using a neural network classification technique.

Before describing in detail the techniques for ball location and verification, methods for detecting balls are first discussed. A ball by definition is spherical, and on a 2D plane it approximately projects to a circle. One can assume that the detection of snooker balls involves some form of circle detection technique, e.g. the circle Hough transform. The circle Hough transform is a computationally intensive technique, but it is a possible candidate for detecting snooker balls. One of the parameters in the Hough transform is the radius of the circle, and since snooker balls are within a known range, for top-views computation efficiency may be gained. However, the snooker balls in top-view scenes are tiny which discourages the use of the circle Hough transform. The major factors for not using

the Hough method are the intensive computation required for precise location of balls, and the low definition of the balls.

Jebara *et al.* (34) presents a wearable-based augmented reality system for enhancing the game of billiards. Their system, Stochasticks, locates billiard balls for the purpose of suggesting shots and assisted targeting through a head mounted video unit. The detection of billiard balls is a two stage process where candidate balls are found and then they are classified. The location of candidate balls relies on a circle detection technique based on the Generalised Symmetry Transform. The recognition and classification use probabilistic colour models of billiard balls.

The next two Subsections 5.4.1 and 5.4.2 present the processes involved in the location of balls.

5.4.1 Ball Prediction

The initial stage of detecting balls begins with estimating likely ball locations. This process is confined to the area defined by the top-view table SCD-Object. The premise of the technique used to isolate balls is based on the observation that snooker balls are filmed with a high degree of specular reflection (the highlights).

Figure 5.10 shows a snooker table with magnified images of the balls. The balls themselves are not very distinct for reasons explained in Section 5.2, and they occupy a very small pixel grid area. However, the highlights of snooker balls are less affected by the clarity, and they offer a consistent and detectable feature.

Figure 5.10: Magnified views of snooker balls.

Highlights in top-view snooker video appear as tiny regions of high intensity pixels, e.g. pixels of maximum whiteness. The visual processes for finding highlights that are carried for a single frame of video are as follows:

1. Extract the area of table from given video frame.
2. For the table area, locate regions of high intensity.
3. Store all high intensity regions which have the dimensions typical of a snooker ball highlight.

Figure 5.11 presents a typical result for the detection of candidate balls for the frame shown in Figure 5.1. The figure indicates detected *highlights* with cross hairs. The frame shows that all the highlights belonging to the balls are found, but there are detected highlights which need to be eliminated. The positions of the *highlights* provide information for the next stage of processing which verifies whether or not the highlights belong to a ball. The verification process is presented in the next subsection.

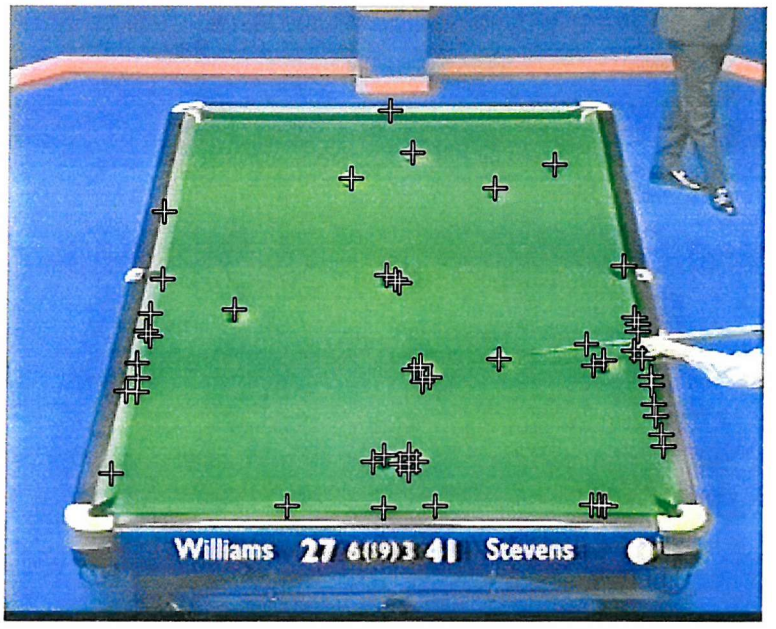


Figure 5.11: The first stage of locating candidate balls.

5.4.2 Ball Verification

The previous stage generates candidate ball locations by searching for highlights. The process is restricted to the table area which improves the processing time, and reduces the number of irrelevant candidate locations. However, objects such as the cue, the referee's head etc, can give off distinctive highlights. The next stage of processing aims to identify the detected highlights that belong to balls.

The premise of the technique is based on appearance matching. In this case the local areas around candidate highlights are compared for similarity to local scene archetypes of snooker. Rickman *et al.* (54) used Self Organising neural networks to learn scene archetypes for a content-based retrieval application, where similar scenes in a video are identified from a given target scene. The technique of ball verification, in this case, uses the feature extraction mechanisms of a fully interconnected MLP network. The decision to use an MLP neural network rather than Rickman's use of Self Organising neural networks is due to the immediate availability of the code which allowed some initial experimentation to be performed.

The neural network is configurable to have an N_a by N_b node input window to N_l hidden layers each of which can have $N_{l,c}$ nodes. Scenes are to be classified as either containing balls or not containing balls by the neural network. The neural network described in this chapter has a 12 x 12 node input window connected to a 50 node hidden layer which is connected to a single node output layer. The dimensions of the input window is based on the pixel dimensions of snooker balls where each pixel corresponds to each input node. The size of the input images is defined such that it captures the whole ball. The minimum size is found to be 12 x 12 pixels.

The procedure for the collection of training patterns begins with capturing 12 by 12 pixel images from the corresponding video frames of candidate highlights. Two hundred and ninety examples of 12 x 12 sub-images, of which one hundred and thirty are exemplar ball scenes, formed the training set for the neural network, and each subimage is manually labelled as either *is a local ball scene* or *is not a local ball scene.*

The training examples are collected from the location information of highlights generated in the previous stage where candidate highlights may belong to balls. The centre of the captured sub-images corresponds to the centres of highlights. The subimages are normalised by applying greyscale conversion and histogram equalisation. Figure 5.12 shows examples of captured ball scenes before normalisation. The neural network is trained on the collected data set, and it is then tested on a test set of video frames. Miss classifications are collected and added to the training set, and the neural network is retrained.

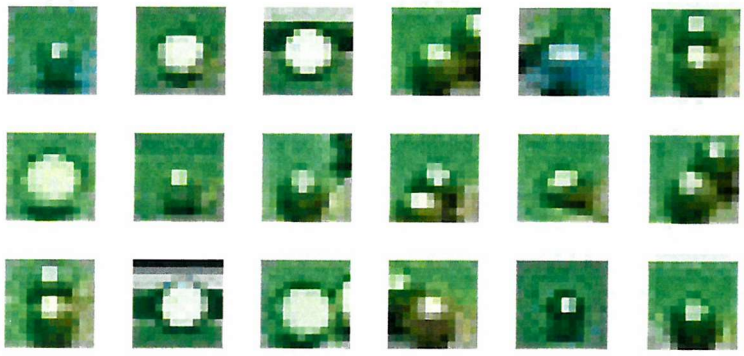


Figure 5.12: Archetypal scenes of balls.

The classification procedure of candidate highlights using the trained neural network is summarised as follows:

1. Extract 12 by 12 pixel images from the corresponding video frame of candidate highlights
2. Normalise (greyscale, and histogram equalise).
3. Map normalised image to neural network inputs.

The classification performance of the trained neural network is tested on a set of 11,000 previously unseen sub-images. Table 5.1 shows a 99% success rate for correctly classifying balls samples and a 98% success rate for correctly classifying non-ball samples.

Classification	No. of test samples	No. of correct classifications	No. of false positives
Balls	6684	6644	80
Non-balls	4293	4213	40

Table 5.1: The classification performance of the ball neural network.

Within the SnookerBall SCD, the neural network classification represents the last stage of vision analysis. After classification, SnookerBall SCD-Objects are created and stored in the SCD Correlator. The SCD-Objects represent the shape of balls as an eight point polygon which approximates a circle. The size of

the polygon is fixed because the visual shape variation of balls in snooker video is nearly zero, and the perspective effects on the size are minimal. Experiments with the snooker ball SCD are presented in the next section.

5.5 Experimental Results and Evaluation

This section presents experimental results and evaluations of the VCR System which uses the snooker SCDs. Results which demonstrate the different aspects of the VCR System are presented and are as follows:

- Recovery of objects.
- Motion correspondence.
- Querying using SCDs and the detection of events.

The data set used for these experiments consists of 10 video clips. It must be noted that the results are generated in near real-time, and faster than real-time processing is possible if the software is optimised.

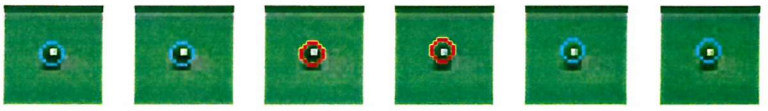
Before progressing to the next subsection, the nomenclature used in the presentation of figures is explained. The generated results are overlayed on their corresponding video frames and they are cropped to show the areas of interest. Video frames of sequences are given letters to denote the order in which they are displayed, or otherwise the order is assumed to start from left to right and top to bottom.

5.5.1 Recovery of Objects

Chapter 3.3.1 describes the recovery of objects by interpolating missing SCD-Objects across frames. The recovery of objects is possible if frame(s) that should contain the missing object(s), f_n to f_m, have object correspondences in frames f_{n-1} and f_{m+1}. This subsection presents some visual results which focus on the

recovery of undetected snooker ball objects. The process to predict where missing objects should have been located is performed by the SCD Correlator. The SCD Correlator proved to be invaluable for the detection of objects such as the snooker balls within the video clips.

A typical result for the recovery of objects is presented in Figure 5.13 which shows a stationary snooker ball. For the Frames 5.13(a, b) and 5.13(e, f) the snooker ball is detected as indicated by the light coloured polygons overlaid on the video images. Frames 5.13(c, d) represent predicted snooker ball positions, and the results are correct in that it is visually faithful to the actual location of the misdetected snooker ball.

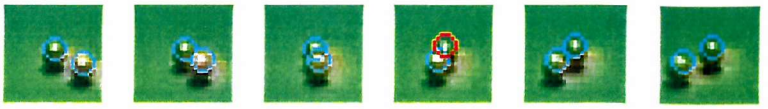


(a) (b) (c) (d) (e) (f)

Figure 5.13: Recovery of missed detections (image c and d) for a stationary ball.

The reason for the missed detections is because the highlights are not registered which leaves nothing for the SnookerBall SCD to lock on. *Noise* in the video can cause localised variations in intensity, and in this case the intensity is reduced below the fixed threshold which determines whether regions are candidate highlights. Although some missed highlights could be revealed by lowering the threshold, the SCD Correlator has overcome the problem successfully without this being necessary.

One of the problems in computer vision is the occlusion of objects. Figure 5.14 shows a snooker ball in the process of being occluded by another ball. As the occlusion takes place in Frames 5.14(b, c, d), a miss detection in Frame 5.14(d) is correctly predicted. The same reasons as discussed for the previous result also contribute to the miss detection.



(a) (b) (c) (d) (e) (f)

Figure 5.14: Recovery of occluded balls.

Figure 5.15 shows another miss detection of a ball in motion. Frame 5.15(c) is the frame in which the SnookerBall SCD failed to register the presence of the ball, but the position is correctly predicted by the SCD Correlator.

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

Figure 5.15: Recovery of a miss detection of a snooker ball object in motion.

The aim of the recovery process is to interpolate SCD-Objects such that the positions of missing objects are correctly predicted. The performance of the recovery of ball positions may be measured by counting the number of correctly positioned predicted SnookerBall SCD-Objects. The positions are manually confirmed by visual inspection of graphical representations of SnookerBall SCD-Objects such as those shown in Figures 5.13, 5.14 and 5.15.

For the video test set over 32450 SnookerBall SCD-Objects are generated, of which 227 are predicted SnookerBall SCD-Objects. For evaluating the accuracy of the recovery, only the sequences representative of actual balls are considered and in this case this amounts to 185 SCD-Objects. The outstanding 42 SnookerBall SCD-Objects are recoveries due to the SnookerBall SCD submitting false positives to the SCD Correlator, which represents a very low 0.13% *error*.

Of the 185 SCD-Objects 180 are considered to be correct because they correspond visually to the ball locations in their corresponding video frames. Examples of some of the rejected recoveries are shown in Figure 5.16. The figure shows that misalignment of the SCD-Objects occur when a large amount of acceleration (indicated by the blurring of the snooker ball) is present. This may be improved by

using an acceleration motion model in the SCD Correlator.

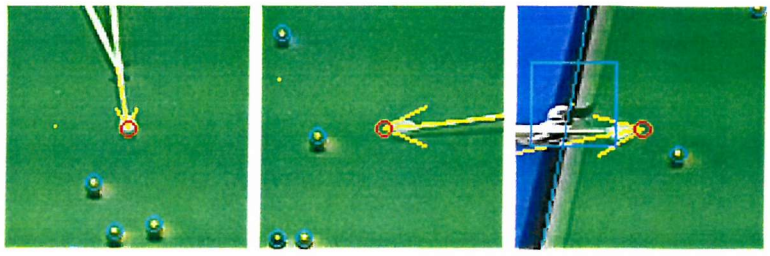


Figure 5.16: Examples of mispredicting SnookerBall SCD-Object locations.

The results in this subsection only show the recovery of objects, but the results do not indicate the motion correspondence from object to object. The next subsection will present experimental results on the tracking of balls.

5.5.2 Motion Correspondence

This subsection presents experimental results of linking SnookerBall SCD-Objects across frames which creates motion trails of objects. This is performed by the SCD Correlator which was described in Section 3.3. The motion tracking is the result of matching balls with their corresponding new positions in future frames performed automatically in the SCD Correlator. Connecting objects across frames creates temporally invariant object entities which may be referenced independently.

Figure 5.17 shows a player striking the cue (white) ball down the table. As the ball moves into different positions over each frame, an arrow indicates the direction and magnitude of the motion.

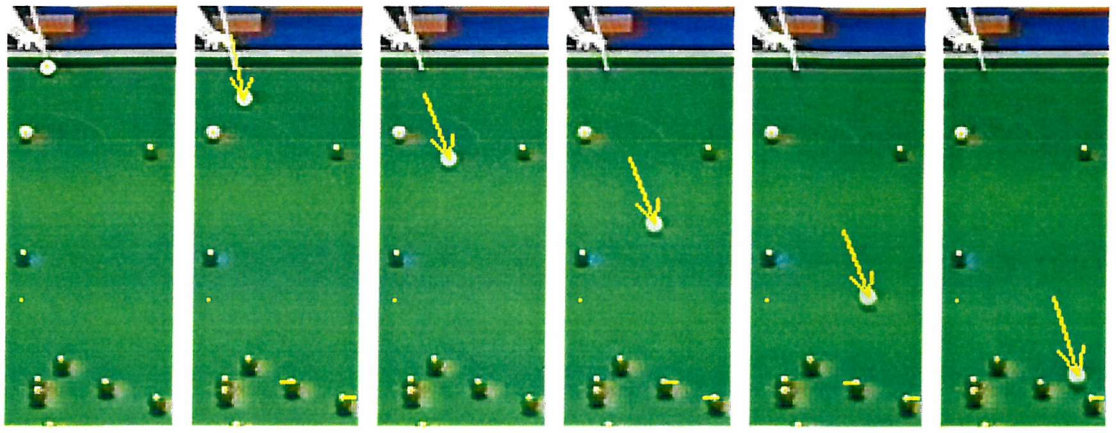


Figure 5.17: Motion of a single ball.

Figures 5.18, 5.19, 5.20 show a ball striking another ball which produces two or more motion trails. The sequence in Figure 5.18 shows the cue ball being struck towards a target ball in the centre. After the collision at Frame 5.18(d) the cue ball changes direction moving down the table. The small motion trail of the cue ball indicates that much of the momentum is lost after making contact with the target ball. The target ball moves off after the impact into the left middle pocket. Two unique motion trails are correctly produced.

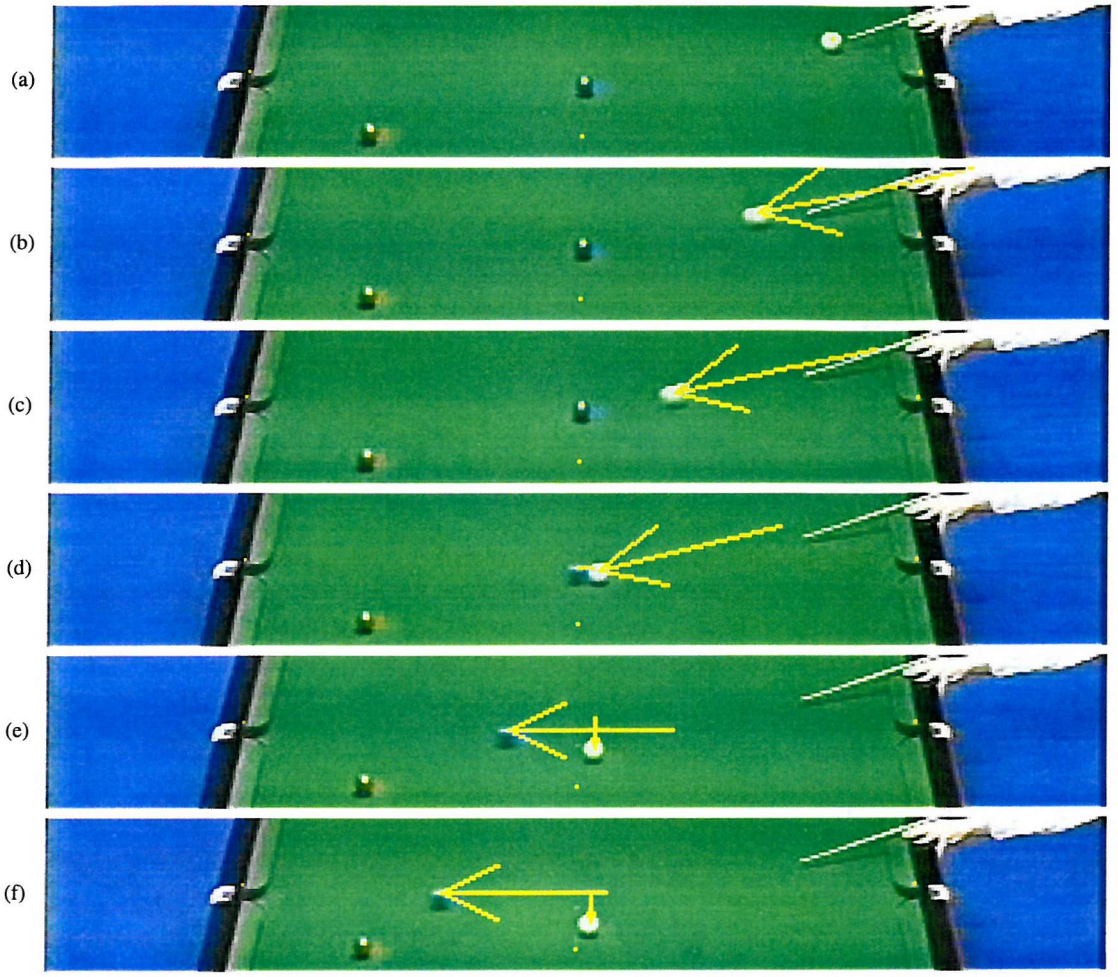


Figure 5.18: Motion of a ball striking another ball.

Figure 5.19 shows a pronounced result of tracking two balls whilst their direction of motion changes.

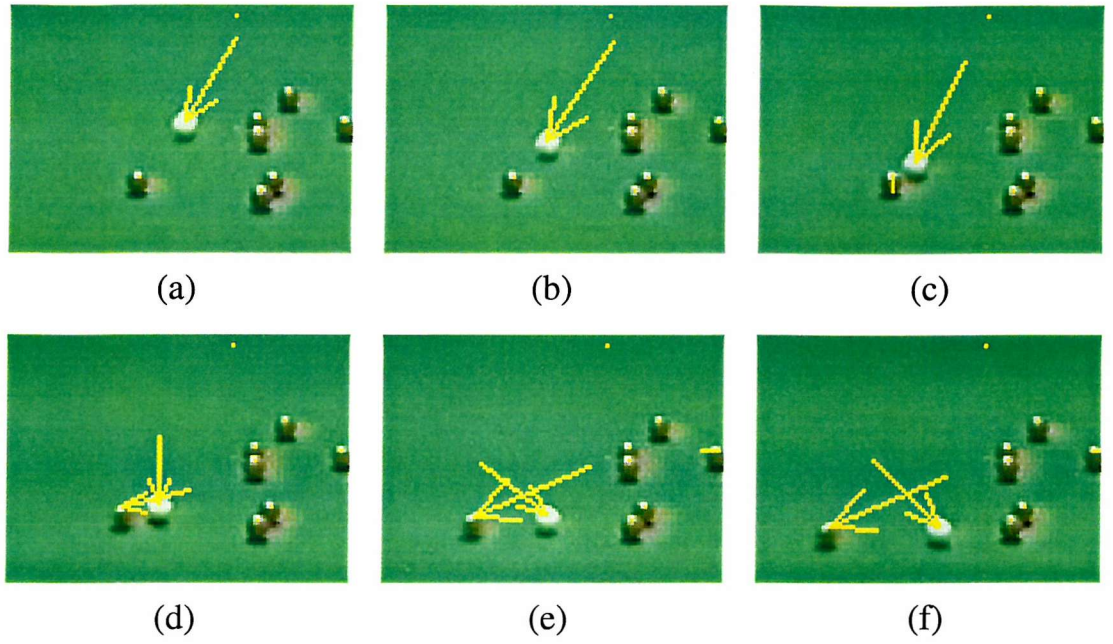


Figure 5.19: Motion of a ball striking another ball.

Figure 5.20 shows an interesting result of the cue ball travelling towards and colliding with a target ball. The resulting deflection of the cue ball moves in the opposite direction to its original motion. Both balls are correctly tracked.

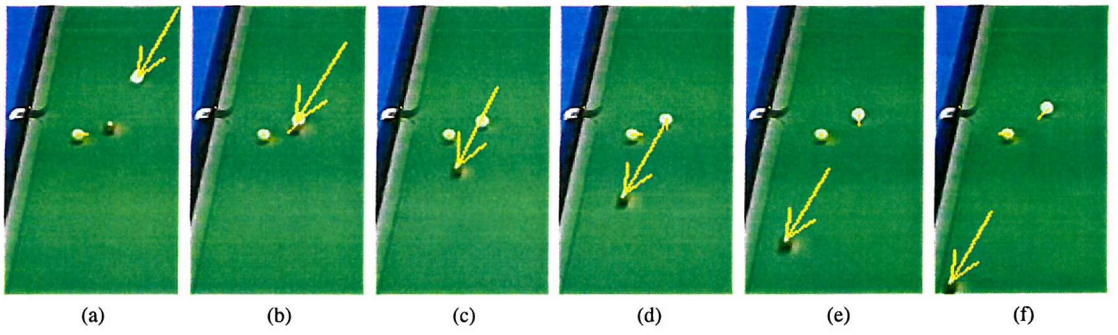


Figure 5.20: Motion of a ball striking another ball with rebound in the opposite direction of the original travel.

Figure 5.21, presents an elaborate result of multiple collisions between balls. Frames 5.21(a) and 5.21(b), shows the cue ball travelling towards the first target ball which results in a collision. The cue ball is then deflected towards another

ball, as shown in Frames 5.21(c), 5.21(d), 5.21(e), which in turn hits another ball. The rest of the frames show the motion trails of three snooker balls.

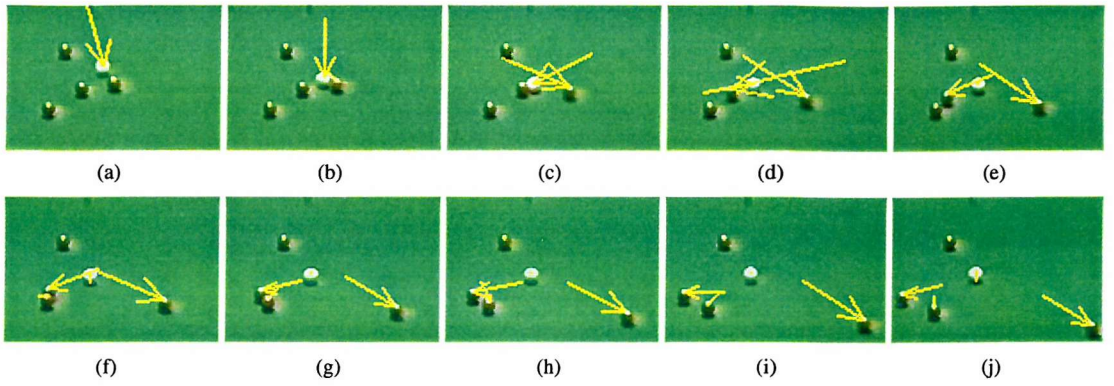


Figure 5.21: Motion tracking of balls in multiple collisions.

The motion correspondence reflects how accurately the SCD Correlator can match objects from one frame to the next. The physics of billiards is well understood, and it may be used to model the motion by predicting more accurately the likely positions of where balls should travel. However, one of the design aims of SCD Correlator is to be as generic as possible and avoids the inclusion of physics for specific situations at the fundamental design level.

The objects that the snooker framework can detect are balls, tables, and pockets. The set of videos contained a total of 202 detectable objects. To evaluate the performance of the linking process one would need to track and observe every single *physical* object frame-by-frame in each of the videos. The observer will have to visually determine whether or not the generated SCD-Objects match the physical objects in terms of temporal coherence, e.g. if $O_{i,j}$ represents object i in frame j, can this object type be referenced as the same object across all frames for which it exists within the shot, i.e. $0 \leq j < F_{max}$ where F_{max} is the number of frames in the shot. For this to be achieved a significant effort in terms of time and user interface design for distinguishing unique video-objects is required.

An alternative method which will give an indication of the performance is to count the number of unique IDs generated for each object type. The ethos of this

approach is that for n objects in a video shot there should ideally be n number of uniques IDs generated by the SCD Correlator. Referring to Section 3.3.1 an ID is used to reference an object independent of its frame index, thus an object may be referred as the same object anywhere within the same shot.

Table 5.2 displays the number of unique IDs generated for each object type for the video test set. A total of 220 unique IDs are generated, where the most accurate results are for the table and pocket objects. The actual performance of the SCD Correlator for linking each table in each frame for each video is shown to be 100%, and likewise for the pockets. As for the balls there are actually 132 different ball video-objects but the SCD Correlator reported 150 unique instances. The figure of 150 contrasts with the figure for when the SCD Correlator has no filtering mechanism, see Section 3.3.1, which would bring the total up to 254 unique instances. Another aspect to the SCD Correlator is that without the recovery mechanism a higher number of unique instances would occur for the same object.

Unique Ball IDs (Actual)	Unique Table IDs (Actual)	Unique Pocket IDs (Actual)
150 (132)	10 (10)	60 (60)

Table 5.2: The number of unique object IDs generated by the system. The actual number of objects are represented by the number in the brackets.

The higher than expected number of unique instances is not due to the SCD Correlator but an explanation can be provided when the behaviour of the SnookerBall SCD is examined. The SnookerBall SCD simply detects snooker balls in each frame and submits an abstract representation of the objects in the form of SCD-Objects to the SCD Correlator. The SCD Correlator automatically associates SCD-Objects in one frame to the next which creates temporally invariant SCD-Object entities. However, in this case the correlation process generated *extra* IDs in two ways:

1. Spurious SnookerBall SCD-Objects submitted to the SCD Correlator. These

are commonly seen on the cues of the players.

2. Multiple IDs are generated for the same objects in situations where they are detected, then undetected for a significant number of frames, and then are detected again.

The first case may be addressed by retraining the network classifier within the SnookerBall SCD with more non-ball examples, and it is true to say that not all possible exemplars have been accounted for due to the large number of possibilities. Spurious SCD-Objects are characterised by their brief existence and it is possible to tune the SCD Correlator to reduce the number of false positives. (See L_{min} in Section 3.3.1).

The second case may be addressable by increasing the number of buffers within the SCD Correlator such that correlation may be carried across a larger number of frames. However, a more productive approach is to use better quality and higher definition snooker videos, and it is anticipated that there will be further robustness of object detection, thus better tracking of objects.

Resolving the temporal correspondence of snooker balls in one frame to another creates motion paths which relate to individual balls and creates unique video-objects. The elimination of the temporal component of objects in video makes the manipulation of the objects a *simpler* task. The next section focusses on the querying aspect of the VCR System.

5.5.3 Querying

The querying aspect of the VCR System is implemented in the SCD Manipulator. Some experimental results in Chapter 4.4.3 show the querying of the colour of faces. In this subsection, experimental results of querying the colour of snooker tables, and snooker balls are presented.

In Chapter 3.6 it was suggested that the use of a Prolog engine might form the SCD Manipulator Core. However, due to time and software constraints, the

current implementation of the VCR System implements queries by coding them in C++ in the SCD Manipulator. It is worth noting that the object based approach provides for a clear implementation of querying logic. Integrating the system with a more flexible querying interface is left as a future extension. The current implementation of queries nonetheless demonstrates the querying principles of the system.

A query programmed into the SCD Manipulator posed the question "What colour is the snooker table?". The program fragment which expresses this is shown below:

```
// For the SCD-Object, see if it represents a snooker
// table, if so, report the colour.
if( IsTable(anSCD) )
{
        // Perform colour recognition.
        SCDColourRecogniser.Look(anSCD);

        // Report the result.
        printf("The Colour Of the Table is %s\n",
                anSCD->GetColour());
}
```

In detail, the query uses the predicate IsTable() which tests to see if a given SCD-Object represents a table. If a table is confirmed, the Colour Recognition SCD is called which calculates the colour (*SCDColourRecogniser.Look(anSCD)*). The result of the query proved to be a success as the correct colour of green is reported for all frames in all videos.

Unfortunately, querying the colour of snooker balls cannot produce accurate results. The reason is due to the very small number of pixels which make up balls. Small numbers of pixels means that objects are likely to suffer from aliasing effects which contaminates pixels with colours close-by. In the case of snooker balls, the

green colour of the table affects their colour clarity. The avenues that may be explored for accurate colour recognition of snooker balls include:

1. Higher quality compression to reduce the onset of artefacts.
2. Obtain a higher resolution version of the videos for increasing the number of pixels which make up snooker balls, thus improving the definition, e.g. HDTV, broadcast quality recordings.
3. Create a specialised colour recognition SCD which models the colour properties of snooker balls.

The consequence of being unable to identify the colours of balls means that high-level queries such as: "List all red ball pots", and "Identify foul shots", "Identify the white ball" cannot be realised on the video clips. However, the detection of events such as "Indicate when a ball is potted" is possible which is explained below.

The query for the detection of potted balls is expressed as: if a ball enters a pocket area, and disappears then declare a pot. Shown below is pseudo-code which expresses the query:

```
// Identify a ball that is within the vicinity of a pocket.
if( !HaveWeLockedOnToABall &&
    IsThereABallInTheVicinityOfAPocket() )
    Ball_ID = ID_OfMovingBall;

// Is the ball moving
if( IsTheBallMoving(Ball_ID) )
       HaveWeLockedOnToABall = True;
else
       HaveWeLockedOnToABall = False;
```

```
// See if the ball is potted.
if( HaveWeLockedOnToABall && DoesBallStillExist(Ball_ID) )
{
        // Ball has moved out of pocket area.
        if( !IsBallInVicinityOfPocket(Ball_ID) )
                HaveWeLockedOnToABall = False;
}
else
{
        // A pot has been detected.
        IndicateThatBallIsPotted();
        HaveWeLockedOnToABall = False;
}
```

The code initially identifies a moving ball that enters the vicinity of a pocket. A pot is declared if the ball is eliminated, e.g. it does not exist because it is not detected. If an identified ball leaves the vicinity of a pocket then the identified ball is released. The code only functions because objects may be referenced as individual unique entities across video. The need for unique identification of objects across frames contributes to the importance of the SCD Correlator for video analysis.

For the purposes of evaluating the querying performance, a larger test set consisting of 76 snooker clips is used. Table 5.3 lists the results of the query applied to the larger video test set which contains potted shots (72) and non-potted shots (4). The table shows that there are 72 clips which are potted shots, and sixty out of the seventy two are correctly detected. This represents a success rate of 83%. The video clips which show non-potted shots have a 100% success rate where all non-pot shots are correctly confirmed. However, there are a number of false positives and missed detections.

No. of Actual Pots	Correctly Detected Pots	False Positives	Missed Detections
72	60 (83%)	22	12

Table 5.3: Results of detecting potted shots.

There are two main causes for the false positives. The first is the spurious occurrence of snooker balls due to noise effects appearing on the table which then migrate towards pockets. The second main cause is due to the player's cues. As the players prepare or finish their snooker shots they sometimes sweep their cues over the pockets. If the highlights of the cues are mistakenly tracked as snooker balls then pots may be falsely detected.

The missed detections can be traced to the SnookerBall SCD's neural network which failed to register the potted ball to the SCD Correlator. The videos in question showed that the potted balls are flush with the lower border of the table, and the training set did not contain this particular case. Another common cause for missed detections is due to the clarity of the balls in the video clips. If the balls travel too fast then the balls will appear too blurred, thus making them difficult to track. This highlights the need for high-quality video. The SCD Correlator can compensate for these cases but the required conditions for complete recovery do not occur. It must be noted that the false positives and missed detections may be reduced with further training of the SnookerBall SCD's neural network and higher quality video.

Figures 5.22, 5.23, 5.24, and 5.25, presents the results of correctly detected snooker pots, where the frames leading to the pot begins from the left. The images are generated by instructing the SCD Manipulator with the query "Indicate when a ball is potted, and save the last three frames to disk leading to the pot".

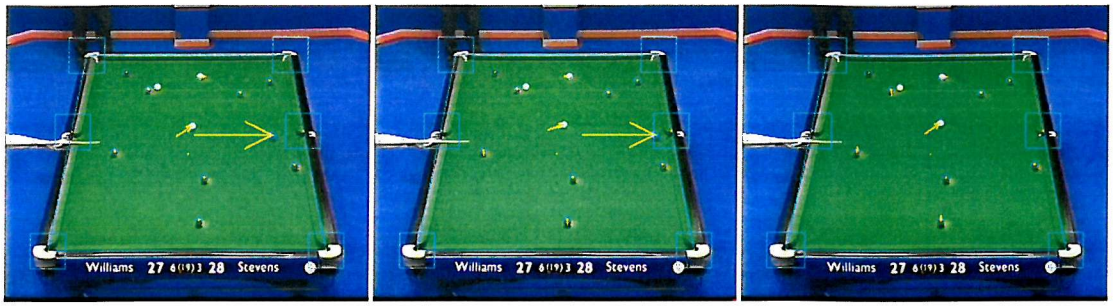


Figure 5.22: Detected potted ball event (a).

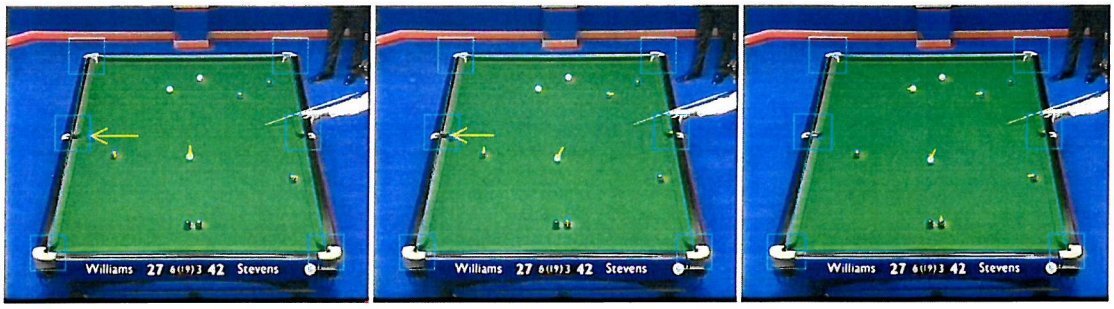


Figure 5.23: Detected potted ball event (b).

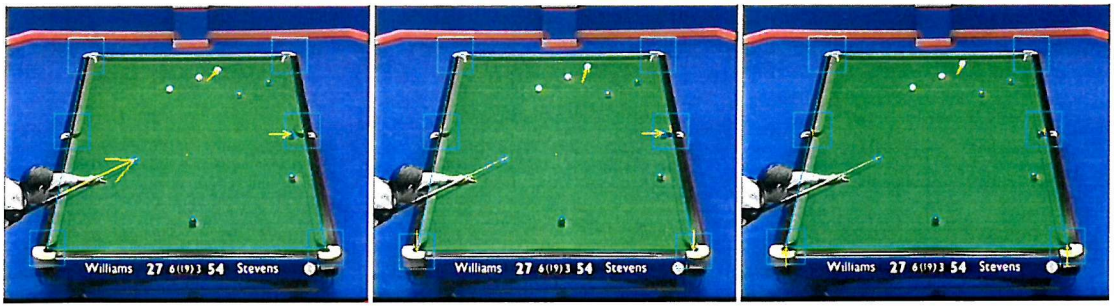


Figure 5.24: Detected potted ball event (c).

Figure 5.25: Detected potted ball event (d).

Figure 5.26, presents an image sequence of a snooker clip where a ball is struck by the cue ball and is successfully tracked as it rebounds twice off the cushions of the table before disappearing into the middle pocket. A potted event was correctly identified in the clip.

Figure 5.26: Selected frames from a video clip showing moving balls being tracked with one of the balls rebounding off the cushions into a pocket.

The high-level querying aspect of the VCR System stems from the fact that videos are analysed for objects. For a multimedia video database application processing video on the fly is not the best approach, but rather processing an intermediate form, e.g. feature vectors, of the video is more computationally efficient. The SCD-Objects which are filtered through the SCD Correlator and which are ultimately fed into the SCD Manipulator can be stored on disk and reloaded to be used as an intermediate format for rapid analysis and querying in video applications.

The SCD-Objects represent *feature vectors* in video. The unique aspect of the VCR System is that stored SCD-Objects can be used to regenerate video in the manner the SCDs *saw* the video. The capability to regenerate the original video from the extracted objects and their semantics reflects the quality and completeness of the representation. The next subsection evaluates users' perceptions of SCD-Object representations by exploring the visualisation aspects of stored SCD-Objects.

5.5.4 Visualisation

The Introduction of this thesis used the statistics of features, such as the rate of occurrence of shot breaks to infer the type of events happening within video, as an example to justify the case for deriving objects which represents more concrete and meaningful processing handles. The derivation of higher semantics using objects, as demonstrated in the previous section, justifies the level of semantic coarseness.

Often, features generated from video in multimedia applications cannot be used to determine high-level events such as the detection of pots demonstrated in the previous section. The link between users and the VCR System is the high-level interaction through queries. The enabler of this link is the unique type of data that is generated by the VCR System, which allows sophisticated and powerful formulation of queries.

The SCD-Objects generated by the VCR System can potentially be used by content-based retrieval systems or video analysis applications. Assuming that the SCD-Objects are representative of the original video, then the SCD-Objects are immediate processing handles in which rapid content-based analysis may be performed. Another aspect of SCD-Objects is visualisation. Assuming that the SCDs generate the elements faithful to the video source then the resulting SCD-Objects can be used to 'recreate' the original video content. Given that the assumption is true for all video within a chosen domain then one can state that SCD-Objects are representative of the video source. An evaluation of this is provided in this sub-section. But first the visualisation aspect is explored.

SCD-Object representations of snooker videos generated by the VCR System may be stored in separate files, and they can be *replayed* to create an artificial video. Objects in each frame of video are represented by 'object-type' SCD-Objects which contain enough information to recreate the basic outline of the objects in their detection positions. Replaying SCD-Object representations of video involve drawing their data representation of objects into their respective frames. The frames are then displayed in succession to create the illusion of video.

This experiment involves showing users synthetic video generated from SCD-Objects but reconstructed in four different ways which are all visually different but intrinsically identical. Figures 5.27, 5.28, 5.29, and 5.30 present the four types of video reconstruction. Figure 5.27 shows a wire-frame reconstruction of the video, and it is completely recreated from the SCD-Object representation set itself. Figure 5.28 shows the same video but the objects are drawn to have a solid appearance by filling them with different intensities. Figure 5.29 is equivalent to Figure 5.28 but with colour information added to the objects. However, the typical colours for all objects could not be reproduced because of the Colour Recognition SCD being unable to resolve the colour of snooker balls. Finally, in Figure 5.30 realistic texture maps are used to render the SCD-Objects. The correct texture mappings are determined by the semantic labels relating to each

SCD-Object. Figure 5.31 shows the different texture maps used for the rendering to reconstruct a more realistic view of the video.

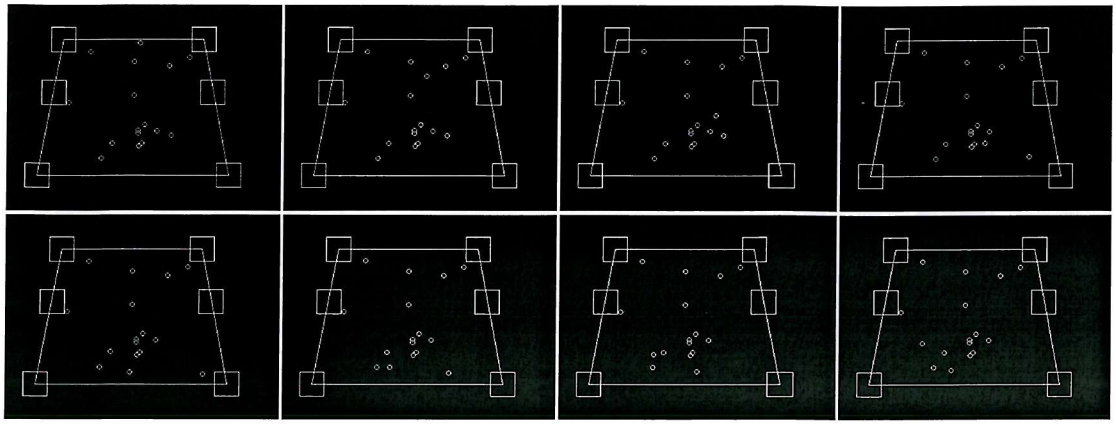


Figure 5.27: A wire-frame reconstruction.

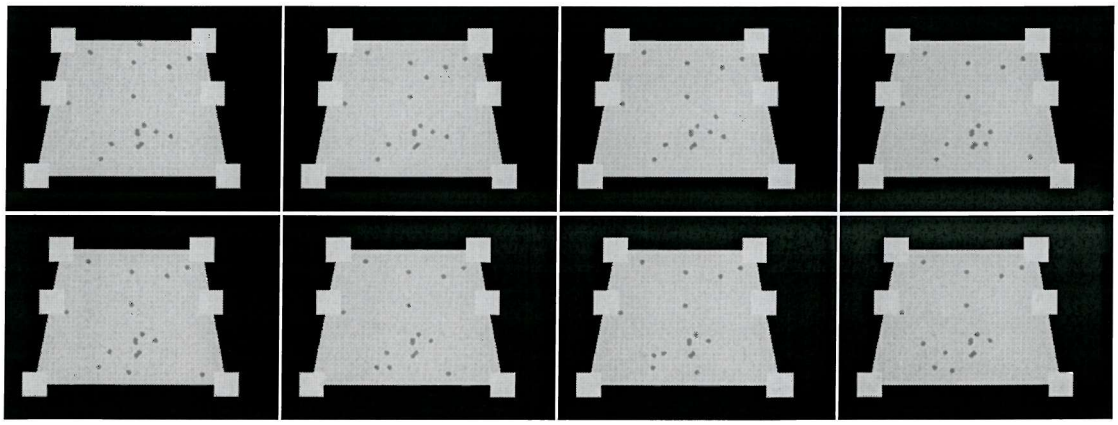


Figure 5.28: A monochromatic reconstruction.

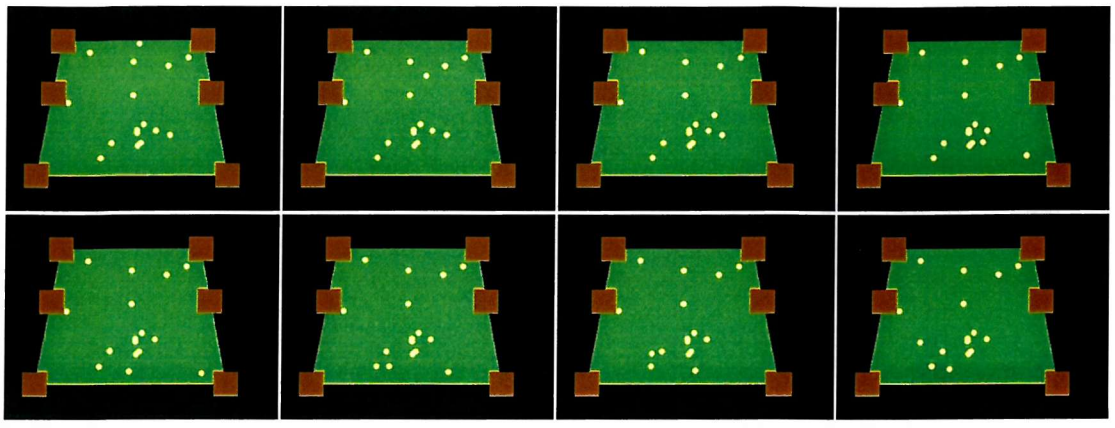


Figure 5.29: A coloured reconstruction.

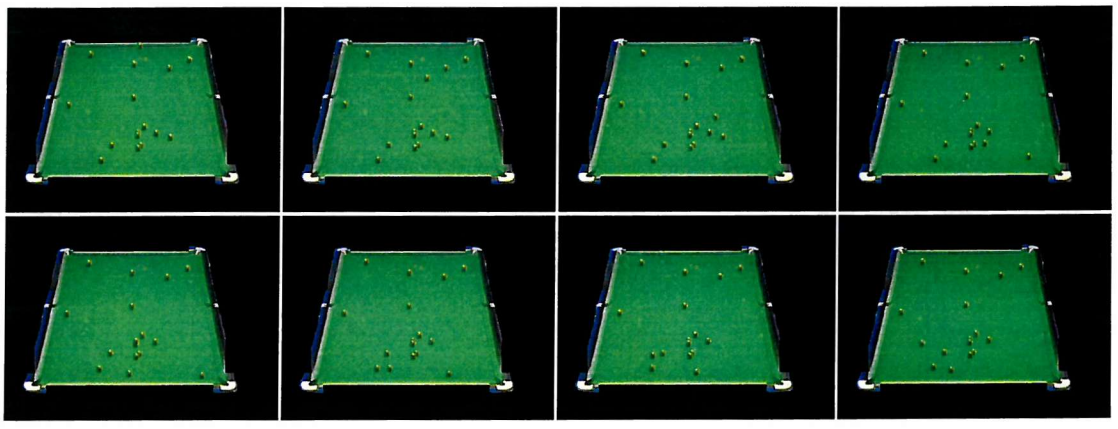


Figure 5.30: A texture-map reconstruction.

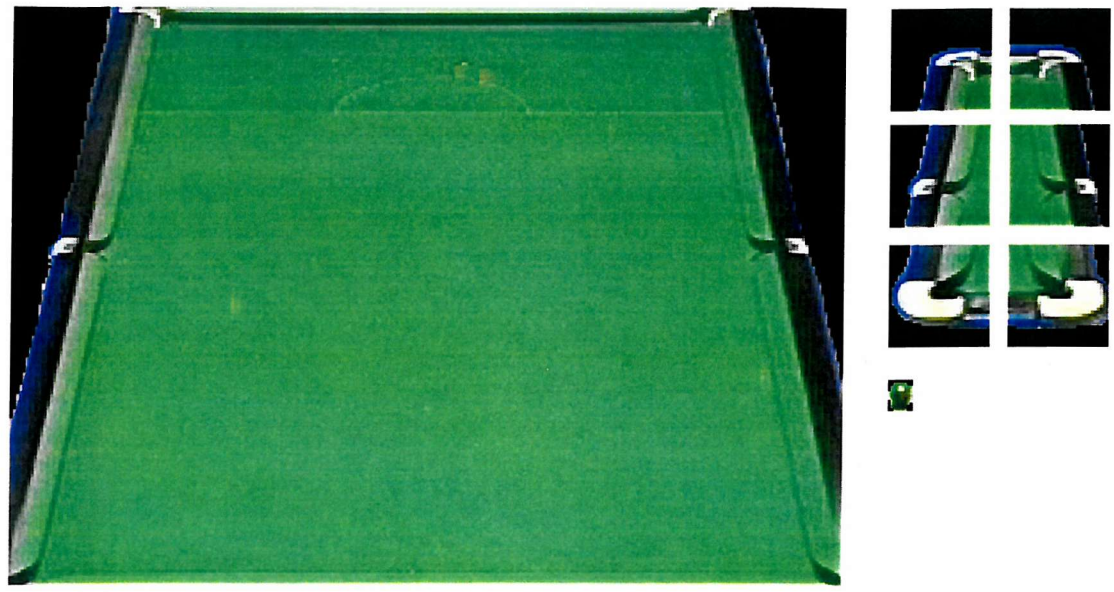


Figure 5.31: The texture maps used for realistic video reconstruction.

The different video reconstructions are designed such that the first video shows a minimal amount of visual data compared to the last which is the most visually realistic. The intention of showing four different videos to users is to allow them to rate their confidence in what they recognise in each of the videos. The procedure for this evaluation is explained as follows:

1. Make sure that the candidates are not aware of the work. This avoids sub-conscious/ conscious suggestion that will bias the results.

2. Explain to the candidates that four videos are shown to them, and that each video is shown in the order of least visually detailed to the most visually detailed.

3. Show the wire frame video reconstruction.

4. Ask the candidates if they recognise what the video was showing, and if so, ask them to give a rating between 1 for least confident to 10 for very confident. The rating of 0 is reserved for totally unrecognisable.

5. Repeat from step 3, but showing the monochrome, coloured, and texture-map reconstructions, respectively.

Before this experiment was conducted, the expected and predicted outcome of the experiment is a general trend of increasing confidence values from the wire-frame to the texture-mapped video. The idea being that, as more visual information is introduced the higher the confidence, e.g. simple wire frame objects, to solid filled objects, to solid filled objects with *representative* colours, and to realistic texture maps of objects. Table 5.4, presents the outcome of the evaluation.

Candidate	Wire framed	Monochrome	Colour	Texture-map
1	9	8	9	10
2	9	7	7	10
3	6	6	7	9
4	3	5	5	10
5	6	2	2	10
6	4	5	8	10
7	10	8	10	10
8	10	9	9	10
Average	7.1 ± 2.6	6.3 ± 2.1	7.1 ± 2.4	9.9 ± 0.3

Table 5.4: Table of confidences.

All the candidates correctly recognised the artificially generated video as either pool or snooker. The texture-mapped reconstruction is given the highest confidence. The average confidence values show that the Colour reconstruction ranked second place. The third and fourth rankings are the Wire-frame and Monochrome reconstructions respectively.

The outcome of the results may be explained by the human psychology of perception which is difficult to factor out in this experiment. However, the success of this visualisation experiment is reflected in the average confidence values, where high positive values are calculated for all video reconstructions.

The one question that remains is how can one tell if the synthetic video is truly representative of its associated video source? To answer this question a user evaluation is performed to determine whether or not a synthetically generated video can be matched to the original video based on the user's perception of similarity.

The user is shown a synthetically generated video that is constantly played back in a window on the screen. Another window allows the user to select and play back actual videos. The user watches the synthetic video to determine the associated real video which can be selected through the interface. The procedure is repeated for a total of 10 videos. A user trial showed that all the synthetic videos are correctly associated with the original videos which suggests that the stored SCD-Objects are highly representative of the original video source. See Figure 5.32 for a screen shot.

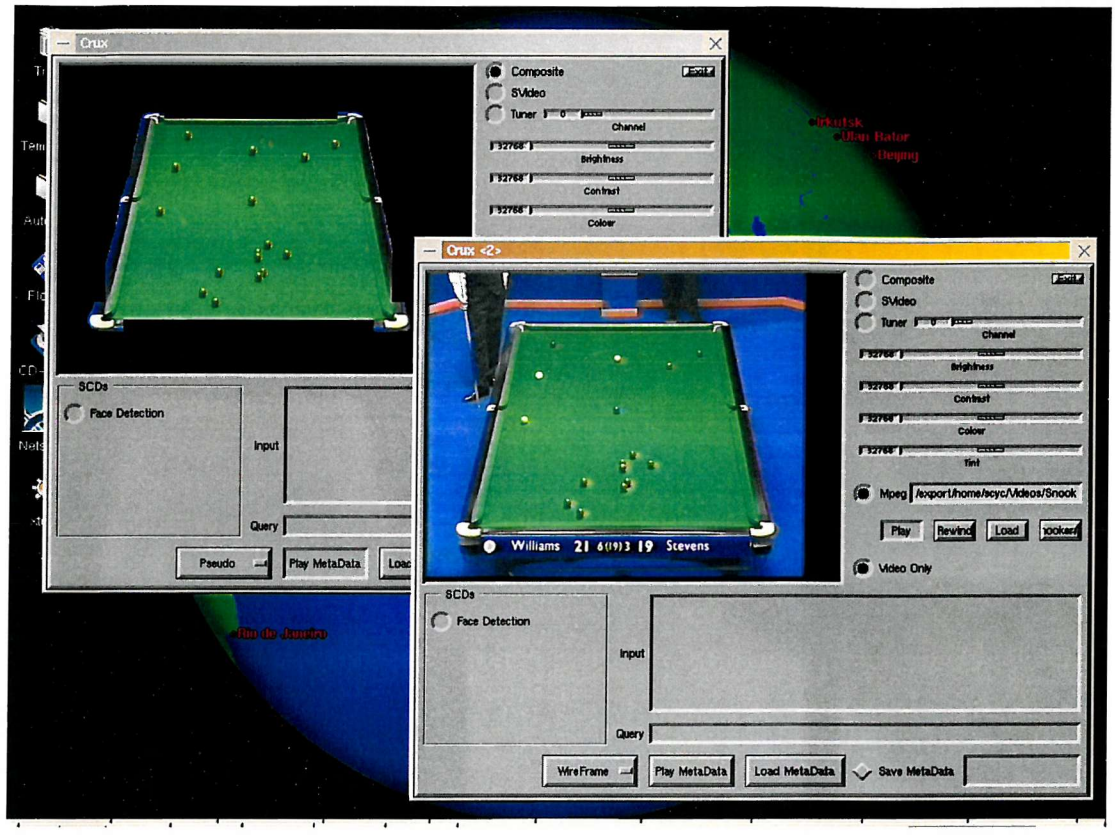


Figure 5.32: A screen shot of a user matching synthetic video to the original video.

5.6 Summary

In summary, this chapter has presented a snooker application developed in the framework of the VCR System. Experimental results and evaluations on parts of the VCR System have also been presented. This chapter has introduced and shown the development of the elements (the SCDS) which make up an SCD hierarchy for processing top-views of snooker video.

The SCDs presented include, SnookerBall for the detection of balls, SnookerTable for the detection of tables, and PocketLocator for the detection of pockets. Each SCD demonstrates a different aspect of processing video within the VCR System. The first aspect is the processing of raw video frames for objects such

as snooker tables. The second aspect uses knowledge to reduce the computational overhead for improving the performance of processing raw video frames for objects, e.g. assuming that snooker balls are always on the table, then the processing is confined to the table area. The third aspect builds on the knowledge of detected SCD-Objects for generating more objects, e.g. pockets are produced from the fact that they exist on the corners and the longer sides of snooker tables. The SCD hierarchy allows other SCDs to be integrated to handle other characteristic aspects of snooker videos such as anchor-person shots, side-view shots, player tracking shot, and close-up of balls shots.

The evaluation focussed on three important aspects of the VCR System, recovery of objects, motion correspondence and high-level based querying. The performance of recovering objects has shown to be 97% accurate for predicting the correct positions.

The motion correspondence was evaluated on the number of unique motion trails which gave an indication of the performance. A total of 220 unique video-objects were detected. However, the actual number of objects in the video was 202. The extra objects are due to spurious false positives and minor recognition errors, and these may be suppressed by further tuning of the VCR System.

The querying of the system for high-level events such as the detection of pots proved to be 83% successful, but a higher success rate is anticipated when the SnookerBall neural network classifier is trained on more snooker ball exemplars, e.g. cases where balls are near the borders. Querying is possible because the VCR System converts video into SCD-Objects which allows content-based manipulation. SCD-Objects may be stored on disk for later use.

The visualisation aspects of the SCD-Objects resulting from the system was presented where video was recreated in increasing degrees of *realism*. User trials to ascertain whether users could visually recognise the *contents* of the synthetic video were carried out. The results of the trials showed that all users could recognise the content of the artificially generated videos as the game of billiards.

Measuring how faithful the synthetic video was to the original video relied on a trial whereby a user is asked to match the recreated videos to the corresponding *real* videos. The trial produced a perfect match between the original and the synthetic videos.

In the final chapter the overall conclusion of the research will be presented together with suggestions for further work.

Chapter 6

Conclusions And Future Work

This research has been concerned with the innovation and development of video analysis techniques which support the extraction of semantics such as object and event identity from video. A new methodology for video analysis in which video is processed through a hierarchy of visual analysis modules has been proposed and developed.

The research began by tackling video content analysis primarily for content-based retrieval applications. Initially, the problem of face detection in video was tackled with the aim of establishing a correspondence of objects through video as a way to suppress false detections and to interpolate missed objects in the video was explored. A generic architectural framework for video content recognition was developed and tested using both the face detection work, and additional work on video segmentation and experimentation with object colour coding. A further evaluation of the approach was made using the analysis of snooker video as the test bed.

This chapter presents the main innovations of this research by summarising the main approach presented in Chapter 3 and the developments of the approach

in Chapters 4 and 5. A final section will identify areas for further research.

6.1 Conclusions

The overall aim of this research was to develop new techniques to analyse digital video for high-level content. The research began with identifying the need for high-level access to digital video content in multimedia applications. Unlike text, raw video data does not contain handles with which the underlying content may be accessed. The general approaches for generating handles to access video content in multimedia applications rely on the analysis of global statistical features, and the examination of regions. Although these approaches are used to infer high-level content, however, the granularity at which the meaning is conveyed may be too rich or too weak to be used effectively. To overcome the semantic predicament, the approaches are often constrained to a particular video genre which adds context to the generated content. The specific aim of this research was to explore the extraction of object-level content from video for increasing the versatility of applications and to improve the extraction of high-level semantics.

It was decided that the fundamental units from which higher-level content is to be inferred are objects. However, the inevitable consequence of this approach was the need for the recognition of objects in video. Since general object recognition is currently not possible, innovative ways for extracting objects from video were developed. The method for extracting objects is based on and motivated by success of constrained video analysis. One of the novel aspects of this research is the confinement of areas within video frames for improving the extraction of objects.

There are two stages in the extraction of objects in video. The first stage is a frame-based analysis of video where each video frame is processed for objects. Since video is inherently temporal, the first stage of analysis results in objects that are independent from one another across frames. Inorder to facilitate the effective manipulation of content, the detected objects must be uniquely identified

regardless of frame index. The second stage eliminates the temporal dependency from objects by associating corresponding objects across frames. A new and novel way for creating temporally invariant objects was developed as part of this research.

To support object oriented video analysis a prototype system, the Video Content Recognition (VCR) System, has been designed, implemented, and evaluated. The VCR System provides an infrastructure in which frame-based processing, performed by the SCDs, is seemlessly integrated with the necessary components, the SCD Correlator, for video analysis. The system is composed of four separate modules: Sensors, Specific Content Detectors (SCDs), an SCD Correlator, and an SCD Manipulator, and they are described in Chapter 3.

The SCDs are responsible for locating objects in video frames. SCDs are hierarchically ordered for selecting and decomposing video frames into regions for object analysis. The hierarchical nature of the SCDs has the potential to expand towards general video processing. The SCDs use a common structure called an SCD-Object for storing information in processed video such as objects and video events. To aid the process of visual analysis, the Sensors provide the basic image processing functionality. The SCDs are instantiated by the SCD Correlator.

The introduction and concept of the SCD Correlator has shown to have integrated three different beneficial aspects to the area of video analysis. The SCD Correlator is the entry point at which video first arrives into the system. The internal composition of the SCD Correlator are two buffers, one which buffers video frames, and the other to store the processed equivalent (the SCD-Objects) of the buffered frames. The first benefit that the SCD Correlator offers to the area of general video analysis is the automatic consolidation of objects. This is favourable and useful for applications which handle video because it significantly simplifies the access and tracking of objects in video.

The second benefit is that under the correct situations it can recover missing

objects during object correlation. Objects can be missed during the raw video analysis stages within the SCDs which can be caused by object occlusion or noise. The automatic recovery process injects predicted object locations by interpolation. An evaluation has shown the recovery mechanism to be highly accurate.

The final identified benefit and unique feature of the SCD Correlator is that it can not only recover missing objects, but also suppress the occurrence of false positives which improves the overall robustness. It cannot be guaranteed that the image processing on each frame of video is completely robust. In many image and video based applications the occurrence of false positives is not uncommon. False positives are characterised by appearing for a very short length of time, and these are easily filtered.

The development of the SCD Correlator has resulted in a new and efficient algorithm for filtering and linking objects across video frames. The general nature of the SCD Correlator for handling object identity and events may be applied to other aspects of video analysis which require object-level analysis. The results from the SCD Correlation process may be manipulated for higher-level semantics in the SCD Manipulator.

The SCD Manipulator receives objects and is able to query the objects for further semantics either through calling another set of SCDs or by applying a set of rules. The SCD Manipulator demonstrated the potential for deriving higher-level semantics from objects by composing a query to detect potted balls in snooker video. The current implementation of the SCD Manipulator only allows hardwired queries, though further work on querying composition capabilities will allow greater flexibility.

To demonstrate the application of the VCR System a snooker video analysis system was developed. The snooker video analysis system is able to detect and process top-view scenes of snooker video for prominent objects such as the table, balls, pockets, and events such as shot breaks. High-level semantics such as the detection of potted ball events were largely successful. It is anticipated that a

snooker scoring system is possible if the colours of the balls are recognised. An SCD for detecting the colour of objects was developed, but at present the balls are too small to be resolved for colour. The video was of recorded VHS quality, and the problem of detecting colour may be resolved by using a higher resolution recording. It is fair to say that the use of high quality video feeds is not uncommon in specialist applications.

In tackling the extraction of objects from video the problem of face detection was explored. Face detection can be regarded as one of the most heavily researched topics in object recognition. The work contributed towards rapid face(s) location at varying scales and orientations in images, the results of which were published in Chan *et al.* (10). The face detection work was packaged in an SCD to demonstrate recovery and object filtering aspects of the SCD Correlator.

The SCD-Objects exiting the SCD Correlator represent video events and object descriptions, and they may be stored and retrieved for later use. The benefits of this approach are realised when raw visual data can be processed into a content addressable format (the SCD-Objects) which offer multimedia applications the capability to access the underlying video content. The explicit nature of the data stored in SCD-Objects provides applications with the flexibility of processing content in whichever manner is suited for the application.

Finally, it has been shown that high-level information can be derived from objects thus satisfying the objectives to develop new techniques to analyse video for high-level content.

6.2 Future Work

The VCR System presented in this thesis successfully demonstrated innovation and novelty. The results that have been obtained are highly promising and have reasonably demonstrated a proof of concept. However, there is substantial scope for further research and development to the system.

One of the current limitations of the system is that queries have to be hard-

wired into the SCD Manipulator. Extending the interface of the VCR System to allow for query composition could be explored. This will allow cleaner and more flexible query composition. As a starting point, the use of the language Prolog is a recommended candidate.

The VCR System currently uses the visual portion of video. Audio is increasingly playing a significant role in video analysis. Extending the VCR System to have speech recognition and audio recognition capabilities could be explored. The combination of visual and audio analysis could conceivably produce more semantically relevant results.

Finally, the trend for exploring new ways to gain semantic-level access in video will no doubt continue as it is the inevitable challenge that must be faced as video information becomes increasingly significant and pervasive in our everyday lives.

References

[1] A. Müfit Ferman and A. Murat Teklap. Editing Cues for Content-Based Analysis and Summarization of Motion Pictures. In *Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 71–80. SPIE, 1997.

[2] J. A. Anderson. *An Introduction to Neural Computing.* MIT Press/Bradford Books, Cambridge, MA, 1995.

[3] J. Assfalg, C. Colombo, A. Del Bimbo, and P. Pala. Embodying Semiotic Cues in Video Retrieval. In *International Workshop on Multimedia Information Analysis and Retrieval*, pages 47–59. IAPR, 1998.

[4] Bruce G. Batchelor. Colour Recognition in Prolog. In *Machine Vision Applications, Architectures, and Systems Integration*, volume 1823, pages 294–305. SPIE, 1992.

[5] Bilge Günsel and A. Müfit Ferman and A. Murat Teklap. Temporal video segmentation using unsupervised clustering and semantic object tracking. *Journal of Electronic Imaging*, 7(3):592–604, 1998.

[6] Michael J. Black and Yaser Yacoob. Recognizing Facial Expressions In Image Sequences Using Local Parameterized Models Of Image Motion. *International Journal Of Computer Vision*, 25(5):23–48, 1997.

[7] Ivan Bratko. *Prolog - Programming for Artificial Intelligence.* Addison Wesley, 1991.

[8] R. Brunelli, O. Mich, and C. M. Modena. A Survey on the Automatic Indexing of Video Data. *Journal of Visual Communication and Image Representation*, 10(2):78–112, June 1999.

[9] M. Caliani, C. Colombo, A. Del Bimbo, and P. Pala. Commercial Video Retrieval by Induced Semantics. In *International Workshop on Content-Based Access of Image and Video Database*, pages 72–80. IEEE, 1998.

[10] Stephen C. Y. Chan and Paul H. Lewis. A Prefilter Enabling Fast Frontal Face Detection. In *Lecture Notes in Computer Science, on Visual Information and Information Systems, Visual 99*, volume 1614, pages 777–784, 1999.

[11] Shih-Fu Chang, William Chen, Horace J. Meng, Hari Sundaram, and Di Zhong. VideoQ: An Automated Content Based Video Search System Using Visual Cues. In *Multimedia*, pages 313–324. ACM, 1997.

[12] C. Chen and S. P. Chiang. Detection Of Human Faces In Colour Images. In *Vision Image Signal Processing*, volume 144 of *6*, pages 384–388. IEE, 1997.

[13] Gloria Chow and Xiaobo Li. Towards A System For Automatic Facial Feature Detection. *Pattern Recognition*, 26:1739–1755, 1993.

[14] Carlo Colombo, Alberto Del Bimbo, and Pietro Pala. Semantics in Visual Information Retrieval. *IEEE MultiMedia*, 6:38–53, 1999.

[15] J. M. Corridoni and A. Del Bimbo. Film Editing Reconstruction and Semantic Analysis. In *Lecture Notes in Computer Science, on Computer Analysis of Images and Patterns*, volume 970, pages 938–943, 1995.

[16] J. M. Corridoni and A. Del Bimbo. Film Semantic Analysis. In *Computer Architectures for Machine Perception*, pages 202–209. IEEE, 1995.

[17] Jonathan D. Courtney. Automatic Video Indexing Via Object Motion Analysis. *Pattern Recognition*, 30(4):607–625, 1997.

[18] Ying Dai and Yasuaki Nakano. Face-Texture Model Based on SGLD And Its Application In Face Detection In A Color Scene. *Pattern Recognition Society*, 29(6):1007–1017, 1996.

[19] Yining Deng, Debargha Mukherjee, and B. S. Manjunath. NeTra-V: Towards an Object-based Video Representation. In *Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 202–213. SPIE, 1998.

[20] Nevenka Dimitrova and Forouzan Golshani. Motion Recovery for Video Content Classification. In *ACM Transactions On Information Systems*, volume 13 of *4*, pages 408–439. ACM, October 1995.

[21] Stephan Fischer, Rainer Lienhart, and Wolfgang Effelsberg. Automatic Recognition Of Film Genres. In *MultiMedia*, pages 295–304. ACM, 1995.

[22] Stephan Fischer and Ralf Steinmetz. Choosing Efficient Feature Sets for Video Classification. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 196–207. SPIE, 1999.

[23] Ralph M. Ford. A Fuzzy Logic Approach To Digital Video Segmentation. In *Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 360–370. SPIE, 1997.

[24] Brian V. Funt and Graham D. Finlayson. Color Constant Color Indexing. In *Transactions On Pattern Analysis And Machine Intelligence*, volume 17 of *5*, pages 522–529. IEEE, May 1995.

[25] Christophe Garcia and Georgios Tziritas. Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis. In *Transac-*

tions On Multimedia, volume 1 of *3*, pages 264–277. IEEE, September 1999.

[26] Yihong Gong, Chua Hock Chuan, Zhu YongWei, and Masao Sakauchi. A Generic Video Parsing System With a Scene Description Language (SDL). *Real-Time Imaging*, pages 45–59, 1996.

[27] Arun Hampapur. Semantic Video Indexing: Approach and Issue. *SIGMOD Record*, 28(1):32–39, 1999.

[28] Alan Hanjalic, Reginald L. Lagendijk, and Jan Biemond. Semi-Automatic News Analysis, Indexing and Classification System based on Topics Preselection. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 86–97. SPIE, 1999.

[29] N. Herodotou, K. N. Plataniotis, and A. N. Venetsanopoulos. Automatic Location and Tracking of the Facial Region in Color Video Sequences. *Signal Processing: Image Communication*, 14:359–388, 1999.

[30] Qian Huang, Zhu Liu, and Aaron Rosenberg. Automated Semantic Structure Reconstruction and Representation Generation for Broadcast News. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 50–62. SPIE, 1999.

[31] Daniel Huttenlocher, Greg Klanderman, and William Rucklidge. Comparing Images Using The Hausdorff Distance. In *Transactions On Pattern Analysis And Machine Intelligence*, volume 15 of *9*, pages 850–863. IEEE, September 1993.

[32] Yuri Ivanov, Chris Stauffer, Aaron Bobick, and W. E. L. Grimson. Video Surveillance of Interactions. In *The Second IEEE Workshop on Visual Surveillance*, pages 82–89. IEEE, 1999.

[33] Alejandro Jaimes and Shih-Fu Chang. Model-Based Classification of visual Information for Content-Based Retrieval. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 402–414. SPIE, 1999.

[34] Tony Jebara, Cyrus Eyster, Josh Weaver, Thad Starner, and Alex Pentland. Stochasticks: Augmenting the Billiards Experience with Probabilistic Vision and Wearable Computers. In *The First International Symposium on Wearable Computers (ISWC '97)*, 1997.

[35] Andrew C. Jones and Bruce G. Batchelor. PIP - an integrated environment for developing Prolog-based image processing applications. In *4th International Conference on Practical Applications of Prolog*, pages 145–158, 1996.

[36] Paul Juell and Ron Marsh. A Hierarchical Neural Network For Human Face Detection. *Pattern Recognition*, 29:781–787, 1996.

[37] Weixin Kong, Xianfeng Ding, Hanqing Lu, and Songde Ma. Improvement Of Shot Detection Using Illumination Invariant Metric And Dynamic Threshold Selection. In *Lecture Notes in Computer Science, on Visual Information and Information Systems, Visual 99*, volume 1614, pages 277–282, 1999.

[38] Young H. Kwon and Niels da Vitoria Lobo. Age Classification From Facial Images. *Computer Vision and Image Understanding*, 74(1):1–21, April 1999.

[39] Choong Hwan Lee, Jun Sung Kim, and Hyu Ho Park. Automatic Human Face Location In A Complex Background Using Motion And Color Information. *Pattern Recognition*, 29(11):1877–1889, 1996.

[40] Suh-Yin Lee and Huan-Ming Kao. Video Indexing - An Approach base on Moving Object and Track. In *Storage and Retrieval for Image and Video Databases*, volume 1908, pages 25–36. SPIE, 1993.

[41] Rainer Lienhart. Comparison of Automatic Shot Boundary Detection Algorithms. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 290–301. SPIE, 1999.

[42] Rainer Lienhart, Christoph Kuhmünch, and Wolfgang Effelsberg. On the Detection and Recognition of Television Commercials. In *International Conference On Multimedia Computing and Systems*, pages 509–516. IEEE, 1997.

[43] Jiebo Luo, Chang Wen Chen, and Kevin J. Parker. Face Location In Wavelet-Based Video Compression For High Perceptual Quality Videoconferencing. *IEEE Transactions On Circuits And Systems For Video Technology*, 6(4):411–414, 1996.

[44] T. McGee and N. Dimitrova. Parsing TV programs for identification and removal of non-story segments. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 243–251. SPIE, 1999.

[45] Ali Reza Mirhosseini and Hong Yan. Human Face Image Recognition: An Evidence Aggregation Approach. *Computer Vision and Image Understanding*, 71(2):213–230, August 1998.

[46] M.Nisida and Y. Ariki. Speaker Indexing for News Articles, Debates and Drama in Broadcasted TV Programs. In *Multimedia Systems*, pages 466–471. IEEE, 1999.

[47] David E. Benn Mark S. Nixon and John N. Carter. Robust Eye Centre Extraction Using the Hough Transform. In *1st International Conference on Audio-and Video-Based Biometric Person Authentication, Lecture Notes in Computer Science*, pages 3–9, 1997.

[48] Greg Pass, Ramin Zabih, and Justin Miller. Comparing Images Using Color Coherence Vectors. In *MultiMedia*, pages 65–73. ACM, 1996.

[49] Nilesh V. Patel and Ishwar K. Sethi. Video Shot Detection And Characterisation For Video Databases. *Pattern Recognition*, 30(4):583–592, 1997.

[50] Gopal Sarma Pingali, Yves Jean, and Ingrid Carlbom. Real Time Tracking for Enchanced Tennis Broadcasts. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 260–265. IEEE, 1998.

[51] Gopal Sarma Pingali, Yves Jean, and Ingrid Carlbom. LucentVision: A System for Enhanced Sports Viewing. In *Lecture Notes in Computer Science, on Visual Information and Information Systems, Visual 99*, volume 1614, pages 689–696, 1999.

[52] Steven Pinker. *How the Mind Works.* Penguin Books, 1997.

[53] Daniel Reisfeld and Yehezkel Yeshurun. Preprocessing of Face Images: Detection of Features and Pose Normalization. *Computer Vision and Image Understanding*, 71(3):413–430, September 1998.

[54] Rick Rickman and John Stonham. An Intelligent Video Editing System Using A Neural Network Coding Scheme. In *Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 137–143. SPIE, 1994.

[55] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Human Face Detection In Visual Scenes. Technical report, CMU-CS-95-158R, Carnegie Mellon University, http://www.cs.cmu.edu/ har/faces.html, November 1995.

[56] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural Network-Based Face Detection. In *Transactions On Pattern Analysis And Machine Intelligence*, volume 20 of *1*, pages 23–38. IEEE, January 1998.

[57] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Exploring Video Structure Beyond The Shots. In *International Conference on Multimedia Computing and Systems*, pages 237–240. IEEE, 1998.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back Propagating Errors. *Nature*, pages 533–536, 1986.

[59] Ashok Samal and Prasana A. Iyengar. Human Face Detection Using Silhouettes. *International Journal Of Pattern Recognition and Artificial Intelligence*, 9(6):845–867, 1995.

[60] Shin'ichi Satoh. Towards Actor/Actress Identification in Drama Videos. In *Multimedia*, pages 75–78. ACM, 1999.

[61] Shin'ichi Satoh, Yuichi Nakamura, and Takeo Kanade. Name-It: Naming and Detecting Faces in News Videos. *IEEE MultiMedia*, 6:22–35, 1999.

[62] Drew D. Saur, Yap-Peng Tan, Sanjeev R. Kulkarni, and Peter J. Ramadge. Automated Analysis And Annotation Of Basketball Video. In *Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 176–187. SPIE, 1997.

[63] I. K. Sethi, I. Coman, B. Day, F. Jiang, D. Li, J. Segovia-Juarez, G. Wei, and B. You. Color-WISE: A System For Image Similarity Retrieval Using Color. In *Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 140–149. SPIE, 1997.

[64] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-Based Image Retrieval at the End of the Early Years. In *Transactions On Pattern Analysis And Machine Intelligence*, volume 22 of *12*, pages 1349–1380. IEEE, December 2000.

[65] Alvy Ray Smith. Color Gamut Transform Pairs. In *SIGGRAPH 78*, pages 12–19, 1978.

[66] M. S.Toller, P. H. Lewis, and M. S. Nixon. Video Segmentation Using Combined Cues. In *Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 414–425. SPIE, 1997.

[67] G. Sudhir, John C. M. Lee, and Anil K. Jain. Automatic Classification of Tennis Video for High-level Content-based Retrieval. In *International Workshop on Content-Based Access of Image and Video Databases (CAIVD '98)*, pages 81–90. IEEE, 1998.

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

[69] Candemir Toklu and Shih-Ping Liou. Semi-Automatic Dynamic Video Object Marker Creation. In *Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 178–185. SPIE, 1999.

[70] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[71] Alan Watt. *Fundamentals of Three-Dimensional Computer Graphics*. Addison Wesley, 1989.

[72] Haiyuan Wu, Qian Chen, and Masahiko Yachida. Face Detection From Color Images Using a Fuzzy Pattern Matching Method. In *Transactions On Pattern Analysis And Machine Intelligence*, volume 21 of *6*, pages 557–563. IEEE, June 1999.

[73] Wei Xiong and John Chung-Mong Lee. Efficient Scene Change Detection and Camera Motion Annotation for Video Classification. *Computer Vision and Image Understanding*, 71(2):166–181, August 1998.

[74] Wei Xiong, John Chung-Mong Lee, and Man-Ching Ip. Net comparison: A fast and effective method for classifying image sequences. In *Storage*

and Retrieval for Image and Video Databases III, volume 2420, pages 318–328. SPIE, 1995.

[75] Li-Qun Xu, Dave Machin, and Phil Sheppard. A Novel Approach to Real-time Non-intrusive Gaze Finding. In *Proceedings of the 9th British Machine Vision Conference, Southampton*, volume 2, pages 428–437, 1998.

[76] Ramin Zabih, Justin Miller, and Kevin Mai. A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. In *MultiMedia*, pages 189–200. ACM, 1995.

[77] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic Partitioning Of Full-Motion Video. In *Multimedia Systems*, pages 10–28. ACM, 1993.

[78] HongJiang Zhang, Yihong Gong, Stephen W. Smoliar, and Shuang Yeo Tan. Automatic Parsing Of News Video. In *Multimedia Computing Systems*, pages 45–54. IEEE, 1994.