Content-based Video Database Retrieval using Shape Features

Farahnaz Mohanna

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Centre for Vision, Speech and Signal Processing School of Electronics and Physical Sciences University of Surrey Guildford, Surrey GU2 7XH, U.K.

October 2002

© Farahnaz Mohanna 2002
In Memory of My Father

To My Mother
Summary

In a typical content-based video retrieval system, the user submits a query and will then expect the system to locate all similar data in the corresponding video database. Content-based means representing the image content by means of low level visual features that are extracted through image processing techniques. The low level features include colour distributions, textures, shape and motion, in the case of video.

In this thesis, a novel content-based video retrieval system using shape features is proposed. To provide indices, first all the shots in a video sequence are extracted. Then corners of all frames in each video shot are detected applying proposed curvature scale space corner detector using different scale of smoothing. As a user interface, a proposed fast active contour model is used to specify one object of interest as a query in one of the frames of each video shot. Afterwards, closest corners of query object to the final snake in this frame are extracted and tracked forward and backward through the whole of that shot using proposed multiple-match tracker. This tracker, which does not make any important assumptions or use any motion models, can retrieve the query in any video sequence even when there is non-smooth and unconstrained motion. By tracking the corners of query object forward and backward, the positions of similar objects in each video frame are determined. Two methods are considered for demonstrating the query and its similar objects to the user. Experiments have been carried out on a wide range of real video databases. All the results confirm that the proposed method which exploits frame corners is more efficient and more generally applicable.

Key words: Content-based retrieval, Curvature Scale Space, Corner detection, Energy-minimising active contour model, Corner matching, Feature tracking, Multiple hypotheses
Acknowledgements

I would like to express my deepest appreciation and gratitude to my supervisor, Dr. Farzin Mokhtarian for his invaluable guidance and encouragement.
I would like to thank all the members of the Centre for Vision Speech and Signal Processing, for providing a friendly environment by giving advice, help, encouragement during my studies and especially for their support after my father's death that helps me to continue my studies. This is because I believe deeply that my parents' love, constant support and specially their belief in me helped me to be a better person.
Special appreciation to my mother, my husband, Reza, and to my dearest children, Saeid and Sahar for their love, patience, and support during this period.
Finally, thanks to God for blessing all that I have and I can not be grateful enough.
# Contents

## 1 Introduction

1.1 Content-based Video Retrieval System ........................................................ 2
1.2 Proposed Content-based Video Retrieval System ..................................... 4
1.3 Contributions ................................................................................................... 6
1.4 Outline ............................................................................................................ 8

## 2 Literature Review

2.1 Cut Detection Methods ................................................................................ 13
2.2 Corner Detectors ............................................................................................. 15
2.3 Active Contours ............................................................................................. 17
2.4 Feature Tracking Algorithms ....................................................................... 19
   2.4.1 Tommasini Feature Tracker .............................................................. 22
2.5 Content-based Video Database Retrieval .................................................... 25

## 3 Multi-scale Corner Detector

3.1 Introduction ..................................................................................................... 29
3.2 The Curvature Scale Space Technique ....................................................... 30
3.3 Overview of Original CSS Corner Detector ................................................. 31
3.4 The Original CSS Shortcomings .................................................................... 32
3.5 New Method of CSS Corner Detection ....................................................... 33
   3.5.1 Using Different Scales of CSS ........................................................... 33
   3.5.2 Smoothing the Absolute Curvature Function of Long Contours . 35
   3.5.3 Tracking ............................................................................................... 36
   3.5.4 Unifying Close Corners .................................................................... 37
3.6 Multi-scale Edge Detector ............................................................................. 37
4 Accurate and Fast Active Contour Models

4.1 Introduction ................................................. 41
4.2 Improved Curvature Estimation for Active Contours .......... 44
4.3 Fast Active Contour Convergence through CSS Filter ......... 47
4.4 An Adaptive Smoothed Active Contour Model ................. 50

5 Robust Corner Tracking for Multimedia Applications ......... 53

5.1 Introduction .................................................. 53
5.2 Tommasini Tracker Shortcomings ............................... 54
5.3 Multiple-match Corner Tracker ................................ 55
5.4 Two-frame Corner Matching .................................. 56
5.4.1 Single Matching Process ................................. 56
5.4.2 Multiple Matching Process ............................... 57
5.5 Corner Monitoring Process ................................... 60
5.6 Corner Tracking including New Corners ....................... 63
5.6.1 Multiple Matching including New Corners ............... 64

6 Content-based Video Retrieval through Robust Corner Tracking 69

6.1 Introduction .................................................. 69
6.2 User Query Specification .................................... 70
6.3 Selecting Corners in Query Closest to the User Snake ......... 71
6.4 Tracking Selected Corners in Query ........................... 72
6.5 Demonstrating Identified Similar Objects as Query .......... 72
6.5.1 Bounding Box Method ................................... 73
6.5.2 Least-Squares Estimation Method ......................... 73

7 Results, Discussion, and Performance Evaluation of Proposed System 79

7.1 Introduction .................................................. 79
7.2 Corner Detector Results, Discussion, and Performance Evaluation ... 80
7.2.1 The ECSS Corner Detector Results and Discussion ......... 80
7.2.2 Performance Evaluation of the ECSS Corner Detector ....... 82
7.2.3 The Multi-scale Corner Detector Results and Discussion ... 90
7.3 The Active Contours Results, Discussion, and Performance Evaluation . 92
7.3.1 The ICEAC Active Contour Model Results, Discussion, and Performance Evaluation ................................................. 92
7.3.2 The SAC Active Contour Model Results, Discussion, and Performance Evaluation ................................................. 93
7.3.3 The ASAC Active Contour Model Results, Discussion, and Performance Evaluation ................................................. 96

7.4 Corner Tracking Results, Discussion, and Performance Evaluation ................................................................. 108
7.4.1 The Multiple-match Corner Tracker Results and Discussion ............................................................................... 108
7.4.2 The Results of Corner Tracker including New Corners ..................................................................................... 109
7.4.3 Performance Evaluation of the Multiple-match Corner Tracker ............................................................................ 117

7.5 Proposed Content-based Retrieval System Results and Discussion ........................................................................... 122

8 Conclusions ........................................................................................................................................... 141
8.1 Main Advantages of the Proposed System ........................................................................................................... 145

A Similarity Functions Metrics ................................................................................................................................ 147
Chapter 1

Introduction

With technological advances in multimedia, digital TV and information highways, a large amount of video data is now publicly available. However, without appropriate search technique all this data is nearly not usable. Therefore the increasing amount of digital image and video data has stimulated new technologies for efficient searching, indexing, content-based retrieving and managing multimedia databases. Before starting to focus deeply on the target of this thesis, a number of technical terms are explained below:

*Video database retrieval system* is a software system capable of managing a video data collection and providing content-based access to user.

*Multimedia* is a description of diverse technologies that combines media of communication such as text, graphics, sound, images, etc.

*Database indexing* consists of analysing data and extracting its visual features and storing them in a meta-database.

*Video parsing* is the process of decomposing video streams into scenes, shots, and frames. A *frame* is the basic unit of the video and corresponds to a single image. *Shots* are unbroken sequences of frames from one camera or one “on-off” of the camera. Finally, *scenes* are a collection of adjoining shots that focus on some objects of interest.

*Cut detection* is the process of partitioning the video sequences detecting shot changes. This is because frames belonging to two successive shots are likely to be visually...
dissimilar, then the partitioning of video sequences are made by the detection of shot changes. Shot transitions are divided into two categories; abrupt and gradual. There is only one type of abrupt shot transition, namely a cut.

In this part, we turn our attention to definition of a content-based video retrieval system. In a typical content-based video retrieval system, user submits a query and will then expect the system to locate all instances of similar objects as query in each frame of the video database. Content-based means representing the image content by means of low level visual features that are extracted through image processing techniques. The low level features include colour distributions, textures, shape (and motion, in the case of video):

- **Colour**: Using colour for video database retrieval means making use of the global colour distribution in images, i.e. the histogram.
- **Texture**: Texture is defined by the repetition of a basic pattern over a given area. Then as it is a neighbourhood property, a single pixel on an image can not have a texture.
- **Shape**: Shapes are often more complex than colour and texture. While colour and texture can be quantified by a few parameters, complex shapes need hundreds of parameters to be represented explicitly. Shape descriptors obtain a more semantic representation of an image. For example between two different objects with same colour or two different objects with same texture only shape descriptors are good discriminators.

One of the most intuitive types of point features on shapes are corners. Corners are points in an image which exhibit two dimensional structure. The fact that they are points such that their nearby regions have considerable image structure makes them easier to match than other features. They can also provide motion information in any direction on an image.

### 1.1 Content-based Video Retrieval System

Content-based retrieval of video data, based on the indexed and annotated information content, can be categorised as follows:
1.1. Content-based Video Retrieval System

- Semantic-based approach
- Audiovisual feature-based approach
- Hybrid approach

Focusing mainly on annotating the semantic content in video data, the semantic-based approach ([36],[38],[66]) is suitable for exploring the content description with meaning and intuition to users. Several annotation structures have been developed to represent semantic information, including keywords, natural languages, and iconic languages. However, semantic-based methods do not lead to an automatic annotation process and to characterise rich visual contents. Moreover, semantic annotation is generally ambiguous and application-dependent. The audiovisual feature-based approach ([22], [68],[35]) derived from image information systems extracts low-level features from video data. While completely automating the indexing processes, this method lacks semantics associated with the extracted features. Despite the visual effectiveness of this method, users have difficulty in querying audio-visual features, such as colour, texture, shape and object motion. A highly promising approach attempts to integrate semantic-based and feature-based contents. This integration is referred to as the hybrid approach ([44],[15]).

The goal of our project is to investigate retrieval from video databases using shape content. Since the most intuitive types of point features on shapes are corners which provide motion information, we have selected them as low-level visual features for retrieving moving objects from video databases. Therefore our retrieval system is the audiovisual feature-based approach and semi-automatic. However the role of a user in our retrieval system is to specify an object of interest as query using a computer mouse. Furthermore, user does not need to select threshold values or to choose parameters and pass them to the proposed system. Then in our system, the problem of querying audio-visual features, which is difficult for users in feature-based retrieval systems, has been solved. A significant amount of work has been carried out on feature tracking. An overview of some of these methods can be seen in Sec.2.4. Among the existing techniques, those that exploit image corners are believed to be more robust and more generally applicable since:
• Prior segmentation of the object is not necessary.

• Corners can be extracted from any free-form object.

• Non-rigid motion of objects can be accommodated through an affine model.

• Corners can be localised very well so tracking will be able to cope with occlusion of objects and clutter.

• Due to small motion of the object of interest in two successive frames of video sequences, tracking can consider small neighbourhoods of object corners.

1.2 Proposed Content-based Video Retrieval System

In this thesis, a novel content-based video retrieval system based on extracting corners from frames in input sequence and tracking them through the video database is introduced. The main stages of the proposed system are as following:

1. Providing indices through pre-processing stage. In this stage, first by employing an existing technique for cut detection, all of the shots in a video sequence are extracted. Then the corners of all frames in each shot in the video sequence are detected and stored in a meta-database.

2. In the second stage, user can select a frame among frames in input sequence as a selected-frame. Afterwards selected-frame will be presented to the user and system will wait until he/she specifies one object of interest in that frame as a query. The user draws an initial snake as close as possible to this object in the selected-frame using a computer mouse. The idea is to use an active contour which locks on the object of interest accurately and quickly. The user snake which locks on the object of interest as close as possible, due to the constraint forces, is referred to as the final snake.

3. In the third stage, closest corners of the query object (or closest corners of the underlying image) to the final snake will be selected and tracked forwardly or/and backwardly through the whole of that shot in order to determine the positions of
1.2. Proposed Content-based Video Retrieval System

similar objects as query in each video frame. Tracking forwardly or/and backwardly is based on which frame number has the selected-frame in input video sequence. For example if a selected-frame is the first frame of input sequence, only forward tracking is needed. If last frame is chosen as a selected-frame, then backward tracking can find the positions of similar objects as query in each video frame. Selecting one of the frames among input sequence, except the first and the last frames, as a selected-frame requires forward and backward tracking together in order to determine the positions of similar objects as query in each video frame in input sequence.

4. Demonstrating the query and its similar objects to the user using two methods:

- drawing a rectangle around identified similar objects in each frame.
- drawing similar snakes as user snake around identified similar objects in each frame.

For such a retrieval system, the required image processing techniques are a robust corner detector, an accurate and high-speed active contour model, reliable corner matching and tracking algorithms, and finally an approach for demonstrating retrieval results. A corner detector can be successfully used for this system if it satisfies these criteria:

- All the true corners should be detected.
- No false corners should be detected.
- Corner points should be well localised.
- Corner detector should be robust with respect to noise.

A good active contour model means the initial snake locks quickly on to the object of interest as a membrane or thin plate. Real-time corner tracking needs robust corners which can be identified and tracked efficiently from frame to frame in a sequence. Then a reliable algorithm for corner matching among frames is required to find correct correspondences. Actually corner matching is commonly referred to as the correspondence problem. This problem is how to automatically match corresponding corners from two
images, while no incorrect matches are assigned at the same time. Finally all the methods that we have implemented for each of these algorithms as the stages of proposed content-based retrieval system will be explained in detail one by one through sections of this thesis.

A number of application areas in which content-based retrieval is becoming a principal activity are, among others, military surveillance problems, processing of video data from security cameras, TV production, video-on-demand, art galleries and museum management, architectural and engineering design, geographic information systems, medical indexing, home entertainment, teleshopping and so on. In all of these applications, we face a huge amount of data which content-based retrieval helps to prepare an appropriate search technique for. Obviously the kind of data is different in these applications. For instance, in video-on-demand, museum management and home entertainment; movies, art production, and games are data respectively. However in medical indexing body scans are data which should be retrieved by doctors for medical diagnosis and treatment.

1.3 Contributions

In comparison to related work in this field, our system has a number of distinctive properties. Some of the main contributions are listed below:

- A novel corner detection algorithm; multi-scale corner detector, which is the enhanced CSS\(^1\) (ECSS) corner detector using different scales of smoothing. The ECSS corner detector is also an improvement of the CSS corner detector with better performance on blunt and rounded corners.

- A novel active contour model; SAC, which includes a new procedure for the smoothing term of the internal energy of active contours in energy minimising active contour models. Application of this new procedure resulted in locking on the interest object quickly and accurately.

\(^1\)CSS stands for “Curvature Scale Space” and the CSS-based shape descriptor has been selected for MPEG-7 standardisation.
1.3. Contributions

- A novel two-frame matching approach; multiple-hypotheses, based on multiple matching and distance criterion which ensures as much as possible the closest and the minimum correspondences that we need to have to continue tracking.

- A novel corner tracker; multiple-match tracker, based on three-frame monitoring process without making any important assumptions or using any motion models. The proposed tracker includes a new approach to three-frame monitoring which helps to make more robust the number and the positions of tracked corners among frames due to sudden change in drift of the tracker. Note that our novel tracker tracks corners during non-smooth motion which is a very difficult task for tracking.

- Elimination of parameters setting and threshold values selecting. It means that the user which has no idea about audio-visual features and he/she often has difficulty to select values for such these features, in proposed system the user does not need to do any extra jobs (such as selecting any threshold values or setting any parameters) except drawing the initial snake around an object of interest.

- A convenient selection of a query by drawing an initial snake as close as possible to the object of interest using a computer mouse. Therefore our user interface not only is not difficult but also it is more attractive to use.

- A snake-based model representation of the retrieved results which draws similar snakes as the user snake around identified similar objects. Therefore user can have very close communication with the system. She/he can learn from these output snakes how to draw her/his initial snake around object of interest that can have better output results. For example drawing a very small snake in a selected-frame where includes no corner, does not return any retrieved results.

- Access to individual video objects in the video stream. The classical approach to content-based video access, which consists of shot boundary detection, followed by selection of key frames that characterise the visual content of each shot, does not provide access to individual video objects. However in our system user snake
can be drawn anywhere in a selected-frame, no matter what is inside that snake from underlying image and what will be the retrieved results.

- Accepting any kind of video data as input sequence including any motion model especially non-smooth motion, any transformations, or any lighting and then giving promising results very quickly and efficiently. These results consist of not only, identified similar objects as query but also a complete record of tracked corners in each frame, user initial and final snakes, intermediate iterations results, closest corners of underlying image to the final snake in selected-frame, and the whole of the selected-frame corners. Therefore by looking at these results, user can visually follow how a query has been retrieved through the stages of the proposed retrieval system.

1.4 Outline

This section discusses the organisation of the remainder of this thesis. Chapter 2 presents a critical overview of a number of published approaches in all key aspects of our projects including cut detection, corner detection, active contours, feature tracking and content-based video retrieval. In the part of feature tracking survey, since our proposed method for corner tracking will be compared to the Tommasini et al.'s feature tracker\(^2\), we discuss their method more specific in 2.4.1

Since for proposed retrieval system the required image processing techniques are a robust corner detector, an accurate and high-speed active contour model, reliable corner matching and tracking algorithms, the theories underlying proposed techniques for these image processing tools are presented in chapters 3, 4, and 5 respectively. In chapter 3, first the multi-scale corner detector which is the enhanced curvature scale (ECSS) corner detector using different scales of smoothing is introduced. Therefore since the ECSS corner detector is also an improvement of the CSS corner detector [62], next after explaining the CSS method generally in Sec.3.2, original CSS corner detector and its shortcomings are described in Sec.3.3 and Sec.3.4 respectively. Afterwards, The ECSS corner detector outline is described in Sec.3.5. In continue, more details about the

\(^2\)http://mvl.dimi.uniud.it/Respro/Tracker
multi-scale corner detector and the multi-scale edge detector are explained in section 3.6.

Our quick and accurate active contour model; SAC, to be used as a user interface is introduced in chapter 4. In this chapter, first the theory underlying the proposed active contour model is explained in Sec.4.1. Then the estimation of curvature for accurate localisation of active contours is presented in section 4.2. Our method for quick active contours is proposed in section 4.3 followed by introducing an adaptive smoothed active contour model in section 4.4.

In chapter 5, the shortcomings of Tommasini tracker have been discussed in Sec.5.2. Afterwards, the multiple-match corner tracker is explained in section 5.3. Furthermore in this chapter, traditional two-frame matching using single matching (5.4.1) is compared to our proposed two-frame matching using multiple matching (5.4.2) combined with three-frame monitoring (5.5). In section 5.6, in the two-frame matching stage of the multiple-match tracker, not only the corners of the first frame but also all the corners that appear fresh in each frame of input sequence are considered. This process which has been explained in 5.6.1, gives this possibility to the user to specify an object of interest as query in any frames in the input sequence.

Our content-based video retrieval system based on the proposed image processing tools is introduced in chapter 6. In this chapter, by selecting a video sequence including affine transformation, we will see the performance of our proposed retrieval system. Therefore by demonstrating the retrieved results, not only the advantages and the attractive properties of our system but also its limitations can be determined explicitly. After explaining all the theories underlying the proposed retrieval system, the results, discussion, and performance evaluation of our proposed system through its stages are demonstrated in chapter 7. This chapter is divided to four main sections; the corner detector results, discussion, and performance evaluation (Sec.7.2), the active contour results, discussion, and performance evaluation (Sec.7.3), the corner tracking results, discussion, and performance evaluation (Sec.7.4) and the content-based video retrieval system results, discussion, and performance evaluation (Sec.7.5). In 7.2.1, the ECSS corner detector results are compared to the results of our four test corner detectors (Plessey [30], Kitchen and Rosenfeld [40], SUSAN [80], Mokhtarian and Suomela [62])
which have been suggested through the survey in Sec.2.2. Performance evaluation of
the ECSS corner detector in comparison to these four test corner detectors under simi-
larliy and affine transforms has been shown in section 7.2.2. The multi-scale corner
detector results have been shown in some randomly selected frames in a number of
video sequences in 7.2.3. Therefore through these results you will find out the multi-
scale corner detector has sufficient robust corners for tracking among frames in input
sequence, even when transformations occur. As result this corner detector was selected
as a corner detection tool for proposed retrieval system.

In section 7.3, the ICEAC, the SAC and the ASAC active contour results in images
with single and multiple objects have been illustrated in 7.3.1, 7.3.2, and 7.3.3 respec-
tively. From these results, the SAC active contour model which is an accurate and
quick active contour model is selected as the best user interface for the proposed re-
trieval system. The ICEAC, the SAC active contours are also compared to the AMI
active contour model individually and together in term of speed. Therefore through
this comparison you will find out why the SAC active contour model is the best user
interface for proposed retrieval system. The multiple-match corner tracker results have
been illustrated in 7.4.1. Furthermore in this section, traditional two-frame matching
using single matching is compared to our proposed two-frame matching using multiple
matching combined with three-frame based monitoring. Results of applying modified
multiple-match tracker to the bream sequence which is an example of a sequence includ-
ing occlusion have been shown in 7.4.2. These results also have been compared to the
results of applying the multiple-match corner tracker to the bream sequence. Perfor-
mance evaluation of the multiple-match corner tracker in comparison to the Tommasini
tracker can be found in 7.4.3. We have tested the proposed retrieval system in a wide
range of real video databases depicting translation, rotation, scaling, affine transfor-
mation, non-smooth motion and combination of them with different lighting and different
camera motions. Some randomly selected of these results have been illustrated in sec-
tion 7.5. All the results confirm that the proposed retrieval system is efficient and
practical. The retrieved results are promising for content-based retrieval using only
the corners on moving objects. Obviously adding an object recognition method or
integrating high-level concepts such as objects and events which are semantic content
1.4. Outline

will change the results to more highly promising. However in our system without any important assumptions, any motion model, any information of camera calibration, and just by tracking corners which are the minimum information that we have on moving object, the retrieved results are quite good. Therefore the proposed system is very efficient and also practical especially based on our user interface which is easy to apply. Finally, a summary of the thesis and possible extensions for future research can be found in chapter 8.
Chapter 1. Introduction
Chapter 2

Literature Review

The aim of this chapter is to present a brief literature overview relevant to key aspects of our project. These key aspects are cut detectors, corner detectors, active contours, feature trackers, and content-based video database retrieval.

2.1 Cut Detection Methods

Video applications will require the capability for users to search efficiently through large databases. For this search to be viable a video sequence must be partitioned into meaningful segments. A video sequence can be decomposed into scenes, shots and frames. Frames, shots, and scenes have been defined in Chap.1. Since frames belonging to two successive shots are likely to be visually dissimilar, then the partitioning of video sequences are made by the detection of shot changes. Shot transitions are divided into two categories; abrupt and gradual. There is only one type of abrupt shot transition, namely a cut.

The cut detector takes video as its input and outputs the locations of the cuts. Cut detection is subjective, meaning that what one person perceives as a cut, another may not. In practice the majority of cut detectors calculate a parameter between adjacent frames. This parameter is thresholded to detect cuts. A brief review of some cut detection methods is presented here. In Pixel Level Change Detection or Pixel Differences ([1],[8],[32],[100]), a change between a pair of images is detected by comparing
corresponding pixels in the two frames. This method counts the number of pixels that change in value more than some threshold. Then this total or percentage of this total is compared to a second threshold to determine if a shot change has occurred. This method requires the setting of two thresholds and is sensitive to motion. Sensitivity to motion may be reduced by using block based techniques. Block-based techniques ([1], [8],[32],[100]) are closely associated with the likelihood ratio. Each frame is partitioned into blocks. Rather than comparing individual pixels to generate the difference picture the average of each block is compared. Weighting factors may be applied to each block allowing greater consideration of given regions. Compared to pixel level change detection, block-based techniques have improved robustness to motion. Likelihood ratio ([1],[8],[32],[100]) is an example of a statistical method. Each frame is broken up into blocks and statistical measures of temporally adjacent blocks are compared. When a given number of blocks have a likelihood ratio greater than a threshold, a shot change is detected. The problem is that two corresponding blocks can have the same likelihood ratio even though they are different. Histogram Comparison ([1],[8],[32],[100]) is the most common method for detecting shot changes. One histogram per frame is computed from the grey level values. Then histograms are compared by summing the absolute difference of the corresponding bins. If this value is greater than a threshold a shot change is detected. Another method is $\chi^2$-test comparison [100] of histograms. This method enhances the difference between shot changes, but also enhances small changes due to camera or object motion. Histograms are invariant to image rotation, scale and occlusion. Histograms may be extended to compare colour information as well as grey level values. However, there are cases in which two frames have similar histograms but completely different content. Average intensity measurement [28], instead of considering individual pixels or blocks of pixels, considers the whole of the frame. If the absolute difference between the average of intensity between two successive frames is greater than a threshold a shot change is detected. Some methods such as Twin Comparison ([1],[32],[100]) are techniques for the detection of gradual shot transitions. In our retrieval system we have used an existing technique for cut detection developed at CVSSP\(^1\) which has been presented in [98]. This method is a combination of multiple

---

\(^1\)Centre for Vision, Speech and Signal Processing, in University of Surrey
experts to achieve better results in detecting shot cut boundaries, by exploiting the fact that different experts detect different video features.

2.2 Corner Detectors

Considerable research has been carried out on corner detection in recent years. Kitchen and Rosenfeld [40] computed a cornerness measure based on the change of gradient direction along an edge contour multiplied by the local gradient magnitude as follows:

$$C_{KR}(x, y) = \frac{I_{xx}I_y^2 - 2I_xI_yI_{xy} + I_{yy}I_x^2}{I_x^2 + I_y^2}$$

(2.1)

The local maximum of this measure isolated corners using non-maximum suppression applied on gradient magnitude before its multiplication with the curvature. This detector is sensitive to noise with poor localisation and unstable. Plessey [30] cornerness measure is:

$$C_P(x, y) = \frac{(I_x^2) + (I_y^2)}{(I_x^2)(I_y^2) - (I_xI_y)^2}$$

(2.2)

Where he found $I_x$ and $I_y$ using the $(n \times n)$ first-difference approximations to the partial derivatives and calculated $I_x^2$, $I_y^2$, and $I_xI_y$. Then used a Gaussian smoothing, computed the sampled means $\langle I_x^2 \rangle$, $\langle I_y^2 \rangle$ and $\langle I_xI_y \rangle$ using the $(n \times n)$ neighbouring point samples. Sampled mean here is a weighted average of neighbouring values. This algorithm in the case of large Gaussian convolution does not have a good localisation. Even more the application of constant-variable false corner response suppression causes it to be unstable. Smith and Brady [80] used a circular mask for corner detection and no derivatives were used. They introduced the Susan algorithm as follows:

Consider an arbitrary pixel in the image and corresponding circular mask around it (the centre pixel shall be called the ‘nucleus’). Provided image is a compact region within the mask whose pixels have similar brightness to the nucleus and this area whose be called USAN, an acronym standing for “Univalue Segment Assimilating Nucleus”. The USAN area is at a maximum when the nucleus lies in a flat region of the image surface, it falls to half of this maximum very near a straight edge, and falls even further when inside a corner. To find corners they computed the area and the centre of gravity of the USAN,
and developed a corner detector based on these parameters. Mokhtarian and Suomela [62] detected image corners based on the curvature scale space (CSS) representation. Initial corners were defined as points where image edges have their maxima of absolute curvature. The corner points were extracted at a high scale of the CSS. Finally in this algorithm, initial corners were tracked through multiple lower scales. Moravec [63] observed that the difference between the adjacent pixels of an edge or a uniform part of the image is small but at the corner, the difference is significantly high in all directions. The idea was later used by Harris [30] to develop Plessey algorithm. In [31], they looked for points of high curvature in the autocorrelation function. First image was smoothed to remove noise and then directional derivatives $I_x$ and $I_y$ were calculated. These were formed into a second moment matrix $C$:

$$C = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

The products of the image derivatives were also smoothed, denoted by the $\langle \rangle$. The value of this matrix at each pixel was examined for points of high curvature. Next they used the following “cornerness” operator:

$$R(x, y) = \text{det}(C) - \kappa \text{trace}^2(C)$$

which gives a maximum at points of high curvature. A value of $\kappa = 0.04$ was recommended in most of the literature. A value of $R$ can be calculated at every pixel of the image and corners extracted at the peaks of this distribution. Quddus and Fahmy [69] presented a wavelet-based scheme for detection of corners on 2D planar curves. Arrebola et al. introduced different corner detectors based on local [4] and circular [5] histogram of contour chain code. Chabat et al. [12] introduced an operator for detection of corners based on a single derivative scheme that had already been introduced in [93] by Yang et al. In [101], Zheng et al. proposed gradient-direction corner detector that was developed from the popular Plessey corner detector. Beaudet [6] proposed a determinant operator which has significant values only near corners. Lee et al. [45] proposed an algorithm for detecting and locating corners of planar curves based on the multi-scale wavelet transform. Tsai [86] proposed a method for boundary-based corner detection using neural networks. Kohlmann [41] proposed corner detection in
natural images based on the 2-D Hilbert transform. Nobel [65] proposed a solution to finding 'T', 'X', 'L' junctions which have a local 2D structure. Wu and Rosenfeld [89] proposed a technique which examines the slope discontinuities of the x and y projections of an image to find the possible corner candidates. Paler et al. [67] proposed a technique based on features extracted from the local distribution of grey level values. Rangarajan et al. [70] proposed a detector which tries to find an analytical expression for an optimal function whose convolution with the windows of an image has significant values at corner points. Davies [18] applied the generalised Hough transform to corner detection. Mehrotra et al. [48] proposed two algorithms for edge and corner detection. The first is based on the first directional derivative of the Gaussian and the second is based on the second directional derivative of the Gaussian. Trajkovic and Hedley [85] described a corner detection algorithm based on the property of corners that the change of image intensity should be high in all directions. Consequently, the corner response function was computed as a minimum change of intensity over all possible direction. Our survey suggested that Plessey, Kitchen and Rosenfeld, SUSAN, Mokhtarian and Suomela corner detectors are among the well-known methods in this area.

2.3 Active Contours

In recent years, active contour models or snakes have become one of the most powerful algorithms for image segmentation, boundary extraction, image matching and tracking. Our interest in active contours comes from their use as a user interface for our video database retrieval system. Energy minimising active contour models have been proposed by Kass et al. [37] as a top-down mechanism for locating features of interest in images. Amini et al. [2] discussed the problems of Kass et al.'s algorithm and presented an algorithm for active contours based on dynamic programming. Their algorithm needs proper setting of model parameters and does not lock on to objects of interest very accurately. It is also time consuming. Williams and Shah [88] presented a fast active contour model based on a greedy algorithm with three parameters; $\alpha$, $\beta$ and $\gamma$ and two thresholds. Therefore their method is more dependent on setting model parameters and the values of thresholds. They improved the time complexity
of energy minimising active contours model to $O(nm)$ in simple images with single object. $n$ and $m$ are the number of points and possible directions at each point on active contour. However this point-wise method does not guarantee convergence. Brownian Strings method [26] controlled the evolution of the active contour by a simulated annealing process which causes the contour to settle into the global minimum of a non-parametric and image-derived energy function. Due to obtaining energy function from the statistical properties of previously-segmented images, Brownian Strings method is more time consuming than energy minimising active contour models. Bayesian wavelet snake [96] was developed for identifying a closed-contour object with a fuzzy and low-contrast boundary. In [71], they proposed an active contour for shape extraction of objects based on deriving the Euler-Lagrange equations corresponding to the energy function. Their energy function takes into account two requirements of object isolation and smoothness of the contour. Hui et al. [47] proposed a robust snake model with using the reformulated internal energy and the combination of both region and edge information to enlarge the capture range of external energy. They also reduced the requirements of initial contours. Giraldi et al. [25] addressed the limitations of dynamic programming (DP) by reducing the region of interest for image segmentation through the use of Dual-T-Snake approach. The solution of this method consists of two curves enclosing each object boundary which allows the definition of more efficient search space for a DP technique. Kim et al. [39] presented new contraction energy of active contours independent of the object’s form for segmentation. This approach was developed for segmentation and used as a tracking tool for rigid or non-rigid objects. In [97], a new contour detection method based on the snake model was developed. The method consists of two steps; first locating the initial contour with an initialisation algorithm and second, locating the final contour with an improved snake algorithm. Xu and Prince [92, 91] proposed a new external force for active contours called gradient vector flow (GVF). The GVF was computed as a diffusion of the gradient vectors of a gray-level or binary edge map derived from the image. Ip and Shen [33] showed that existing shaped-based active contour models are not affine-invariant (AI) and presented that features based on areas formed by successive points along a contour are AI. Then they presented a AI snakes model based on these features. Cohen [16] used
a new efficient numerical method to find the shortest path of active contour model's energy between two iterations for presenting a new boundary detection approach in shape modelling. In [87], high performance snake based on spline representation and multiple stage energy minimisation process was proposed. Delagnes proposed a new contour model, “Adjustable Polygons” which is a set of active segments that can fit any object shape including corners. This approach can be used for tracking corners and any deformable object in moving images [19]. Eviatar and Somorjai [21] described a method for the application of active contours to biomedical images. The basic idea of active contours, the minimisation of an energy has been retained. However both the internal energy of snake and the solution method have been modified. Lam and Yuen [42] proposed unbiased active contour algorithm that used edge feature based tracking, modified the energy functions to correct the bias towards equal-distanced snaxels and small curvature. Davatzikos and Prince [17] considered the nature of the convergence problem through a study of the convexity of the active contour energy function. Our proposed active contour model; SAC which has been introduced in section 4.3 is based on reformulating internal energy of active contours in energy-minimising model using dynamic programming.

### 2.4 Feature Tracking Algorithms

Several feature point trackers have been proposed. Bretzner and Lindeberg[9] presented a framework for feature tracking in which a mechanism for automatic scale selection had been built into the feature detection stage. However except for scaling they have not considered any other transformations which may happen in a complex environment. In [20], the development and the implementation of a line segments based token tracker was described. Their tracking approach combines prediction and matching steps. Prediction step was a Kalman filtering based approach, which suffers from sudden changes in motion direction. Their correspondence approach was done in the search area around each corner using similarity function based on Mahalanobis distance between attributes chosen of the line segments. Lee and Deng [43] presented a three-frame matching method to compute the displacement field in three consecutive
frames for finding the correspondences. After extracting the corners of frames, they found candidate transition paths, which are formed by three consecutive corners. Each transition path was assigned an initial probability based on the similarity of curvatures of three corner points. These probabilities were iteratively modified and the paths with sufficiently high probabilities were taken as the correct transition paths. Yao and Challapala [94] proposed an algorithm for tracking a dynamic set of feature points over a sequence of images based on 2-D kinematic motion model which exploits the temporal information contained in a sequence. Broida et al. [10] introduced a recursive method of estimating 3D kinematics and structure of a rigid moving object from a sequence of noisy monocular images. They described models for the kinematics of rigid object, and for the observation of the discrete features of the object using a single camera. Lucas and Kanade [46] have worked on the tracking problem and proposed a method for registering two images for stereo matching based on a translation model between images. From the initial work of Lucas and Kanade, Tomasi and Kanade [83] developed a feature tracker based on the “sum of squared intensity differences (SSD)” matching measure, using a translation model. Then, Shi and Tomasi [77] proposed an affine translation model. Their system classified a tracked feature as “good (reliable)” or bad (unreliable) according to the residual of the match between the associated image region in the first and current frames. If the residual exceeded a user-defined threshold, the feature was rejected. Visual inspection of the results demonstrated good discrimination between good and bad features, but the authors did not specify how to reject bad features automatically. Over small inter-frame motion, the translation model has higher reliability and accuracy than the affine model. However, the affine method is preferable and more adequate over a longer time span.

Tommasini et al. [84] proposed a robust tracker based on the work of Shi and Tomasi [77] by introducing an automatic scheme for rejecting spurious features. Details about Tommasini et al.’s tracker is presented in 2.4.1. In Tommasini et al.’s algorithm a feature is defined as a region that can be tracked easily from one frame to the other. Then in their algorithm feature is not always a corner. Also in their framework, a feature can be tracked reliably if they can find a numerically stable solution to the equation of their assumed motion model. They did not track single features, but windows of features,
2.4. Feature Tracking Algorithms

and they looked for windows that contain sufficient texture. They gave a definition of what sufficient texture is for reliable feature tracking. Unfortunately, different points within a window may behave differently. The points are moving at different velocities and may disappear or appear a new. Therefore how do they determine that they are following the same window, if its contents change over time? Second, if they measure the displacement of the window, how are the different velocities combined to give the one resulting vector? Their solution to the first problem was residue monitoring. They kept checking that the appearance of a window has not changed too much. If it has, they discarded the window. For second problem rather than describing window changes as simple translations, they modelled the changes as a more complex translation, such as affine map. Selecting affine motion model will cause another problem, the size of the match window. This is because more parameters to estimate require the use of larger windows to constrain the parameters sufficiently. On the other hand windows should be as small as possible, combined with good noise rejection and the size of the window can improve performance considerably. Over all, both of their solutions are not practical for unconstrained motions in video sequences. If unconstrained motion occurs, none of these assumptions remain satisfied. Then not only the match window will be discarded but also all the tracked features in and around the location of unconstrained motion occurrence will be rejected based on their rejection rule; X84, since they assume a Gaussian distribution of the residual for the good features. Therefore in unconstrained motion the residuals are not samples from the Gaussian distribution of good features. Furthermore in their algorithm, occlusion problem has not been addressed. Note that in Tommasini et al.'s algorithm feature extraction and tracking are combined but in our tracking algorithm, feature extraction is separate from the tracking algorithm. We are tracking the corners which have been extracted by the CSS corner detector. Then actually these two trackers are completely different but we compared our tracker with Tommasini et al.'s tracker as it is a popular feature tracker. Many other authors have referred to Tommasini et al.'s tracker. Therefore obtaining better results for our tracker in comparison to Tommasini tracker confirms the better performance of our tracker. Some other techniques have used only corners for tracking. Shapiro et al. [75] used corners and focused on solving the correspondence problem in an image sequence. However
their algorithm selected the match with highest score of cross-correlation as the best match which is not always correct. In [72], they used corners as object tokens which are tracked independently. To extract corners, Harris corner detector was applied only in areas of image plane containing useful information. Correspondence problem was solved using a function which compares the normalised magnitude of the difference vectors between each candidate corner vector and the tracker's current corner vector. But still the problem of combining different vectors into the resulting vector remains.

In [78], the problem of obtaining a good initial set of corner matches between two images was tackled and several different matching metrics were evaluated. However there is no indication of which one is the best in feature tracking area. A 3D model based tracking using texture learning and matching was proposed in [23]. The snake-based tracker is CONDENSATION (Conditional Density Propagation) tracker of Isard and Blake [34]. They avoided the Kalman filter and its Gaussian assumption in order to find a tracker which works well in the presence of clutter. In their algorithm they established a stochastic framework for tracking curves in visual clutter, using a sampling algorithm. In [7] an approach was described for tracking rigid objects by extending work on eigenspace representations, estimation techniques, and parameterised optical flow estimation. Smith and Brady [79] proposed a system employing corner features which incorporates cluster tracking using an affine model but no examples of non-rigid motion or non-smooth motion are given. Finally these techniques and many of the others do not address the problem of recovering points lost during the tracking. Our proposed technique in section 5.3 takes a step towards this problem.

2.4.1 Tommasini Feature Tracker

In [84], Tommasini et al. extended the Shi-Tomasi-Kanade [77, 83] tracker by introducing an automatic scheme for rejecting spurious features. Therefore first the Shi-Tomasi-Kanade tracker will be briefly described here.

Consider an image sequence \( I(x,t) \), where \( x = [u,v]^T \), are the coordinates of an image point. If the time sampling frequency (that is the frame rate) is sufficiently high, we can assume that small image regions undergo a geometric transformation, but their
2.4. Feature Tracking Algorithms

Intensities remain unchanged:

\[ I(x, t) = I(\delta(x), t + \tau) \]  \hspace{1cm} (2.3)

Where \( \delta(.) \) is the motion field, specifying the warping that is applied to image points. They rely on the fast-sampling hypothesis to approximate the motion with a translation, i.e.

\[ \delta(x) = x + d \]

where \( d \) is a displacement vector. The Shi-Tomasi-Kanade tracker task is to compute \( d \) for a number of automatically selected point features for each pair of successive frames in the sequence. As the image motion model is not perfect, and because of image noise, Eq.2.3 is not satisfied exactly. The problem is then finding the displacement \( d \) which minimise the SSD residual

\[ \epsilon = \sum_{W}[I(x + d, t + \tau) - I(x, t)]^2 \]  \hspace{1cm} (2.4)

where \( W \) is a given feature window centred on the point \( x \). In the Shi-Tomasi-Kanade tracker this problem have been solved by means of a Newton-Raphson iterative search. They approximated \( I(x + d, t + \tau) \) with its first-order Taylor expansion based on fast-sampling assumption:

\[ I(x + d, t + \tau) \approx I(x, t) + \nabla I(x, t)^T d + I_t(x, t) \tau \]  \hspace{1cm} (2.5)

where

\[ \nabla I^T = [I_u, I_v] = \left[ \frac{\partial I}{\partial u}, \frac{\partial I}{\partial v} \right] \text{ and } I_t = \frac{\partial I}{\partial t}. \]

We can then rewrite the residual (2.4) as

\[ \epsilon \approx \sum_{W}(\nabla I(x, t)^T d + I_t(x, t) \tau)^2 \]  \hspace{1cm} (2.6)

To minimise the residual (2.6), they differentiated it with respect to the unknown displacement \( d \) and set the result to zero, obtained the linear system

\[ Cd = g \]  \hspace{1cm} (2.7)
Chapter 2. Literature Review

Where

$$C = \sum_W \begin{bmatrix} I_u^2 & I_u I_v \\ I_u I_v & I_v^2 \end{bmatrix}$$  \hspace{1cm} (2.8)

$$g = -\tau \sum_W I_t[I_u I_v]^T$$  \hspace{1cm} (2.9)

If \(d_k = C^{-1}g\) is the displacement estimate at iteration \(k\), and assuming a unit time interval between frames, the algorithm for minimising 2.6 is the following:

$$\begin{cases} 
  d_0 = 0 \\
  d_{k+1} = d_k + C^{-1} \sum_W [(I(x,t) - I(x + d_k, t + 1))\nabla I(x, t)]
\end{cases}$$

For their algorithm a feature was defined as a region that can be tracked easily from one frame to the other. In this framework, a feature can be tracked reliably if a numerically stable solution to Eq.2.8 can be found, which requires that \(C\) is well-conditioned and its entries are well above the noise level. In practice, since the larger eigenvalue is bound by the maximum allowable pixel value, the requirement is that the smaller eigenvalue must be sufficiently large. Calling \(\lambda_1\) and \(\lambda_2\) the eigenvalues of \(C\), they accepted the corresponding feature if

$$\min(\lambda_1, \lambda_2) < \lambda_t$$

Where \(\lambda_t\) is a user-defined threshold. The Shi-Tomasi system classified a tracked feature as “good (reliable)” or bad (unreliable) according to the residual of the match between the associated image region in the first and current frames. If the residual exceeded a user-defined threshold, the feature was rejected. Over small inter-frame motion, the translation model has higher reliability and accuracy than the affine model. However, the affine method is preferable and more adequate over a longer time span.

By considering an affine model,

$$\delta(x) = Mx + d$$

Where \(M\) is a 2 x 2 matrix accounting for affine warping, and can be written as \(M = 1 + D\), with \(D = (d_{ij})\) a deformation matrix and \(1\) the identity matrix. Similar to
the translational case, we can estimate the motion parameters, \( D \) and \( d \) by minimizing the residual

\[
\varepsilon = \sum_{W} [I(M\mathbf{x} + d, t + \tau) - I(x, t)]^2
\]

Therefore as result in the Shi-Tomasi tracker, visual inspection of the results demonstrated good discrimination between good and bad features, but they did not specify how to reject bad features automatically. This problem was solved in [84] by introducing an automatic scheme for rejecting spurious features.

**The X84 Rejection Rule**

In Tommasini et al. [84] tracker when the two regions over which they computed the residual were bad features, it means that the residual is not a sample from the Gaussian distribution of good features: it is an outlier. Hence in their algorithm, the detection of bad features reduced to a problem of outlier detection. This is equivalent to the problem of estimating the mean and variance of the underlying Gaussian distribution from the corrupted data \( e_i \), the residuals between the \( i \)th feature in the last frame and the same feature in the first frame. To do this, they employed a simple rejection rule, X84 [29], which uses estimates for location and scale to set a rejection threshold. The median is a robust location estimator, and the Median Absolute Deviation (MAD), defined as

\[
MAD = \text{mad} \{|e_i - \text{med}_j e_j|\}
\]  

is robust estimator of the scale. The X84 rule presents rejecting values that are more than \( k \) Median Absolute Deviations away from the median. A value of \( k=5.2 \) was suggested in practice.

### 2.5 Content-based Video Database Retrieval

Recently several content-based image and video retrieval systems have been developed. These systems differ in terms of features extracted, degree of automation reached for feature extraction and level of domain independence. Query By Image Content (QBIC)
is a content-based image retrieval system which relies on a combination of automatic and semi-automatic image indexing. QBIC [22] uses colour, texture, shape and sketch features which are stored along with the images. Photobook [68] is a content-based image retrieval system including three flavours: face, shape for recognising images of tools, and texture. The types of images that can be retrieved with photobook are quite restricted and each category requires a separate content representation and retrieval model. VisualSEEK [35] represents the global colour information using colour histogram similarity measures for comparing the image colour features. VideoQ [13] segments video based on global motion, and tracks objects based on colour, motion and edge information. It also supports sketch queries based on animated sketches, and thus includes motion and temporal aspects of objects occurring in a video. In [3], first a video sequence was splitted to shots, then a few representative frames called r-frames were extracted from each shot and r-frames descriptors were computed based on colour, texture and motion. The first and the last shot frame have been suggested as representative frames. In continue their approach is based on statistical analysis of feature distances. Zhang et al. [99] described a representation that extended the ideas of the key-frame to “key-object”. Typically in retrieval applications, a video sequence is subdivided in time into a set of shorter segments which each of them contains similar content. The segments are represented by 2D representative images called “key-frames” that reduce amount of data. However, key-frames do not describe the motions and the actions of objects within the segment. Key-objects consist of regions within a key-frame that move with similar motion and allow a retrieval system to present more information to users. They used motion segmentation techniques to identify with in a shot a group of regions that move coherent motion. Then they tracked the regions throughout of the shot. The classical approach to content-based video access has been “frame-based”, consisting of shot boundary detection followed by selection of key frames that characterise the visual content of each shot. In these systems, the basic indexing unit is the shot and visual representation of shots is provided by a set of static keyframes. Although these frame-based access methods may be well-suited for broadcast and video-on-demand but they do not provide access to individual video objects. In [27], an 'object-based' approach to temporal video partitioning and content-
2.5. Content-based Video Database Retrieval

Based indexing was introduced. They represented each video object by an adaptive 2D triangular mesh. Then a mesh-based object tracking scheme was employed to compute the motion trajectories of all mesh node points until the object was defined to detect content changes. The tracking step defined a segmentation map with polygonal object boundaries for each video object plane (VOP). A VOP is a 2D snapshot of a video object at a particular time instant. The entry and exit frames of video objects as well as their segmentation maps were available. In [14], a video data model and annotation language for describing complex information of video data was presented. Jiang et al. [36] introduced a video data model called VideoText for semantic content-based video query and retrieval. VideoText is based on text annotations associated with logical segments and a corresponding query language. Gevers and Smeulders [24] focused on image retrieval by image example, where an example query image is given by the user on input. As the basic idea to image retrieval by image example is to extract characteristic features from target images, they used colour information and they proposed a new set of colour features. From this set, they selected various colour features to construct colour pattern-cards. Colour pattern-cards indicate whether a particular colour feature value is dominantly present in an image or not. Matching measures were defined, expressing similarity between colour pattern-cards due to a substantial amount of object occlusion and cluttering. Therefore the problem of content-based image retrieval was reduced to the problem to what extent the colour pattern-card derived from the query image is similar to the colour pattern-card constructed for each image in the image database. The objective of [76] was to study the use of object shape as a feature for image indexing and retrieval. They presented a multi-scale similarity matching method for encoding and comparing shapes. They approached the problem of establishing correspondence of two shapes as an error minimisation process of matching two feature sets. In [82], an image was divided into a number of non overlapping blocks, and each individual block was abstracted as a unique feature point labelled with its spatial location, dominant hue, and dominant saturation. For each set of feature points labelled with the same hue or saturation, they constructed a geometry-based spatial colour indexing for image retrieval. Actually, they encoded the spatial colour information using geometric triangulation, which is translation, rotation, and scale independent.
Yoshitaka and Tadao [95] surveyed content-based retrieval for multimedia databases from the point of view of three fundamentals: first ability to manage spatio-temporal relationships between pieces of video data, second recognition and/or interpretation process of contents in multimedia data, and third query representation. They proposed two principal ways for the representation of queries, namely, “query by subject/object” and “query by example (QBE)”. In processing a query by example, the system computes the static feature vectors of the query image and compares them with the feature vectors pre-stored in the feature database. Also in QBE, a query condition for non-textual data was presented, for example, in the form of a rough sketch, a rough painting with colours, or a motion example of a trajectory. Such representations express the query condition for non textual data better than keywords. It is often difficult to express slight differences of shapes, colour or spatio-temporal relations with keywords. QBE works well for content-based retrieval in the case where contents are found in terms of a single data type. However, the QBE approach is not adequate when two or more heterogeneous types of data form the content. Rather “query by subject/object” is more suitable for such cases, where keywords can well represent the semantic content. A significant amount of work has been carried out on content-based video retrieval. Among the existing techniques, we are believed that our proposed retrieval system due to its properties which have been mentioned in Sec.1.3 is more efficient and applicable. Our experimental results on very different video databases which have been shown in 7.5 have supported this claim.
Chapter 3

Multi-scale Corner Detector

3.1 Introduction

In many applications, the detection of feature points in images is essential. Feature points provide sufficient constraints to compute image displacements reliably, and the data is reduced by orders of magnitude compared to the original image data. One of the most intuitive types of feature point is a corner. A large number of methods have been proposed for corner detection. One of these methods is multi-scale corner detection. Our interest in corner detection is for corner tracking which is an important issue in computer vision and its applications. Real-time corner tracking which is used for multimedia applications needs robust corners that can be identified and tracked from frame to frame in image sequences. This chapter presents the multi-scale corner detector [49] which is the enhanced curvature scale (ECSS) corner detector using different scales of smoothing. By applying the ECSS corner detector to the frames in video sequences, the positions of some corners can change from frame to frame. The percentage of the ECSS corner detector robustness is higher than other famous corner detectors which was reported in [51]. The positions of corners in an image, extracted by the ECSS corner detector, depend on the edge contours of that image and their local maxima. Therefore the problem that the positions of corners are changeable can be decreased easily. We can minimise this problem using different levels of smoothing in the stage of Canny edge detector [11] of the ECSS corner detector. By applying Canny edge
detector with different scales of smoothing, we consider all changes that may happen in
the edge contours of a frame in comparison to its next frame in video sequences due to
any transformations between these two frames or variation of illumination or moving
camera. Therefore we select the local maxima of all the edge contours of that frame as
corner candidates which are extracted using the multi-scale edge detector. The follow­
ing is the organisation of the remainder of this chapter. Since the ECSS corner detector
is an improvement of the CSS corner detector [62], in continue after explaining the CSS
method generally in section 3.2, original CSS corner detector and its shortcomings are
described in Sec.3.3 and Sec.3.4 respectively. Then the new corner detector; ECSS is
presented in 3.5. Afterwards, section 3.6 explains the multi-scale corner detector and
the multi-scale edge detector in detail.

3.2 The Curvature Scale Space Technique

The CSS technique is suitable for extraction of curvature features such as curvature
extrema or zero-crossing points from an input contour at multiple scales. The curve \( \Gamma \)
is first parametrised by the arc length parameter \( u \):

\[
\Gamma(u) = (x(u), y(u)).
\]  

(3.1)

An evolved version \( \Gamma_\sigma \) of \( \Gamma \) is then computed. \( \Gamma_\sigma \) is defined by [56]:

\[
\Gamma_\sigma = (X(u, \sigma), Y(u, \sigma))
\]  

(3.2)

where

\[
X(u, \sigma) = x(u) \otimes g(u, \sigma) \quad Y(u, \sigma) = y(u) \otimes g(u, \sigma)
\]

Note that \( \otimes \) is the convolution operator and \( g(u, \sigma) \) denotes a Gaussian of width \( \sigma \). The
process of generating evolved versions of \( \Gamma \) as \( \sigma \) increases is referred to as the evolution
of \( \Gamma \). In order to find curvature zero-crossings or extrema from evolved versions of the
input curve, one needs to compute curvature accurately and directly on an evolved
version \( \Gamma_\sigma \). Curvature \( \kappa \) on \( \Gamma_\sigma \) is given by [56]:

\[
\kappa(u, \sigma) = \frac{X_u(u, \sigma)Y_{uu}(u, \sigma) - X_{uu}(u, \sigma)Y_u(u, \sigma)}{(X_u(u, \sigma)^2 + Y_u(u, \sigma)^2)^{1.5}}
\]  

(3.3)
where the first and second derivatives of $X$ and $Y$ can be computed from the following:

$$X_u(u, \sigma) = x(u) \otimes g_u(u, \sigma) \quad X_{uu}(u, \sigma) = x(u) \otimes g_{uu}(u, \sigma)$$

$$Y_u(u, \sigma) = y(u) \otimes g_u(u, \sigma) \quad Y_{uu}(u, \sigma) = y(u) \otimes g_{uu}(u, \sigma)$$

3.3 Overview of Original CSS Corner Detector

The following is the algorithm (Orig-CSS) used by Mokhtarian and Suomela [62] to detect corners of an image.

- Apply Canny edge detector to the gray level image and obtain a binary edge image.
- Extract image edge contours from the output of Canny edge detector, fill the gaps and find T-junctions.
- Compute curvature at a high scale, $\sigma_{high}$, for each edge contour.
- Consider those local maxima whose curvature is above a threshold $t$ and twice as much as one of the neighbouring local minima, as initial corners.
- Track the corners to the lowest scale to improve localisation.
- Compare the T-junctions to other corners and remove one of the two corners which are very close.

In this algorithm, the first step is to extract edges from the original image using Canny edge detector. Edge detector may produce gaps in some edges. To achieve the best performance of the CSS corner detector these gaps should be filled at this stage. When the edge extraction method arrives at the endpoint of a contour, it performs two checks:

- If the endpoint is nearly connected to another endpoint, fill the gap and continue the extraction.
- If the endpoint is nearly connected to an edge contour, but not to another endpoint, mark this point as a T-junction corner.
Chapter 3. Multi-scale Corner Detector

The next step is to smooth edge contours by a Gaussian function and compute curvature at each point of the smooth contours. The width of the Gaussian function indicates the scale of the smoothing and must be large enough to remove noise and small enough to retain the real corners. The third step is to determine corner candidates on smoothed contours, which are normally the local maxima of absolute curvature. The final step is localisation. As a result of smoothing, the edge contours shrink. The locations of corners on the shrunk contours differ from the locations of actual corners on the original contour. For each smoothed contour, the level of smoothing is reduced gradually and the corners are tracked down to the original contour.

3.4 The Original CSS Shortcomings

The second step of Orig-CSS algorithm is to smooth edge contours by Gaussian function and compute curvature at each point of the smooth contours. The width of the Gaussian function indicates the scale of the smoothing and must be large enough to remove noise and small enough to retain the real corners. Interestingly, we discovered that this scale should not be the same for all edge contours of the image. While for long contours, a large scale may be suitable; short contours need smaller scale of smoothing. The remedy to this shortcoming is to choose different scale of smoothing for contours with different lengths as described in the following section. The third stage of Orig-CSS algorithm is to determine corner candidates on smoothed contours, which are normally the local maxima of absolute curvature. However, we noticed that for long contours, the absolute curvature function must be smoothed prior to initial corner selection. Note that smoothing of absolute curvature function is different from smoothing of contours in 3.5.1. Furthermore Orig-CSS performance depends on the selection of threshold value, $t$. The proper value of $t$ may change from one image to another. It is also subject to change for a particular image which transforms under rotation or scaling. Therefore, methods with threshold are not robust and their performance depends on selecting the threshold values. The ECSS corner detector mainly tackles these problems. It is described in more detail in the following section.
3.5 New Method of CSS Corner Detection

The outline of ECSS [57] corner detector is as following:

- Extract edges from the original image.
- Extract image edge contours, filling the gaps and finding T-junctions.
- Use different scales of the CSS for contours with different lengths.
- Compute the absolute curvature on the smoothed contours.
- Smooth the absolute curvature function for long contours.
- Detect initial local maxima of the absolute curvature for short contours.
- Detect initial local maxima of the smoothed absolute curvature functions for long contours.
- Consider those local maxima as initial corners that are more than twice as much as one of the neighbouring local minima.
- Track the corners down to the lowest scale for each contour to improve localisation.
- Compare the T-junction corners to the corners found using the curvature procedure to unify close corners.

The new steps have been considered in detail one by one in the remainder of this section.

3.5.1 Using Different Scales of CSS

After extracting image edge contours, the number of contours and the number of points on each contour are known. Based on the number of points on each contour, our algorithm categorises all the image edge contours to three categories: long, medium and short contours. Through many experiments carried out on our image databases, contours which consist of 3 to 200 points are considered short. Also, contours which consist of 200 to 600 points are considered medium and more than 600 points are considered long contours. Note that these values did not change in the program. Furthermore, we
consider $\sigma_{\text{high}}$ as 4, 3 and 2 for long, medium and short contours respectively. These values have also been selected through many experiments. Evaluation carried out for the ECSS corner detector in chapter 7 shows that these selections perform well for a large number of images. As a result, short contours are not smoothed too much which might remove their corners and long contours are smoothed enough. In Fig.3.1, one of our test images and its edge contours with two marked contours, short C1 and long C2 have been illustrated. The effect of selecting different scales in computation of absolute curvature for long and short contours can be seen in Fig.3.2. In Fig.3.2(a), the absolute curvature of C2 with $\sigma_{\text{high}}=4$ has fewer false maxima due to noise in comparison to Fig.3.2(b). Fig.3.2(b) shows the computation of absolute curvature of C2 with $\sigma_{\text{high}}=2$. Obviously for long contours computation of absolute curvature should be done using $\sigma_{\text{high}}=4$. Also Fig.3.2 (c) illustrates the computation of absolute curvature of C1 with $\sigma_{\text{high}}=4$. If we use the local maxima of this absolute curvature function for detecting corners, only two corners are detected. Therefore using high scale for smoothing short contours removes some local maxima of absolute curvature of these contours that correspond to real corners. In other words, on short contours computing of absolute curvature should be done using $\sigma_{\text{high}}=2$. Fig.3.2(d) shows the absolute curvature of C1 with $\sigma_{\text{high}}=2$. It can be seen that four corners are available from this figure.
3.5. New Method of CSS Corner Detection

3.5.2 Smoothing the Absolute Curvature Function of Long Contours

In this stage, after smoothing edge contours for computation of absolute curvature, some false maxima due to noise can still be seen (see Fig. 3.2(a)). The simplest solution is to compute the absolute curvature of contour C2 with higher σ such as 8. But as we mentioned in Sec. 3.3, if higher σ is chosen not only false corners but also many real corners are removed as well. Therefore our solution in this stage is to smooth absolute curvature function of long contours by σ=4. This has been illustrated in Fig. 3.3(a). Note that smoothing of absolute curvature function is different from smoothing of contours in Sec. 3.5.1. In Fig. 3.3(a) in comparison to Fig. 3.2(a), after smoothing, many false maxima of absolute curvature are removed. In this step, if curvature function of short contours becomes smoothed, as it is seen in Fig. 3.3(b), a number of real corners are lost. As we can see for C1, in Fig. 3.3(b) only two local maxima remain that indicate two corners, whereas C1 is the window of airplane with four corners. Final criterion for removing false corners, after initialising local maxima points is to compare the initial local maxima with two neighbouring local minima. The curvature of a corner should be more than twice as much as one of the neighbouring local minima. With this criterion,
false corners such as 1, 2 and 3 are removed in comparison to the nearest local minima of absolute curvature in Fig. 3.3(a). The positions of initial corners of figures 3.3(a) and 3.2(d) after taking into account this criterion have been illustrated in Fig. 3.4(a) and 3.4(b) respectively. The method finds four corners on C1 and no false corners on C2. C2 should have nine corners that are seen in Fig. 3.4(a). Remember that for short contours our method first computes absolute curvature with $\sigma_{\text{high}}=2$, then uses the final criterion that we discussed above. The positions of corners of C1 are marked in Fig. 3.2(d) and have been illustrated in Fig. 3.4(b).

### 3.5.3 Tracking

After the initial corner points are located, tracking is applied to the detected corners. As the corners were detected at scale $\sigma_{\text{high}}$ the corner localisation might not be good. We compute curvature at a lower scale and examine the corner candidates in a small neighbourhood of the previous corners. Corner locations are updated, if needed, in
3.6. Multi-scale Edge Detector

this neighbourhood. Note, if initial corners in one contour are extracted at \( \sigma_{\text{high}}=4 \), tracking for this contour can be accomplished at \( \sigma=3 \), \( \sigma=2 \) and \( \sigma_{\text{final}}=1 \). If initial corners are extracted at \( \sigma_{\text{high}}=2 \), tracking can be accomplished at \( \sigma_{\text{final}}=1 \). In other words tracking is continued down to a very low scale. The localisation of corners for contour C2 and C1 after tracking have been shown in Fig.3.5(a) and (b) respectively.

This process results in very good localisation. The number of corners is determined at the initial \( \sigma_{\text{high}} \) and tracking only changes the localisation, not the number of corners. Since the location of corners do not change substantially during CSS tracking, only a few other curvature values need to be computed.

3.5.4 Unifying Close Corners

As described before, corners are detected using ECSS technique taking T-junctions into consideration. In some cases the two methods mark the same corner. The final part of the ECSS is to examine T-junctions and the corners that are extracted from tracking in Sec.3.5.3. If they are very close to each other, the T-junction corners are removed.

3.6 Multi-scale Edge Detector

The Canny edge detector smoothes an image by a Gaussian filter prior to edge detection to remove the noise. The level of smoothing which is indicated by \( \sigma \), the filter width,
affects the output of the edge detector and any consequent process which may be applied to the edge contours. In Fig.3.6, three different binary images corresponding to three levels of smoothing; 1, 2, and 3 are produced from test image; Fig.3.6(a) which is the input grey level image. From this figure it is visible, at a low level of $\sigma$ such as 1, the output is rather noisy, and the edges tend to be cut more frequently, but more details from the image edges can be grabbed. At higher levels of $\sigma$ such as 2, and 3, however, longer edges are observed but some details disappear. We believe that by taking into account both lower and higher levels of smoothing more information can be extracted from the image which is used later on in other stages of the ECSS corner detection algorithm. We also emphasise that selecting $\sigma$ more than three, does not give more information about the edge contours as we got from previous levels of $\sigma$. In Fig.3.6, by using the multi-scale edge detector, due to filling the gaps at points 1, 2, 3, and 4 in Fig.3.6(c) in comparison to the Fig.3.6(b) and also filling the gaps at points 5, and 6 in Fig.3.6(d) in comparison to the Fig.3.6(c), we do not lose those local maxima that should be at points a, b, and c of the edge contours of Fig.3.7(b). Accordingly, the corners of the test image at points a, b, and c in Fig.3.7(a) are not missed as it has been shown in the Fig.3.7(b) for $\sigma=2$. Now it is more clear why using the multi-scale edge detector with three different scales of smoothing can affect the extracting of all possible positions of the local maxima in the edge contours of a frame.

In Fig.3.8(a), (b) and (c), three outputs of the ECSS corner detector using a multi-scale canny edge detector with three values of $\sigma$ have been illustrated. Therefore by considering all the corners which are marked in these three images as the output of the
3.6. Multi-scale Edge Detector

Figure 3.7: Missed corners a, b and c in test image (a) based on selecting the local maxima of its edge contours (cnts.) in (b) as corner candidates due to the gaps at points a, b, c of these contours.

Figure 3.8: Three outputs of the multi-scale corner detector and its final output on bream image

multi-scale corner detector on test image, we can have all the possible corners in this image. This procedure will be carried out for all frames of an image sequence. Then due to slight motion between two successive frames, we can have a sufficient number of robust corners to be tracked. The position of these corners remain stable among these frames for matching and tracking process. The final results of the multi-scale corner detector can be seen in Fig.3.8(d). In fact if we compare the corners which have been demonstrated on three images of Fig.3.8, some of them are marked twice (corner at point 2) or thrice (corner at point 1). Those corners which are marked more than once should be removed from the final corners in the output of the multi-scale corner detector. Therefore we consider a small window around each corner of Fig.3.8(a) and compare this corner and its eight neighbours with all corners of Fig.3.8(b) and (c).
Each corner of Fig.3.8(b) and (c) which has the same coordinates with that corner and its eight neighbour in our comparing window in Fig.3.8(a) will be removed. Finally the corners of Fig.3.8(a) and all the corners of Fig.3.8(b) and (c) that are not removed will be considered as the output of the multi-scale corner detector which has been shown in Fig.3.8(d).
Chapter 4

Accurate and Fast Active Contour Models

4.1 Introduction

In recent years, active contour models or snakes have become one of the most powerful algorithms for image segmentation, boundary extraction, image matching and tracking. Our interest in active contour models comes from their use as a user interface for multimedia database retrieval using shape content. In such a shape-based multimedia database retrieval system, the user submits a query shape and then will expect the system to locate all instances of similar shapes in database. An active contour can be successfully used for these tasks if it has good accuracy and high speed. A number of models have been proposed for active contours. The performance of active contours in these models depends on the proper setting of model parameters and the initial snake. Due to these shortcomings, majority of existing active contour models often fail to converge to the desired solution especially in complex images in which objects are very close to each other, see Fig.4.1. The general idea of active contour models is as follows: first user places a closed contour around a target object in an image. Then the constraint forces act on initial snake and push it towards object until it locks on to object as close as possible. The basic active contour model was proposed by Kass et al. [37]. Its algorithm has a number of shortcomings which Amini et al. [2] pointed
Chapter 4. Accurate and Fast Active Contour Models

Figure 4.1: One example of an active contour model failing to converge to the interest object

out and the solution was presented as a discrete multi-stage decision process using a “time-delayed” discrete dynamic programming algorithm. The behaviour of an active contour is generally controlled by internal and external forces. The internal forces act to shorten and enforce smoothness of active contour and the external forces move it towards image features such as image edges. The total energy of the active contour is as follows:

\[ E_{\text{total}} = E_{\text{int}}(i) + E_{\text{ext}}(i) \]  \hspace{1cm} (4.1)

where \( n \), \( E_{\text{int}} \) and \( E_{\text{ext}} \) are the number of points, internal energy and external energy of active contour respectively. Discretized \( E_{\text{int}} \) for an active contour represented as \( v(s) = (x(s), y(s)) \), can be computed as:

\[ E_{\text{int}}(i) = \frac{1}{2} (\alpha_i |v_i - v_{i-1}|^2 + \beta_i |v_{i+1} - 2v_i + v_{i-1}|^2) \]  \hspace{1cm} (4.2)

where \( v_i \) refers to the \( i^{th} \) point of the active contour and similarly for \( v_{i-1} \) and \( v_{i+1} \). In [2], the minimisation of Eq.4.1 was viewed as a discrete multistage decision process. For each stage the total energy of active contour was computed to be,

\[ E_t(i, j, k) = \min_{0 \leq m \leq N} \{ E_t(i - 1, k, m) + E_{\text{ext}}(v_{i-1} \oplus k) + \frac{1}{2} (\alpha_i |v_{i-1} \oplus k - v_{i-2} \oplus m|^2 + \beta_i |v_i \oplus j - 2v_{i-1} \oplus k + v_{i-2} \oplus m|^2) \} \]  \hspace{1cm} (4.3)

For an explanation of Eq.4.3, see section 4.2. Internal energy of active contour in these formulas is composed of two parts; the first part, helps to reduce the length of the active contour during its movement until it locks on to interest object. The second term, is
4.1. Introduction

the curvature of active contour and smoothes it through these stages. In Amini et al.'s algorithm (AMI model), the three points $v_i$, $v_{i-1}$ and $v_{i-2}$ were used to estimate the curvature of the active contour at point $v_i$. Also the external energy of active contour at this point was calculated as the distance from its previous point to the nearest edge of the underlying image. Then due to both of these problems, locking on to interest object does not occur very accurately especially at some points on the boundary of object where curvature changes very quickly. Furthermore AMI model is low speed especially in complex images. In this chapter the remedy to these shortcomings are presented as follows:

First, a reformulated internal energy is proposed to improve the computation of curvature at point $v_i$ by making use of the three points $v_{i-1}$, $v_i$ and $v_{i+1}$. Also external energy of active contour at any point is defined as its distance to the nearest edge of underlying image. This proposed active contour model is named ICEAC model. ICEAC stands for Improved Curvature Estimated Active Contour.

Second, an accurate and high speed active contour model, SAC is proposed based on reformulating internal energy by removing the curvature part and using Gaussian filtering with low scale for smoothing. The SAC stands for Smoothed Active Contour. The SAC model has only one parameter that affects the internal energy of active contour and as a result of using the CSS technique for smoothing, it is more independent of model parameter setting and initial snake.

Third, another accurate and high speed active contour model, ASAC is proposed based on reformulating internal energy by removing the curvature part and using adaptive curvature scale space (CSS) filtering for smoothing. By applying the adaptive CSS filtering, we smooth the input contour of new algorithm (which is the user initial snake or the output of each iteration of the energy minimising process) using the Gaussian function with adaptive scale (the standard deviation of the Gaussian, $\sigma$ is referred to as scale) until input contour reaches the underlying image edges. The ASAC stands for Adaptive Smoothed Active Contour. The ASAC model has also one parameter that affects the internal energy of active contour. Images with single and multiple objects are selected to evaluate the capability of third proposed methods. The results show that the ICEAC model [50, 52] locks on to the interest object completely like a membrane
or thin plate and the SAC [58] model converges quickly to final solution using rough user initial snakes and approximate setting of model parameter. The ASAC [61] model also converges very quickly to final solution without losing the smoothness in the final active contour shape. Through many experiments on complex and simple images, it has been discovered that the SAC model has quicker performance in complex images while the ASAC model is faster in single object images. Therefore we have selected the SAC active contour model as a user interface for our retrieval system. The following is the organisation of the remainder of this chapter. The estimation of the curvature for accurate localisation of active contours is presented in section 4.2. Our method for fast active contours is proposed in section 4.3 followed by introducing the ASAC model in section 4.4.

4.2 Improved Curvature Estimation for Active Contours

In [2], the minimisation of the total energy of active contour was viewed as a discrete multistage decision process. As we mentioned in section 4.1, for each stage the total energy of active contour was computed from Eq.4.3. Note the following points about Eq.4.3:

- \( E_t(i, j, k) \) denotes the total energy of active contour at iteration \( t \). Energy values are stored in a 3D matrix. For any point of the active contour, we have a 2D matrix with \( j \) rows and \( k \) columns. \( k \) is 9 due to the possible directions for each point of active contour to move to a new position. Each point can remain in previous location or move to one of its eight neighbours. Then the variation range of \( k \) and \( j \) is 0 to 8.

- \( N \) presents the number of possible directions for each point of active contour, then \( N=9 \) and \( 0 \leq m \leq 8 \).

- \( v_i \) and \( v_{i-1} \) illustrate the \( i^{th} \) point and its previous one on the active contour. For example the previous point for starting point is the end point due to active contour being closed.
4.2. Improved Curvature Estimation for Active Contours

- $v_{i-1} \oplus k$ represents the $k^{th}$ neighbour of point $v_{i-1}$ on the active contour. The zeroth neighbour refers to the original location of a point, not its neighbours.

- $|v_{i-1} \oplus k - v_{i-2} \oplus m|^2$ means the distance between the $k^{th}$ neighbour of point $v_{i-1}$ and the $m^{th}$ neighbour of its previous point.

- $E_{ext}(v_{i-1} \oplus k)$ shows the distance from the $k^{th}$ neighbour of point $v_{i-1}$ on active contour to the nearest edge of the underlying image.

- $E_t(i - 1, k, m)$ represents an element of energy matrix in iteration $t$ at row $k$ and column $m$. Actually it is the total energy of active contour at previous point of $v_i$ in row $k$ and column $m$.

- $|v_i \oplus j - 2v_{i-1} \oplus k + v_{i-2} \oplus m|^2$ is the curvature of active contour at point $i$ estimated using the three points; $v_{i-2}, v_{i-1}$ and $v_i$.

In order to obtain a more accurate active contour model, we redefine the total energy of active contours. Then when expressed using same notation as [2], we have

$$E_t(i, j, k) = \min_{0 \leq m \leq N} \{E_t(i - 1, k, m) + E_{ext}(v_i \oplus k) + \alpha|v_i \oplus k - v_{i-1+n} \oplus m|^2 + \beta|v_{i+1} \oplus j - 2v_i \oplus k + v_{i-1+n} \oplus m|^2\} \quad (4.4)$$

This new active contour model is named the ICEAC model. For an explanation of Eq.4.4, the differences in comparison to Eq.4.3 are as follows:

- $v_{i-1+n}$ illustrates the previous point of point $i$ on the active contour. For example the previous point for starting point is the end point due to active contour being closed.

- $|v_i \oplus k - v_{i-1+n} \oplus j|^2$ means the distance between the $k^{th}$ neighbour of point $v_i$ and the $j^{th}$ neighbour of its previous point.

- $E_{ext}(v_i \oplus k)$ shows the distance of the $k^{th}$ neighbour of point $v_i$ on active contour from the nearest image edge (external energy).

- $|v_{i+1} \oplus j - 2v_i \oplus k + v_{i-1+n} \oplus m|^2$ is the curvature of active contour at point $i$ estimated using the three points; $v_{i-1}, v_i$ and $v_{i+1}$.
The outline of the ICEAC method is as follows: First, the elements of energy matrix are computed using Eq. 4.4. Second, 2D location matrix with dimension $i \times k$ is computed as well. Third, the minimum energy of active contour in first iteration is found. This minimum energy is the minimum entry of energy matrix in last row, $E_{\text{min}}(t) = \min_k E_t(n - 1, k)$. Then the column of this minimum energy shows the number of neighbour that the end point on active contour should move towards on new active contour. Finally the entry of location matrix in the end row with the same column of minimum energy shows the number of neighbour that the point $v_{n-2}$ on active contour should move towards on new active contour. This backward process continues until the new position of all points on the new active contour will be achieved. If the new active contour does not reach the underlying image edges, other iterations start with the same procedure as first iteration on their new active contours. In ICEAC model setting parameters is very simple and does not need any repetition when considering the following guidelines:

- For single object images, $\alpha = \beta = 1$.

- For multiple objects images; first setting $\alpha$ to high values, $10 \leq \alpha \leq 20$ without changing $\beta$ and in special cases if changing $\beta$ becomes necessary, only low values of $\beta$, $2 \leq \beta \leq 5$ are sufficient.

- For complex images setting $\alpha$ to higher values, $20 \leq \alpha \leq 50$ is sufficient.
4.3 Fast Active Contour Convergence through CSS Filter

As an example, it has been shown in Fig.4.2 that the AMI model computes the incorrect value for curvature at point \( v_i \) based on Eq.4.3 using its two previous points. Note that point \( v_i \) is a maximum of curvature on curved line ABC while the value of curvature at point \( v_i \) using its two previous points \( v_{i-2} \) and \( v_{i-1} \) becomes zero as these three points are placed on a straight line AB. However the ICEAC model computes the curvature of point \( v_i \) correctly using two neighbours \( v_{i-1} \) and \( v_{i+1} \) which confirms that the value of curvature is high at \( v_i \).

4.3 Fast Active Contour Convergence through CSS Filter

The second part of internal energy in the AMI model is the curvature of the active contour. Only by removing this part from Eq.4.3, this formula summarises to Eq.4.5 as follows:

\[
E_t(i, k) = \min_{0 \leq j \leq N} \left\{ E_t(i - 1, j) + \alpha |v_i \oplus k - v_{(i-1+n)} \oplus j|^2 + E_{ext}(v_i \oplus k) \right\}
\]  

(4.5)

Then the total energy of active contour in this formula can be computed through a single 2D matrix instead of a 3D matrix at any point of active contour. On other hand, without the effect of the curvature part in internal energy of active contour, final snake loses its smoothness and does not behave like a membrane or thin plate. Then the final snake can not be used in image matching and tracking tasks to find interest objects. As a result, the existence of this part in total energy of active contour is necessary. However, an alternative idea is to replace this part with another step that ensures smoothness but converges faster.

The SAC model [58] is based on reformulating internal energy in Eq.4.3 by removing the curvature part of this formula and using Gaussian filtering with low scale of smoothing. By applying the CSS technique for smoothing, we smooth the output of Eq.4.5 in each iteration until locking on to the underlying image edges occurs at least at one point. The outline of the SAC method has been illustrated in Fig.4.3. Our reformulated internal energy can be seen in Eq.4.5. For an explanation of this equation, the differences in comparison to Eq.4.3 are as follows:
• $E_t(i, k)$ denotes the total energy of active contour at iteration $t$. Energy values are stored in a 2D matrix. Each row of this matrix belongs to one point of active contour.

• $v_{i-1+n}$ illustrates the previous point of point $v_i$ on the active contour. For example the previous point for starting point is the end point due to active contour being closed.

• $|v_i \oplus k - v_{i-1+n} \oplus j|^2$ means the distance between the $k^{th}$ neighbour of point $v_i$ and the $j^{th}$ neighbour of its previous point.

• $E_{ext}(v_i \oplus k)$ shows the distance of the $k^{th}$ neighbour of point $v_i$ on active contour from the nearest image edge (external energy).

• $E_t(i - 1, j)$ represents an element of energy matrix in iteration $t$ at row $i-1$ and column $j$. Actually it is the total energy of active contour at previous point of $v_i$ in column $j$.

The outline of the SAC method is as follows: First, the elements of energy matrix are computed using Eq.4.5. Second, 2D location matrix with dimension $ixk$ is computed as well. Third, the minimum energy of active contour in first iteration is found. This minimum energy is the minimum entry of energy matrix in last row, $E_{min}(t) = \min_k E_t(n - 1, k)$. Then the column of this minimum energy shows the number of neighbour that the end point on active contour should move towards on new active contour. Finally the entry of location matrix in the end row with the same column of minimum energy shows the number of neighbour that the point $v_{n-2}$ on active contour should move towards on new active contour. This backward process continues until the new position of all points on the new active contour will be achieved. Then the output of this iteration is smoothed by a low scale Gaussian filter. If the new active contour does not reach the underlying image edges, other iterations start with the same procedure as first iteration on their new active contours. The process continues until there will be no change in minimum energy of active contour between two successive iterations. Due to using Gaussian filtering, setting $\sigma$ to low value is emphasised. High value of $\sigma$ may smooth the initial snake so much that it cuts the interest object at some
4.3. Fast Active Contour Convergence through CSS Filter

---

Figure 4.3: The SAC model scheme for active contours

(a) Initial snake  
(b) Final iter., $\sigma=1$  
(c) Final iter., $\sigma=20$

Figure 4.4: The effect of smoothing at high scale in the SAC model
parts of its boundary and in worst case goes inside the object, see Fig.4.4. In the AMI model by using Eq.4.3, if there are \( n \) points on active contour and \( m \) possible directions at each point, the time complexity for each iteration is \( O(nm^3) \), but in the SAC model if we consider \( m \) additions and \( m \) multiplications for convolving Gaussian function by the output of each iteration, the time complexity for each iteration is \( O(nm^2) + O(nm) \); while \( O(nm) \) does not exist for some iterations. The comparison of these two formulas shows that the SAC model is faster. Furthermore, it does not lose smoothness in final snake shape. It follows that the SAC model is a good user interface for image/video database retrieval system.

4.4 An Adaptive Smoothed Active Contour Model

In this section, we introduce another accurate and quick algorithm for minimising the energy of active contour models based on Eq.4.5 combined with adaptive CSS filtering for smoothing. The proposed active contour model, ASAC is based on reformulating the internal energy of an active contour by removing its curvature part and using an adaptive CSS filtering with an adaptive scale, \( \sigma \) for smoothing. The ASAC stands for adaptive smoothed active contour. By applying adaptive CSS filtering, we smooth the input contour of the ASAC algorithm (which is the user initial snake or the output of each iteration of the energy minimising process) using the Gaussian function with high scale until input contour reaches the underlying image edges. Then we reduce \( \sigma \) by one and repeat the smoothing process until the smoothed contour locks on to the object of interest or locking on to the underlying image edges occurs. Reduction of \( \sigma \) after each locking on to the underlying image edges continues until \( \sigma = 0 \). After that the remainder of iterations continue without smoothing. The process halts when there is no change in the minimum energy of the active contour between two successive iterations. The outline of the ASAC model has been illustrated in Fig.4.5. The reformulated internal energy of the ASAC model can be seen in Eq.4.5. For an explanation of this equation, the differences in comparison to Eq.4.3 have been presented in Sec.4.3. The outline of the ASAC method is as follows:

First, the input contour (\( \text{Gin} \)) is smoothed by CSS filtering using high value of \( \sigma \), and if the smoothed contour does not reach to the underlying image edges, the energy
4.4. An Adaptive Smoothed Active Contour Model

minimising process will be started. In the energy minimising process, the elements of
energy matrix are computed using Eq.4.5. Second, 2D location matrix with dimension
i×k is computed as well. Third, the minimum energy of active contour in the first
iteration is found. This minimum energy is the minimum entry of the energy matrix
in last row. Then the column of this minimum energy shows the number of neighbour
that the end point on the active contour should move towards on new active contour.
Finally the entry of the location matrix in the end row with the same column of the
minimum energy shows the number of neighbour that the point $n-2$ on an active contour
should move towards on new active contour. This backward process continues until the
new position of all points on the new active contour will be achieved. Then the output
of this iteration is smoothed by the CSS filtering with same $\sigma$, if no connection to the
underlying image edges occurs. Otherwise the $\sigma$ is decreased by one and the input
contour before the last smoothing process will be smoothed with lower $\sigma$. Reduction

Figure 4.5: The ASAC model scheme
of $\sigma$ continues until $\sigma=0$. The process halts when there is no change in the minimum energy of the active contour between two successive iterations. The ASAC model has only one parameter; $\alpha$, that affects the internal energy of the active contour. Setting this parameter to higher value gives a higher weight to internal energy.
Chapter 5

Robust Corner Tracking for Multimedia Applications

5.1 Introduction

Feature tracking especially corner tracking is an important issue in computer vision and its applications. Our interest in corner tracking is for video database retrieval and multimedia applications so it should be robust, efficient and practical. Feature tracking algorithms which make some important assumptions for tracking, are successful in tracking feature points as long as those assumptions are satisfied. Feature tracking algorithms which use special motion models for tracking feature points and finding correspondences are not practical for multimedia applications. This is because there is no guarantee that all video sequences have the same or even close motion to their assumed motion model for tracking. Therefore these groups of feature trackers are not robust and actually fail to retrieve the query when unconstrained motions occur in video sequences. To tackle unconstrained motions, we need a feature tracking algorithm which does not make any limiting assumptions and does not use any restrictive motion models. The underlying principles of a robust algorithm should be general and useful not only for video sequences including perspective distortions and strong intensity changes but also for video sequences including unconstrained motions. As a result, such an algorithm can be utilised for video database retrieval generally and efficiently.
On other hand real-time corner tracking needs robust corners which can be identified and tracked from frame to frame in video sequences. No corner tracker can work for multimedia applications unless good matches can be identified between two successive frames. Tracked corners must be located in each frame using the information about the corners in the previous frames. However due to projective, affine and rotational transformations between two successive frames, even the position of good features can vary so far that no correspondence can be found within the match window. While enlarging the match window not only can not solve the problem but also incorrect correspondences will be found which will result in the corner tracker losing its target. Therefore the best solution for the mentioned problem is that the number of corners and also their positions among frames in image sequences become robust. In other words, the number of corners in successive frames should be constant and their positions should change slightly, if at all. Furthermore, as occlusion happens, missing corners should be tracked again after disocclusion. Otherwise the tracker loses its target.

In this chapter, first the shortcomings of the Tommasini tracker have been discussed in Sec.5.2. Afterwards, the multiple-match corner tracker is explained in section 5.3. Furthermore in this chapter, traditional two-frame matching using single matching (5.4.1) is compared to our proposed two-frame matching using multiple matching (5.4.2) combined with three-frame (Sec.5.5) monitoring. In section 5.6, in the two-frame matching stage of the multiple-match tracker, not only the corners of the first frame but also all the corners that appear fresh in each frame of input sequence are considered. This process which has been explained in 5.6.1, gives this possibility to the user to specify an object of interest as query in any frames in the input video sequence.

5.2 Tommasini Tracker Shortcomings

In Tommasini et al. algorithm a feature is defined as a region that can be tracked easily from one frame to the other. Then feature is not always a corner. Also in their framework, a feature can be tracked reliably if they can find a numerically stable solution to the equation of their assumed motion model. They did not track single features, but windows of features, and they looked for windows that contain sufficient
texture. They gave a definition of what sufficient texture is for reliable feature tracking. Unfortunately, different points within a window may behave differently. The points are moving at different velocities and may disappear or appear a new. Therefore how do they determine that they are following the same window, if its contents change over time? Second, if they measure the displacement of the window, how are the different velocities combined to give the one resulting vector?

Their solution to the first problem was residue monitoring. They kept checking that the appearance of a window has not changed too much. If it has, they discarded the window. For second problem rather than describing window changes as simple translations, they modelled the changes as a more complex translation, such as affine map. Selecting affine motion model will cause another problem, the size of the match window. This is because more parameters to estimate require the use of larger windows to constrain the parameters sufficiently. On the other hand windows should be as small as possible, combined with good noise rejection and the size of the window can improve performance considerably. Over all, both of their solutions are not practical for unconstrained motions in video sequences. If unconstrained motion occur, none of these assumptions remain satisfied. Then not only the match window will be discarded but also all the tracked features in and around the location of unconstrained motion occurrence will be rejected based on their rejection rule; X84, since they assume a Gaussian distribution of the residual for the good features. Therefore in unconstrained motion the residuals are not samples from the Gaussian distribution of good features. Furthermore in their algorithm, occlusion problem has not been addressed.

5.3 Multiple-match Corner Tracker

In this section a multiple-match corner tracking algorithm with all properties that were discussed in Sec.5.1 for real-time feature tracking used in video database retrieval is proposed. The proposed algorithm is based on the multi-scale corner detector and two-frame matching using multiple matches combined with three-frame based monitoring. To extract corners from each frame of video sequences, the multi-scale corner detector (refer to Chap.3) is applied. In the matching stage a new two-frame correspondence
algorithm using multiple matches is employed. Considering multiple hypotheses helps to ensure that if the corner detector is not completely robust, we still have a chance to find as close as possible a match or multiple match hypotheses for the corners. On the other hand, by considering all the match candidates, many false matches will join the tracked corners. False matches are removed using a distance criterion based on a reference point. Problem of recovering points lost during tracking is solved considering three-frame based monitoring. We monitor tracked corners from the third frame of input sequence due to occlusion or sudden change in drift of the tracker. As a result, the proposed three-frame monitoring [55] helps the new tracked corners to be added to list of tracked corners in input video sequence. Therefore the three-frame monitoring helps to ensure that the number of tracked corners and their tracked positions among frames become more robust. Experiments have been carried out on real video sequences depicting translation, scaling, rotation and affine transformation including non-smooth motions with different lighting and different camera motions. Overall, by applying two-frame multiple matching combined with three-frame monitoring the number of tracked corners among frames stays more robust. In the remainder of this section, the two stages of the multiple-match corner tracking algorithm including the corner matching, and monitoring have been considered in detail one by one. The first stage of this tracker; the multi-scale corner detector has been explained in Chap.3.

5.4 Two-frame Corner Matching

Generally corner matching is commonly referred to as the correspondence problem. The problem is how to automatically match corresponding corners from two images, while no incorrect matches are assigned at the same time.

5.4.1 Single Matching Process

Therefore in continue, after extracting the corners of all frames in the off-line stage of our tracker, two-frame corner matching will be started. In this stage, the matcher unit first receives the corners of frame0 and frame1. Then for every corner in frame0,
we construct a small window in frame1 centred at the same position of that corner in frame0. Next all the corners of frame1 lying in this window are match candidates for that corner in frame0. In traditional two-frame matching algorithms always only one match was considered for each corner using different similarity functions [53]. For example if the standard cross-correlation (SCC) was used as similarity metric, match with highest score of the SCC was considered as the best match. However it has been discovered through many experiments that considering only one match is not a good solution, no matter which similarity metric has been used or which definition has been used for the best match. By selecting one match as the best match, the tracker often can not find any match for the tracked corners in a long time span. Our selection among different similarity functions are standard cross-correlation (SCC), zero-mean cross-correlation (ZM-CC), sum of squared differences (SSD), and \( \chi^2 \)-test which have been defined in appendix A. We applied these four similarity functions to the bream sequence and also to many sequences of CVSSP\(^1\) and CMU/VASC\(^2\) video databases. Overall, we found that the SSD, \( \chi^2 \)-test and SCC perform better than the ZM-CC and also the SSD and \( \chi^2 \)-test perform better than the SCC in tested video databases. In Fig.5.1, the results of matching process itself using these similarity functions in some frames of the bream sequence have been illustrated. As it can be seen from these results single matching even using the best similarity function, sometimes can not assign a correct match, or even any match between corners of two successive frames in output of the multi-scale corner detector. In this way, in long video sequences, it may happen that no match corner remains at final frame. The remedy to this problem are considered as two-frame matching using multiple matching combined with three-frame monitoring.

5.4.2 Multiple Matching Process

Therefore as remedy to mentioned problem in Sec.5.4.1, in two-frame matching stage of the proposed tracker, all the match candidates are considered as multiple hypotheses. On the other hand, by applying two-frame multiple matching algorithm to frames of an

\(^1\)http://vssp-www.ee.surrey.ac.uk/data/video/index.html
Figure 5.1: Results of matching process using single matches on some randomly selected frames in the bream sequence using different similarity functions. In first row; from left to right, the results of matching process using SCC similarity function on frames number 6, 11, 14, 15, and 16 are illustrated. In second row to fourth, the same have been carried out for the same frames using ZM-CC, SSD, and $\chi^2$-test similarity functions respectively.
5.4. Two-frame Corner Matching

Consider only $c_1$, $c_3$, and $c_4$ among the corners in first frame of an input video sequence. Also we create for each corner in this frame a family. Corners $c_1$, $c_3$, and $c_4$ have three, two, and one matches respectively. Included three matches of $m_1$, $m_2$, and $m_3$ are multiple matches of $c_1$ from previous frame.

In the last frame, $F_1$ has many members for $c_1$ using multiple matching. But most of them are really mismatches such as $c^*$ and they should be removed.

In fig. 5.2, it can be seen, for instance, corner $c_1$ in the last frame has match $c^*$ which is really a mismatch and too far away. Then we need a criterion [60] to remove all false matches that will appear among the tracked corners. As a solution, for each corner of the first frame, a family is created. These families include all the matches that can be found for the corners of the first frame while tracking through other frames. Then if the first frame has 5 corners, we have a total of 5 families (refer to fig. 5.2). In the start, each family has one member. These members are corners of first frame itself. After each two-frame matching, new members join to the families. For example, in fig. 5.2, $F_1$; the family of corner $c_1$ has 1, 3, 6, and many members in frame 0, 1, 2, and the last frame respectively. Also among the members of each family the match candidate with highest matching score using standard cross-correlation metric is considered as a reference point for that family (such as $m_0$ for $F_1$ in Fig. 5.3). Afterwards in each family, the distances of all the members to the reference point of that family are computed (refer to Fig. 5.3). Then all the members which have distances more than a threshold to their reference points will be removed from the families and actually from the tracked corners.

Note that after continuing the multiple matching and distance criterion to the end of a video sequence, in the last frame, there may be more than one match for each tracked corner. Therefore in the last frame, only a match with highest score of cross-correlation
Chapter 5. Robust Corner Tracking for Multimedia Applications

Consider only two corners and their families in this frame.

After applying distance criterion, members of family F1 which have distances more than a threshold are removed. Therefore only 2 members remain for F1 and similarly for F2, 4 members remain.

Two-frame correspondence corner C1 has 5 matches and C2 has 7 matches. m0 in each family is the match with the highest score of SCC metric. Then m0 is a reference point for F1 and similarly for F2.

After two-frame correspondence, F1, and F2 have 5, and 7 members respectively. Two-frame multiple matching plus distance criterion will continue to the end of tracking.

However, in the last frame only one match remain for each corner in first frame. This match is the match with highest score of SCC metric among the correspondence.

Figure 5.3: An example of the reference points in the families which are created for the corners of frame0 in an example input sequence and the distances of other members to these points. In this figure SCC stands for standard cross-correlation metric.

As a result of using the described distance criterion, we can have the closest and the minimum number of the matches to continue tracking without making any assumptions on motion models and without using any information on camera movement or camera calibration.

5.5 Corner Monitoring Process

A remaining issue is corners that multiple-matching algorithm can not find any match for. We solved this problem by monitoring tracked corners when applying two-frame
5.5. Corner Monitoring Process

multiple matching to the other frames. This is another difference between our two-frame and traditional two-frame matching algorithms. But before explaining the monitoring process, first we describe how the output frames of the two-frame multiple matching are produced in this algorithm.

As we mentioned in Sec.5.4.1, after extracting the corners of all frames of input sequence in off-line stage of the proposed tracker, two-frame corner matching will be started. If we referred to output frames of the multi-scale corner detector as the outputO, output1,..., then first all the corners of the output0 are marked in frame0 of input sequence as the first output of the matcher. Afterwards, the matched corners between output0 and output1 are marked in frame1 of input sequence. Therefore the output1 is the second output of the matcher unit. Consequently, the matched corners between frame1 and output2, which are marked in frame2, is the third output of the matcher unit. This procedure will be continued for all output frames of the matcher unit (Fig.5.4). Therefore if we wanted to compute the translation coordinates of each corner between two successive output frames of the matcher; it is possible by computing the differences between the \textit{x-coordinate} of a corner in an output of the matcher and the \textit{x-coordinate} of a corner in its previous output where those corners were matched. This difference is named \textit{x-translation} of that corner in current output of the matcher. The \textit{y-translation} of each corner in an output of the matcher can be computed in a similar way.

We monitor the matched corners among three successive output images of the matcher as following: consider frame0, frame1, and frame2 of input sequences as three first frames for starting the monitoring process. For each matched corner through multiple-matching stage, we set a flag to 1. The corners in frame0 also have set flags. Furthermore, we have \textit{x} and \textit{y} translations of the corners between each two frames. Then monitoring process will start to mark unmatched corners from frame2 (refer to Fig.5.4). Each unmatched corner in frame1 (that we can not find a match for between the corners of frame2) will be marked in frame2 with zero assigned flag. Zero flag for each corner means that it was marked during monitoring process. Therefore it is not a real match that is derived from the multiple-matching process. Through multiple-matching process between frame2 and frame3, each unmatched corner on frame2 will be marked...
Chapter 5. Robust Corner Tracking for Multimedia Applications

Figure 5.4: Scheme for the monitoring process
5.6. Corner Tracking including New Corners

In this section, in the two-frame multiple matching stage of the proposed tracker, not only the corners of the first frame but also all the corners that appear fresh in each frame of the video sequence are considered. This process which has been explained in 5.6.1, gives this possibility to the user to specify an object of interest as query in any frames in the input sequence. Therefore in proposed corner tracker, in order to extract corners from each frame of video sequences, the multi-scale corner detector (refer to Chap.3) is applied. In the matching stage two-frame correspondence using multiple match hypotheses (refer to 5.4.2) is considered. Therefore if we intend to track the moving objects through a video sequence from any selected frames as the start frame, we have to track not only the corners of the first frame but also all the corners that appear fresh in each frame of the input video sequence. Accordingly by tracking the new corners, the corners of hidden moving objects will appear again in the list of tracked corners among frames of the tested sequence after appearance of those
objects in next frames. Since after each frame some new corners come up on the list of tracked corners, each corner needs to have a record. The record for each tracked corner includes the frame number when tracking started and the frame number when tracking ended. Therefore the paths of tracked corners are available using their record. In the following the process of two-frame multiple matching using not only the corners of first frame but also all the corners that appear for the first time in each frame is presented.

5.6.1 Multiple Matching including New Corners

The matcher first receives the corners of frame0 and frame1 from the output of the multi-scale corner detector. For every corner in frame0, we construct a window in frame1 centred at the same position. Then all the corners of frame1 lying in this window are match candidates for that corner in frame0. If more than one corner lie in this window, we consider all the match candidates of that corner as multiple hypotheses. Therefore all the corners of output0 of the multi-scale corner detector are marked in frame0 as the output0 of the matcher. For second output of the matcher; the matched corners between output0 and output1 of the multi-scale corner detector are marked in frame1 of input sequence as output1 of the matcher together with all the corners that appear fresh in output1 in comparison with the corners in output0. Then all the corners including matched corners between frame1 and output2 of the multi-scale corner detector and all the corners that appear fresh in output2 of the multi-scale corner detector are marked in frame2 of the input sequence as the output2 of the matcher. Note that after adding new corners to the list of the corners of a frame, all these corners are considered for multiple matching process of the next frame. Therefore each new added corner also will find one or multiple matches among the corners of its next frame. This procedure will be continued for creating all the output of the matcher [54].

However by applying this process until the last frame of input sequence, we will find many false matches that cause the tracker to lose its target. The problem of using multiple matches for tracking corners without any limitation has been illustrated in Fig.5.5. As it can be seen in this figure, for instance, corner cl finally has corner c* as one of its matches which is too far and is really a mismatch. It is similar for each
5.6. Corner Tracking including New Corners

Consider frame0 has only 4 corners and only two corners C1 and C4 are shown here. We create for each corner in frame0 a family. F1 and F2 have 1 member in frame0. After two-frame multiple matching, F1 have 3, and 1 members in frame1. Corner nc11, is new corner C1 which appears newly in frame1 and similarly for nc12. The two-frame multiple matching for nc11, and nc12 are similar as m1, m2, m3, and m4, which are the matches for corners in frame0. Also we consider a family for each new corner as well. Assigned number to each family always will start from the last number that was considered for the last corner in frame0. For instance, the last family in frame0 is F4. Therefore two corners in frame1 have families F5, and F6. As result, in each two-frame multiple matching not only the matches of corners in frame0, but also all the corners that appear newly in each frame will be considered. In last frame, we show only the correspondences for new corners, nc21 and corner C1. As it can be seen, family F1 has many members due to multiple matching process. But most of them are really mismatches such as c*. These mismatches should be removed. Note that in frame2, mn111 and mn112 are two matches of new corner nc11.

Figure 5.5: Multiple-matching scheme using the corners of first frame together with all the corners that appear newly in each frame of input sequence.

new added corner as well. Therefore we need a criterion to remove all falsely matches that will appear on the list of tracked corners. First we create a family for each corner of the first frame. These families have to include all the matches that we will find for these corners among the multi-scale corner detector output. Also after each frame, we create a family for each new added corner. These new families have to include all the matches that we will find for the new corners among the multi-scale corner detector output. The new families will be added after each frame to the end of the list of the families that were created in frame0. For example in frame2 of Fig.5.5, F1, F5 and F7 are the families of corner c1, nc11 and nc21 respectively. Family F1 was created in frame0. F5 is the family of new corner; nc11, which appeared firstly in frame1 and has two members in frame2. F7 is the family of new corner; nc21, which appears in frame2 newly. Furthermore, in Fig.5.5, F1 has only one member in frame0. After multiple matching stage between output0 and output1 of the multi-scale corner detector, F1 contains three matches; m1, m2 and m3. They are the new members of family F1 in
Chapter 5. Robust Corner Tracking for Multimedia Applications

Consider only two corners and their families in frame0.

After two-frame correspondence corner C1 has 5 matches and C2 has 7 matches. m0 in both families, F1, and F2 is a reference point. Corner ncl is a new corner in this frame.

After applying distance criterion, members of family F1 which have distances more than a threshold are removed. Note that the two-frame multiple matching and distance criterion process for new comer ncl will be started from the next frame.

After two-frame correspondence corner C1 has 5 matches and C2 has 7 matches. m0 in both families, F1, and F2 is a reference point. Corner ncl is a new corner in this frame.

After two-frame correspondence corner C1 has 5 matches and C2 has 7 matches. m0 in both families, F1, and F2 is a reference point. Corner ncl is a new corner in this frame.

After two-frame correspondence corner C1 has 5 matches and C2 has 7 matches. m0 in both families, F1, and F2 is a reference point. Corner ncl is a new corner in this frame.

After applying distance criterion, members of family F1 which have distances more than a threshold are removed. Note that the two-frame multiple matching and distance criterion process for new comer ncl will be started from the next frame.

In this frame, after two-frame multiple matching again, F1 and F2 have 5, and 7 members. New comer ncl has also 3 matches in its family, F3. m0 is a reference point of family F3. The distances of other members to m0 have been shown in this frame.

After applying distance criterion, families F1, F2, and F3 have 3, 4, and 2 members. Therefore the two-frame multiple matching plus distance criterion process are the same for new families and families which are created for the first time in frame0.

Figure 5.6: An example of the reference points in the families which are created for the corners of frame0 and all the new corners in an input sequence and the distances of other members to these points. In this figure SCC stands for standard cross-correlation metric.

The members of F1 are increased to six after multiple matching stage between frame1 and output2 of the multi-scale corner detector as marked in frame2 of Fig.5.5. Therefore after each two-frame correspondence stage, we add new members to the families which have been created in the first frame of the test sequence. We also find among the members of each family the match candidate with the highest matching score using standard cross-correlation metric as a reference point for that family. Then after each two-frame correspondence, we compute the distances of each member of these families to their reference points, no matter they are new families that are created in each frame (such as family F3 in Fig.5.6) or the families that were created in frame0.
(such as family F1 in Fig.5.6). Afterwards, all the members of each family which have distances more than a threshold will be removed from their families. Note that the value of threshold is constant for all video sequences and does not need any setting. As a result, the multiple matches which have been found using the proposed two-frame multiple matching process are as much as possible the closest and the minimum number of the matches that we need to have for continuing tracking correctly without losing the target and more independently of the corner detector. In Fig.5.6, the process of proposed two-frame multiple matching and distance criterion have been illustrated for three families, F1, F2, and F3 in frame0. As it can be seen, these processes are the same for new families and families which create for the first time in frame0.
Chapter 6

Content-based Video Retrieval through Robust Corner Tracking

6.1 Introduction

So far the required image processing tools that we need for constructing the proposed video retrieval system have been prepared in previous chapters. Therefore in this chapter, a novel content-based video retrieval system [59] using frames corners is proposed. To provide indices, first all the shots in a video sequence are extracted. Then corners of all frames in each video shot are detected applying the multi-scale corner detector. As a user interface, the SAC active contour model is employed to specify one object of interest as a query in one of the frames of each video shot. This frame which is selected by user is referred to as the selected-frame. The closest corners of query object to the final snake in the selected-frame are extracted and tracked forwardly and backwardly through the whole of that shot using the multiple-match corner tracker. The multiple-match corner tracker, which does not make any important assumptions or use any motion models, can retrieve the query in any video sequence even when there is non-smooth and unconstrained motion. By tracking the corners of query object forwardly and backwardly, the positions of similar objects in each video frame are determined. Two methods are considered for demonstrating the query and its similar objects to the user. In this chapter, by selecting hotel sequence including affine transformation, we
Figure 6.1: Output images of applying the proposed content-based retrieval system to the hotel sequence after second stage. Second stage is user query specifying (a) using the SAC active contour model. In this figure two randomly selected of output iterations of the SAC active contour model and its final snake can be seen in (b), (c) and (d) respectively. Also in this figure Itr. stands for iteration. Then Itr.2 means iteration number 2.

6.2 User Query Specification

After pre-processing stage (including shots extraction and corners detection), user can select one of the frames in video shot (selected-frame) for specifying an object of interest as a query. Then in the second stage of our retrieval system, user specifies one object of interest as query on the selected-frame using the SAC active contour model. Afterwards, the user snake will lock on the query very accurately and quickly. In Fig.6.1, the second stage of the proposed system has been shown. User has drawn an initial snake as close as possible around selected query in hotel19 (Fig.6.1(a)). Then by applying the SAC active contour model to the initial snake, this user snake has been locked on the query after twelve iterations. The final snake can be seen in Fig.6.1(d). Two of the middle iterations output snakes are shown in Fig.6.1(b) and (c).
6.3 Selecting Corners in Query Closest to the User Snake

After user specifies a query in the selected-frame using the SAC active contour model, only corners of the selected-frame closest to the final snake will be tracked in order to determine the positions of similar objects as query in each frame. This is another advantage of proposed retrieval system which can consider only small neighbourhoods of query corners for tracking to determine retrieved objects for query based on using only corners as features. Therefore first we need to select only corners in the selected-frame closest to the final snake. For this purpose, the distances of all the corners of the selected-frame to each pixel of the final snake are computed. Then only those corners which have distances less than five will be selected as the corners closest to the final snake. Therefore these closest corners are tracked in the next stage in order to determine retrieved objects for query. In Fig.6.2(a) and (c), all the corners of the selected-frame which were stored as meta-data at pre-processing stage and also corners closest to the final snake in this frame have been shown respectively. In Fig.6.2(b), the final snake which is the final iteration snake of the SAC model has been illustrated as well. Then it is completely visible that the corners in Fig.6.2(c) are the corners closest to the final snake among the whole of the underlying frame corners.
6.4 Tracking Selected Corners in Query

To determine similar objects as user query in the whole of video sequence, we need to track query selected corners in the selected-frame (hotel19) forwardly and/or backwardly in order to determine the positions of tracked corners in the whole of input sequence. Therefore two-frame multiple matching process (refer to 5.4.2) should be started between the selected-frame and its next/previous frame in this stage. Here we consider only forward tracking while backward tracking is similar by changing only next frame to previous frame in the following procedure. In forward tracking, the matcher unit first receives the selected corners from previous stage (closest corners to the final snake in the selected-frame) and all the corners of next frame to the selected-frame (hotel20). Then for every selected corner, multiple match candidates are determined. Afterwards for each selected corner in query, a family is created. These families include all the matches that can be found for selected corners in query while tracking through other frames. After each two-frame matching, new members join to the families. Also among the members of each family the match candidate with highest matching score using standard cross-correlation metric is considered as a reference point for that family. Then in each family, the distances of all the members to the reference point of that family are computed. Next all the members which have distances more than a threshold to their reference points will be removed from the families and actually from the tracked corners. As a result of using distance criterion, we can have the closest and the minimum number of the matches to continue tracking without making any assumptions on motion models and without using any information on camera movement or camera calibration. A remaining issue is corners that multiple-matching algorithm can not find any match for. We solved this problem by monitoring (refer to Sec.5.5) tracked corners when applying the multiple matching process to the other frames.

6.5 Demonstrating Identified Similar Objects as Query

After tracking only the selected corners in query through a video shot, identified similar objects as query should be localised and demonstrated in each frame. In proposed video
retrieval system two approaches are applied for demonstrating the results:

- Using a bounding box around identified similar object as query in each frame
- Using the least-squares estimation method in order to determine transformation parameters for transforming the final snake to the other frames in a video shot.

As result of this snake transformation, a similar snake as the final snake will be drawn in the place of identified similar object as query in each frame.

6.5.1 Bounding Box Method

In this method as in each frame of the input sequence the coordinates of tracked corners are available, we can draw a bounding box including all these corners in each frame. Then we can use the four points: (xmin,ymin), (xmin,ymax), (xmax,ymin), and (xmax,ymax) from the tracked corners to draw a bounding box around identified similar object as query in each frame. In Fig.6.3, some of the retrieved results for a user query (which has been demonstrated in Fig.6.3(a)) have been illustrated using bounding box method.

6.5.2 Least-Squares Estimation Method

Let \( \mathcal{P} = (P_x, P_y) \) be a set of selected corners in the selected-frame and \( \mathcal{E} = (E_x, E_y) \) be the set of corresponding points in next frame to the selected-frame. These corresponding points have been found through forward tracking in Sec.6.4. For each single point, the parameters of following transformation

\[
E_x = aP_x + bP_y + e \\
E_y = cP_x + dP_y + f
\]  

must be solved for. The least-squares estimation method is used to estimate values of \( a, b, c, d, e, \) and \( f \). Let the X-dissimilarity measure \( E_{\text{rrx}} \), which measure the difference between the x-coordinate of single corner and its correspondence and similar for Y-
dissimilarity measure $Err_y$ be defined by

\[ Err_x = (aP_x + bP_y + e - E_x) \]
\[ Err_y = (cP_x + dP_y + f - E_y) \] (6.2)

For computing the total of dissimilarity measure $Err$, all the selected corners in the query should be considered. Then square of this total $Err^2$ can be defined by

\[ \Sigma Err^2 = \Sigma (Err_x^2 + Err_y^2) \] (6.3)

Furthermore as $Err_x$ and $Err_y$ are independent, $Err^2$ can be defined by

\[ \Sigma Err^2 = \Sigma Err_x^2 + \Sigma Err_y^2 \] (6.4)

Therefore for minimizing $Err^2$, both $Err_x^2$ and $Err_y^2$ should be minimized. Let

\[ T_x = (a, b, c) \]
\[ T_y = (c, d, f) \] (6.5)

be two vectors defined by the transformation parameters. Then the solution of

\[ \frac{\partial \Sigma Err_x^2}{\partial T_x} = 0 \]
\[ \frac{\partial \Sigma Err_y^2}{\partial T_y} = 0 \] (6.6)

is the least-squares estimates of those parameters. To compute that estimate, determine the partial derivatives of $\Sigma Err_x^2$ with respect to each of $a, b, c$ and $\Sigma Err_y^2$ with respect to each of $c, d, f$ and set those partial derivatives to zero. The result is a linear system of six equations in six unknowns which can be solved to obtain estimates for $a, b, c, d, e,$ and $f$. After estimating transformation parameters between the selected-frame and its next frame (i.e. hotel19 and hotel20), as we have the x and y coordinates of the final snake pixels, then we can transform these pixels from the selected-frame to its next frame and allocate a new snake in the next frame using

\[ x_{snake,n}[j] = a.x_{snake}[j] + b.y_{snake}[j] + e \]
\[ y_{snake,n}[j] = c.x_{snake}[j] + d.y_{snake}[j] + f \] (6.7)
where \((x_{snak}, y_{snak})\) and \((x_{snak_n}, y_{snak_n})\) are \(x\) and \(y\) coordinates of each pixel of the current snake and the new snake respectively. In Eq.6.7, \(j\) must vary from 0 to \(n\), which \(n\) is the number of the pixels in the current snake, in order to obtain all the pixels coordinates of the new snake. By continuing this procedure (first determining transformation parameters between each two frames and then transforming the current snake to the next frame) for each two frames in a video shot, the place of transformed snake can be localised from current frame to the next frame. Therefore finally the place of all transformed snakes have been localised in each frame in input sequence.

The similar process can be used backwardly. In Fig.6.4, some of the identified similar objects as query (which has been demonstrated in Fig.6.4(a)) have been illustrated using the least-squares estimation method. Note that the standard least-square estimation method that was used in this section does not include a treatment of outliers.
Figure 6.3: Demonstrating some randomly selected results of applying proposed content-based retrieval system in the hotel sequence using bonding box method. Note that query was selected in hotel19, then tracking was continued forwardly and backwardly to find the identified similar objects as query in other frames of the hotel sequence. In each frame of this figure, retrieved objects were surrounded by a rectangle. Furthermore, in this figure Q. stands for query.
Figure 6.4: Demonstrating some randomly selected results of applying proposed content-based retrieval system in the hotel sequence using the least-squares estimation method. Note that query was selected in hotel19, then tracking forwardly and backwardly was continued to determine the similar objects as query in other frames. In each frame of this figure, identified similar objects as query were surrounded by a similar snake as user drawn snake. Furthermore in this figure Q. stands for query.
Chapter 7

Results, Discussion, and Performance Evaluation of Proposed System

7.1 Introduction

After explaining all the theories underlying the proposed retrieval system, the results, discussion, and performance evaluation of our proposed system through its stages are demonstrated in this chapter. This chapter is divided into four main sections; the corner detector results, discussion, and performance evaluation (Sec.7.2), the active contour results, discussion, and performance evaluation (Sec.7.3), the corner tracking results, discussion, and performance evaluation (Sec.7.4) and finally the content-based video retrieval system results and discussion (Sec.7.5). In 7.2.1, the ECSS corner detector results are compared to the results of our four test corner detectors (Plessey [30], Kitchen and Rosenfeld [40], SUSAN [80], Mokhtarian and Suomela [62]) which have been suggested through the survey in Sec.2.2. Performance evaluation of the ECSS corner detector in comparison to our four test corner detectors under similarity and affine transforms has been shown in section 7.2.2. The multi-scale corner detector results have been shown in some randomly selected frames in a number of video sequences in 7.2.3. Therefore through these results you will find out the multi-scale corner detector
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

has sufficient robust corners for tracking among frames in input sequence, even when transformations occur. As result this corner detector was selected as a corner detection tool for proposed retrieval system. In section 7.3, the ICEAC, the SAC and the ASAC active contour results in images with single and multiple objects have been illustrated in 7.3.1, 7.3.2, and 7.3.3 respectively. The ICEAC and the SAC active contours are also compared to the AMI active contour model individually and together in term of speed in this section. The multiple-match corner tracker results have been illustrated in 7.4.1. Furthermore in this section, traditional two-frame matching using single matching is compared to our proposed two-frame matching using multiple matching combined with three-frame based monitoring. Results of applying modified multiple-match tracker to the bream sequence which is an example of a sequence including occlusion have been shown in 7.4.2. These results also have been compared to the results of applying the multiple-match corner tracker to the bream sequence. Performance evaluation of the multiple-match corner tracker, which tracks selected corners in a user query to locate objects similar to the query in each frame of input sequence, is compared to the performance evaluation of the well-known Tommasini feature tracker in 7.4.3. We have tested the proposed retrieval system in a wide range of real video databases depicting translation, rotation, scaling, affine transformation, non-smooth motion and combination of them with different lighting and different camera motions. Some randomly selected of these results have been illustrated in section 7.5.

7.2 Corner Detector Results, Discussion, and Performance Evaluation

7.2.1 The ECSS Corner Detector Results and Discussion

To obtain the most stable and accurate corner detector, our four test detectors; the original CSS [62], Kitchen and Rosenfeld [40] (K&R), Susan [80] and Plessey [30] were implemented and tested on several real images. We attempted to obtain the best possible results for each corner detector by searching for parameter values that appeared to yield the best results. Their results do not satisfy our criteria for being robust
corner detectors and show a lot of false corners. Furthermore these algorithms do not
demonstrate high performance under similarity and affine transforms. The results of
their performance evaluations will be shown in 7.2.2. Among these detectors, the CSS
corner detector had the best performance. Then we focused on this corner detector
and tried to determine its shortcomings and remove them. As result, the ECSS corner
detector which is an improvement of the CSS corner detector was proposed in section
3.5 with better performance on blunt and rounded corners. The ECSS corner detector
was tested on several different images. Only four of them have been illustrated in this
section. The results are compared with the outputs of our four test corner detectors.
The first test image, Fig.7.1 is a real image of airplane. The second test image, Fig.7.2
is another view of first image for demonstrating the robustness of the ECSS. In these
two images we have many blunt corners that Orig-CSS, the best one between our tested
corner detectors, has difficulty finding their correct positions. Another problem of the
Orig-CSS that can be seen in these two images is detection of many false corners. The
performance of the ECSS in comparison to these four corner detectors is best. The
third test image is house image. This image has many small details and texture in
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

(a) Plessey  (b) K&R  (c) Susan  
(d) Orig-CSS  (e) ECSS

Figure 7.2: Another view of airplane image

the brick wall and it was a difficult task for all the detectors as the details are very varied. Again the ECSS corner detector gave the best results amongst the four. The results are seen in Fig.7.3. Finally an image of blocks with many sharp corners which are detected using both the Orig-CSS and the ECSS corner detectors. Fig.7.4 shows the results. These examples show that the ECSS corner detector, especially for blunt corners, performs better than the other methods and that it is robust to image noise, whereas for sharp corners it performs as well as the Orig-CSS and much better than the others.

7.2.2 Performance Evaluation of the ECSS Corner Detector

A corner detector can be successfully used for matching, tracking and motion estimation if it has good stability and accuracy. However the majority of authors of published corner detectors have not used theoretical criteria to measure the stability and accuracy of their algorithms. They usually only illustrate their results on different test images and compare them to the results of other test corner detectors. A few of them have used
only one criterion. This criterion is the number of matched corners between original and transformed images, divided by the number of corners in the original image. This criterion is flawed since it favours algorithms which find more false corners in input images. Therefore as there is no standard procedure to measure stability and accuracy of corner detectors, we propose two criteria to evaluate the performance of corner detectors. We then carried out a number of experiments to compare the stability and accuracy of the ECSS corner detector to our four tested detectors. Therefore in this section first an overview of previous criteria for measuring stability of corner detectors has been described. Then our experiments for measuring these criteria are presented. Next, theory underlying our criteria is explained. Finally the results of experiments to determine stability and accuracy of the ECSS corner detector in comparison to four test detectors are illustrated. Overall, these results show that the ECSS corner detector has the best accuracy and stability among the five corner detectors.
Previous Criteria of Stability for Corner Detection

The majority of published corner detectors have not used properly defined criteria for measuring the stability and accuracy of their corner detectors. They have only demonstrated their results on different images in comparison to other test corner detectors. Some published results on corner detection include studies on the effects of noise and parameter variation on results of their corner detectors. These parameters include Gaussian scale $\sigma \{[101], [80], [74], [69]\}$, $\sigma$ white noise [101], thresholds $\{[85], [81]\}$, signal to noise ratio [12], cross-correlation matching [85], cost function [90] and the width of the gray level transitions in original image [73] but no definition of stability and its results. A few of them have used only one criterion to measure the stability of their corner detectors as follows:

Trajkovic and Hedley [85] used a measure of $k = \frac{N_m}{N_c}$, where $N_m$ and $N_c$ denoted number of strong matches and number of corners in the original image respectively. In terms of stability, a corner detector was better if $k$ is higher. Schmid and Mohr [74], applied the criterion of the ratio of total matches to the number of points extracted. This ratio
Figure 7.5: Airplane image under similarity and affine transforms. In this figure $s$, $xs$, $ys$ and $\theta$ stand for uniform scaling, $x$-scale and $y$-scale in non-uniform scaling and rotation parameters respectively.

varies depending on the image as well as on the type of transformation between the images. The problem of both criteria is that if we have an algorithm which marked all of the pixels in one image as corners then $k$ would become 100%. In other words algorithms with more false corners tend to have a larger number of matched corners. Therefore this criterion is flawed for measuring the stability of corner detectors. Our criteria are Consistency of corner numbers and accuracy. Only with consideration of these criteria together, we can judge correctly on the best corner detectors for tracking and matching tasks.

**Experiments**

We considered a test image as our original image. Then these experiments were performed as follows:

**Experiment 1:** In the first experiment, the number and the corners positions in original image were extracted using tested corner detectors. Next, original image was
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Rotated with rotation angle in range of $-90^\circ$ to $+90^\circ$. Then the number and the corners positions in all rotated images were extracted using tested detectors.

**Experiment 2:** In the second experiment, we did the same for original image and uniform scaling of this image with ten scale factors from 0.5 to 1.5.

**Experiment 3:** We repeated the same in the third experiment with non-uniform scaling. The variation of x-scale and y-scale in this transform were from 0.8 to 1.0 and 0.5 to 1.5 respectively.

**Experiment 4:** Affine transform was our fourth experiment that applied a rotation angle of $-10^\circ$ and $+10^\circ$ combined with x-scale from 0.8 to 1.0 and y-scale from 0.5 to 1.5.

Final test was performed for computation of accuracy. We propose a new approach for creation of ground-truth in this case. After performing the experiments, definition of appropriate criteria for comparing the results was very important. Appropriate criteria means criteria that show exactly which algorithm is the best for matching and tracking tasks taking into consideration all conditions especially extreme ones. Our new criteria have been explained in the following. These criteria were applied to our corner detector and four other test corner detectors. The results of stability and accuracy of these corner detectors have been illustrated in the end of this section.

**New Criteria for Performance Evaluation**

In this section our criteria for evaluating the stability and accuracy of corner detectors are defined theoretically [51]. In the following, let $N_o$ be the number of corners in original image (note that $N_o \neq 0$), $N_m$ number of matched corners in each of transformed images when compared to the original image corners and $N_t$ number of the corners in
7.2. Corner Detector Results, Discussion, and Performance Evaluation

Figure 7.7: Consistency of corner numbers for uniform scaling

each of the transformed images.

Consistency

Consistency means corner locations and numbers should be insensitive to the combination of noise, rotation, uniform or non-uniform scaling and affine transform. More importantly, corner locations and numbers should not move when multiple images are acquired of the same scene. Previous criterion of the consistency has been defined as follows:

\[
Consistency = \frac{N_m}{N_o}
\]

(7.1)

By this definition, algorithms which find more false corners in input images are favoured since they have higher number of the matched corners. Therefore we replace this criterion by two new criteria, consistency of corner numbers and accuracy. We define the criterion of consistency of corner numbers as follows:

\[
CCN = 100 \times 1.1^{-|N_t-N_o|}
\]

(7.2)

where CCN stands for “consistency of corner numbers”. Since stable corner detectors do not change the corner numbers from original image to transformed images then in terms of consistency, the value of CCN for stable corner detectors should be close to 100%. Any differences between the number of corners in the original image \((N_o)\) and the number of corners in transformed images \((N_t)\), causes CCN to drop below 100% as \(|N_t - N_o|\) grows larger. CCN is close to zero for corner detectors with many false corners.
Accuracy

Accuracy means corners should be detected as close as possible to the correct position. In one image, the corner positions and numbers can be different according to different people. Also as there is no standard procedure to measure accuracy of corner detectors we adopted a new approach for creating ground truth. This approach is based on majority human judgement. To create ground truth, ten persons who were familiar with the task of corner detection were chosen. None of them were familiar with the algorithm used by our corner detector. We asked them individually to mark the corners of an image. The corners marked by at least 70% of individuals were selected as the ground truth for that image. The position of a corner in the ground truth was defined as the average of the positions of this corner in individual images marked by those ten persons. Note that test images for computation of accuracy should not have very close corners to avoid confusion. We repeated the same for other images. Then by comparing the detected corners using each of five corner detectors to the list of corners in the ground truth, the accuracy was computed as follows:

Let $N_0$ be the number of corners in the original image (note that $N_0 \neq 0$), $N_a$ the number of matched corners in original image when compared to the ground-truth corners and $N_g$ the number of corners in the ground-truth. The criterion of the accuracy is

$$ACU = 100 \times \frac{N_a}{N_0} + \frac{N_a}{N_g}$$

(7.3)

where ACU stands for "accuracy". In terms of accuracy, the value of ACU for accurate corner detectors should be close to 100%. Then ACU for corner detectors with lower accuracy is closer to zero. Using this definition for accuracy, we compare the value of $N_a$ to both, the number of corners in the original image ($N_0$) and the number of corners in the ground-truth.

<table>
<thead>
<tr>
<th>Average of CCN for</th>
<th>Plessey</th>
<th>K &amp; R</th>
<th>Susan</th>
<th>Orig-CSS</th>
<th>New-CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td>in non-uniform scaling</td>
<td>28%</td>
<td>31%</td>
<td>31%</td>
<td>55%</td>
<td>68%</td>
</tr>
<tr>
<td>in affine transform</td>
<td>14%</td>
<td>11%</td>
<td>9%</td>
<td>42%</td>
<td>51%</td>
</tr>
</tbody>
</table>

Table 7.1: Average of consistency of corner numbers for tested corner detectors
in the ground-truth \((N_g)\). Therefore if a corner detector finds more false corners which implies more matched corners, it does not follow that ACU of this detector is high as well. Because in this case, if \(N_a/N_g\) is near one, \(N_a/N_o\) drops closer to zero. On the contrary, if a corner detector finds less corners which means less matched corners, \(N_a/N_o\) goes to one and \(N_a/N_g\) drops closer to zero. Therefore in both cases, the ACU of such detectors computed through Eq.7.3 is less than 100%. Note that the case of \(N_o=0\) in Eq.7.2 and Eq.7.3 occurs if test images have no corners or the tested corner detectors can not detect any corners. These situations do not arise in practise as only images with many corners are used in experiments and corner detectors under consideration also find many corners in test images.

\[
\text{Figure 7.8: Comparison of consistency in similarity and affine transforms and accuracy for tested corner detectors}
\]
Results

We considered the results of our experiments on many images. Examples of image transforms have been illustrated in Fig.7.5. These experiments were performed as explained in section 7.2.2. After performing our experiments on rotated, uniformly, non-uniformly scaled and affine transformed images, we computed the CCN. The results of these computations for rotation and uniform scaling have been illustrated in Fig.7.6 and Fig.7.7. The average of consistency in non-uniform scaling and affine transform have been shown in table 7.1. The ACCN in this table stands for "Average of Consistency of Corner Numbers". Final test was performed for computation of the accuracy. We computed accuracy using our image collection which included a leaf image, an airplane image, a fish image, a lab image and a building image. All the test images have been illustrated in Fig.7.9. Furthermore, in this figure the corner points of their ground truth also have been shown. The comparison of consistency and accuracy in similarity and affine transforms for tested detectors have been illustrated in Fig.7.8. Overall, the results of these comparisons show the ECSS corner detector has the best accuracy and consistency among these five corner detectors.

7.2.3 The Multi-scale Corner Detector Results and Discussion

The ECSS image corner detection method was very robust with respect to noise and performed better than the other detectors that it was compared to. However by applying the ECSS corner detector to the frames in video sequences, the positions of some corners can change from frame to frame. Therefore we proposed the multi-scale corner detector which is the ECSS corner detector using different scale of smoothing (in the edge detection stage of the ECSS corner detector) in 3.6. We have tested the proposed corner detector on a wide range of real video sequences. The results in some randomly selected frames in a number of video sequences have been demonstrated here. Selected video sequences for demonstration included all transformations, such as translation in bream sequence in Fig.7.10, rotation in Claire sequence in Fig.7.11, scaling in cil sequence in Fig.7.12 and affine transformation in hotel sequence in Fig.7.13. In all the sequences different lighting and different camera motion existed. Results include some
7.2. Corner Detector Results, Discussion, and Performance Evaluation

Figure 7.9: Test images for computation of accuracy
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Figure 7.10: Results of the multi-scale corner detector on some randomly selected frames in bream sequence which included occlusion and non-rigid motion of the bream randomly selected frames of mentioned sequences. All the results confirm the robustness of the multi-scale corner detector with respect to any transformations and moving camera. Therefore a corner tracker based on the multi-scale corner detector has many strong corners on moving objects for tracking.

7.3 The Active Contours Results, Discussion, and Performance Evaluation

7.3.1 The ICEAC Active Contour Model Results, Discussion, and Performance Evaluation

We have applied the ICEAC model to a number of real images. In all experimental tests, initially user draws the position of active contour and sets model parameters. In Fig.7.14 and Fig.7.15, the results of the ICEAC and the AMI on airplane image with same setting of model parameters and initial snake have been illustrated. As locking on to the object as close as possible is very important for some tasks such as image matching and tracking using image features, the final snake of both models on the airplane image and the main edge map of this image are matched in Fig.7.16. The difference between these two models in locking on to the object as close as possible is completely visible through comparison of marked points 1, 2, 3, ..., and 9 in Fig.7.16(a) with similar points in Fig.7.16(b). We can observe that the AMI model has problems
7.3. The Active Contours Results, Discussion, and Performance Evaluation

Figure 7.11: Results of the multi-scale corner detector on some randomly selected frames in the Claire sequence. In this figure, positions of corners around the boundary of claire are more stable than the positions of corners inside her face. However for tracking claire in this sequence, the corners around her boundary are quite enough.

dealing with the parts of image where curvature changes quickly and image contours turn fast. These results show that ICEAC curvature estimation is more accurate. The results of both models with different initial snakes in three lab images are presented in Fig.7.17. In this figure only initial and final snake of these models are illustrated. The results of the ICEAC model on different lip images can be seen in Fig.7.18.

7.3.2 The SAC Active Contour Model Results, Discussion, and Performance Evaluation

We have applied the SAC model to a number of real images. These real images are sorted to two categories; simple and complex images. Also in all experimental tests,
initially user specifies the position of active contour and sets model parameters. Our results are demonstrated with consideration of two points; speed and final shape of the output snake. In Fig.7.20, the final active contour shape of user initial snake using the SAC and the AMI models have been illustrated in some images of the Coil database [64]. The results of applying these methods to complex images have been shown in Fig.7.21. Table 7.2 shows the number of iterations and execution times of the SAC and the AMI active contour models using our tested images. The experiments have been done on an Intel pentium III running at 866 MHz using Linux operating system. The comparison of speed and number of iterations for these two models have been
7.3. The Active Contours Results, Discussion, and Performance Evaluation

Figure 7.13: Results of the multi-scale corner detector on some randomly selected frames in the hotel sequence

Figure 7.14: The behaviour of the ICEAC model, $\alpha=\beta=1$

Figure 7.15: The behaviour of the AMI model, $\alpha=\beta=1$
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Figure 7.16: Comparison between the AMI and the ICEAC models in terms of accuracy with same setting of model parameters and initial snake on the airplane image. In this figure the final snake of each model and the edge map of airplane have been illustrated with continuous and dotted lines respectively.

illustrated in Fig.7.19 using our tested images. All results show the high performance of the SAC model in speed and accuracy. Furthermore in the SAC model, initial snake never cuts the boundary of interest object and never goes inside this object. This improves the stability of the SAC model. The SAC model has only one parameter that affects the internal energy of active contour, therefore it is more independent of model parameters. Also due to using Gaussian filtering with low scale of smoothing, the SAC model is more independent of initial snake as well.

7.3.3 The ASAC Active Contour Model Results, Discussion, and Performance Evaluation

We have applied the ASAC model to a number of real images of the Coil database [64]. In all experiments, initially user specifies the position of an active contour and sets model parameter values. The scale of smoothing is selected by the ASAC algorithm.
and it changes adaptively to the user initial snake. The results in terms of the final snake shape are demonstrated in Fig. 7.23 and compared to the AMI active contour model’s results. Table 7.3 shows the number of iterations and execution times of the ASAC and the AMI active contour models using our tested images. The comparison of speed and number of iterations for these two models have been illustrated in Fig. 7.22 using our tested images. The experiments have been done on an Intel Pentium III running at 866 MHz using Linux operating system. Furthermore, the number of iterations and execution times of applying the SAC, the ASAC and the AMI active contour models to our test images in Fig. 7.26 have been illustrated in Fig. 7.25 and table 7.4. Note that in this experiment, parameters of tested active contour models have been set as following to ensure same conditions in their comparison:

- In the AMI model; $\alpha=1$, $\beta=1$.
- In the ASAC model; $\alpha=1$, $\sigma=10$.
- In the SAC model; $\alpha=1$, $\sigma=1$.

For any image in table 7.2, the number of iterations and execution times of the AMI model are different in comparison to these values for the similar image in table 7.4. Also for any image in table 7.2, the number of iterations and execution times of the SAC model are different in comparison to these values in similar image in table 7.4. These are because the initial snakes which are used in experiments in table 7.2 are different from initial snakes which are used in experiments in table 7.4. However in table 7.2, both models are applied to the same initial snake which is used for each test image and similarly for the third active contour model in table 7.4, which are applied to the same initial snake which is used for each test image.

As results, in the ASAC active contour model, the final snake can lock to the object of interest much quicker than the SAC active contour model only in the single object images. However in complex images still the SAC model is quicker than the ASAC and fast enough to be as our user interface in the proposed content-based video retrieval system.
Figure 7.17: Comparison of the AMI and the ICEAC models in terms of accuracy using different initial snakes in the lab image.
7.3. The Active Contours Results, Discussion, and Performance Evaluation

Figure 7.18: Results of the ICEAC model on different lip images. In each row of this figure, images from left to right are initial snake, final snake and the edges of the underlying image.

Figure 7.19: Comparison of execution time and iterations number between the SAC and the AMI models (refer to table 7.2). In this figure, SACM stands for SAC model.
Figure 7.20: The behaviour of the SAC and the AMI active contour models in terms of the final snake shape in the Coil database. In this figure the images in second column of each row show the initial snakes. The images to the right of the initial snakes show the final snake of the AMI model and the images to the left of the initial snakes show the SAC's. Also the fourth column shows the edges of the underlying images.
Figure 7.21: The behaviour of the SAC and the AMI active contour models in terms of the final snake shape in complex images. In this figure the images in second column of each row show the initial snakes. The images to the right of the initial snakes show the final snake of the AMI model and the images to the left of the initial snakes show the SAC's. Also the fourth column shows the edges of the underlying images.
Table 7.2: Results of execution time (sec) and iterations number of the SAC and the AMI in our tested images. In this table PMT, Time and Iter. stand for parameters, execution time and iterations number respectively.

<table>
<thead>
<tr>
<th>Images</th>
<th>Size</th>
<th>SAC Iter.</th>
<th>SAC Time</th>
<th>AMI Iter.</th>
<th>AMI Time</th>
<th>PMT Iter.</th>
<th>PMT Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cat</td>
<td>156 x 177</td>
<td>31</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>25</td>
<td>6.85</td>
</tr>
<tr>
<td>Piggy</td>
<td>158 x 171</td>
<td>35</td>
<td>7.42</td>
<td>1</td>
<td>1</td>
<td>32</td>
<td>8.80</td>
</tr>
<tr>
<td>Bottle</td>
<td>162 x 178</td>
<td>22</td>
<td>3.16</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>4.5</td>
</tr>
<tr>
<td>Duck</td>
<td>162 x 180</td>
<td>14</td>
<td>1.83</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>3.56</td>
</tr>
<tr>
<td>Box</td>
<td>187 x 117</td>
<td>22</td>
<td>3.81</td>
<td>1</td>
<td>3</td>
<td>29</td>
<td>6.87</td>
</tr>
<tr>
<td>Toy1</td>
<td>152 x 175</td>
<td>17</td>
<td>6.47</td>
<td>1</td>
<td>1</td>
<td>18</td>
<td>9.63</td>
</tr>
<tr>
<td>Toy2</td>
<td>155 x 179</td>
<td>19</td>
<td>4.60</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>6.63</td>
</tr>
<tr>
<td>Cup</td>
<td>153 x 177</td>
<td>15</td>
<td>3.17</td>
<td>1</td>
<td>1</td>
<td>12</td>
<td>5.14</td>
</tr>
<tr>
<td>Glass</td>
<td>150 x 176</td>
<td>15</td>
<td>5.93</td>
<td>1</td>
<td>1</td>
<td>15</td>
<td>6.61</td>
</tr>
<tr>
<td>Car</td>
<td>154 x 178</td>
<td>19</td>
<td>5.04</td>
<td>1</td>
<td>1</td>
<td>17</td>
<td>7.33</td>
</tr>
<tr>
<td>Lab</td>
<td>175 x 153</td>
<td>9</td>
<td>8.66</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Olympic1</td>
<td>176 x 144</td>
<td>11</td>
<td>1.01</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>Olympic2</td>
<td>176 x 144</td>
<td>5</td>
<td>0.47</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>1.06</td>
</tr>
<tr>
<td>Children</td>
<td>176 x 144</td>
<td>9</td>
<td>0.85</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>2.03</td>
</tr>
</tbody>
</table>

Figure 7.22: Comparison of the execution time and iterations number between the ASAC and the AMI models (refer to table 7.3).
Figure 7.23: The behaviour of the ASAC and the AMI active contour models in terms of final snake shape in images: Cat, Piggy, Bottle, Duck, Car, Toy1, Toy2, Cup, Glass and Box. In this figure, from left to right, first column, second column and third column show the final snakes of the ASAC, the initial snakes for both models and the final snakes of the AMI respectively. The fourth column is the edges of the underlying images.
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Figure 7.24: Results of the multiple matching process itself in some randomly selected frames of the bream sequence.

Table 7.3: Results of execution time (sec) and iterations number of the ASAC and the AMI in our tested images. In this table PMT, Time and Iter. stand for parameters, execution time and iterations number respectively.
Table 7.4: Results of execution time (sec) and iterations number of the AMI, SAC, and the ASAC in our tested images. In this table PMT, Time and Iter. stand for parameters, execution time and iterations number respectively.
Figure 7.25: Comparison of the execution time and iterations number among the SAC, the ASAC and the AMI active contour models (refer to table 7.4) using our test images.
Figure 7.26: Test images for evaluating the performance of the SAC active contour model in comparison to the ASAC and the AMI active contour models.
7.4 Corner Tracking Results, Discussion, and Performance Evaluation

7.4.1 The Multiple-match Corner Tracker Results and Discussion

We applied the multiple-match corner tracker to many video sequences of the CVSSP\(^1\), the CMU/VASC\(^2\), and SAMPL\(^3\) databases. In all the sequences different lighting and different camera motion existed. Selected video sequences included all transformations and unconstrained motions; such as translation and occlusion in bream sequence, rotation in Claire and mom-daughter sequences, scaling in cil sequence, affine transformation in hotel sequence and sudden motions in children, tennis, trevor, boxw, heart, mug, and MOVI sequences. In OSU-1 and OSU-2 sequences, all transformations and non-smooth motions are included together and these are good examples of complicated tasks for corner tracking. Bream sequence can also be used as an example of non-rigid motion of the object. The tail or the fin of the bream can move left or right while the whole of the body moves forward. Claire sequence is an example of face and head movements. In cil sequence, camera has forward motion. All the results were compared to the well-known Tommasini feature tracker. Due to having same conditions for both tracker, in Tommasini tracking algorithm the part of feature extraction was removed. Afterwards, experiments were performed as follows: first, the multi-scale corner detector automatically extracted a number of corners in first frame of an input video sequence for both trackers based on a selected $\sigma$. Number of frames in a video sequence was held constant as well. Furthermore, the size of comparing window for removing similar corners in the output of the multi-scale corner detector, the match window, and the value of threshold used for distance criterion in matching process were held constant for all the sequences and for both trackers. Therefore the multiple-match and Tommasini trackers started with same corners in first frame in input sequence and with same conditions. We also attempted to obtain the best possible results for Tommasini tracker by searching for parameter values that appeared to yield the best re-

\(^1\)http://vssp-www.ee.surrey.ac.uk/data/video/index.html
\(^3\)http://sampl.eng.ohio-state.edu/sampl/data/motion/index.htm
7.4. Corner Tracking Results, Discussion, and Performance Evaluation

Results. Some of the multiple-match tracker results have been demonstrated in Fig. 7.27, Fig. 7.28, Fig. 7.29, and Fig. 7.30. In Fig. 7.31 and Fig. 7.32, the path of corners when tracking have been shown as well. In Fig. 7.31 and Fig. 7.32, black point at one end of white path of tracked corners in the frames show the previous position of each moving corner from the first frame of these sequences. Furthermore, in Fig. 7.35, the first frame of all test sequences have been shown. In this figure, for OSU-1 and OSU-2 sequences four frames have been illustrated to show how difficult tasks they are for corner tracking algorithms. Results of the multiple matching process on some randomly selected frames of the bream sequence have been shown in Fig. 7.24. Therefore in comparison to the results in Fig. 5.1, the number of tracked corners among frames in input sequence is more robust and reliable. However still we have a problem with the corners that the multiple matching algorithm can not find any match for them. We solved this problem by monitoring tracked corners when applying two-frame multiple matching to the other frames as it is clear in Fig. 7.27. In this figure, the results of matching process using \( \chi^2 \)-test similarity function combined with monitoring process are illustrated in fifteen successive frames of the bream sequence starting from frame 2. By comparing the same frames such as 6, 11, 14, 15 and 16 in Fig. 5.1 and Fig. 7.27, it is visible that applying the proposed monitoring procedure makes great differences to the results of corner tracking. In frames of Fig. 7.27, only two monitoring corners; MN corner 1 and MN corner 2 ( MN stands for monitoring corner) were selected to show how monitoring corners appear in a frame and then they are removed from or matched with the corners in next frame or next two frames in the input sequence. Also from this figure it is clear why we should have a criterion for removing falsely marked corners, otherwise false MN corners (such as MN corner 1 in frame 3 of the bream sequence in Fig. 7.27 which was a falsely marked corner if it had not found a match in frame 5) will remain in the list of tracked corners. Overall, by applying two-frame multiple matching combined with three-frame monitoring the number of tracked corners stays more robust.

7.4.2 The Results of Corner Tracker including New Corners

We have tested the modified multiple-match corner tracker including New Corners on a wide range of real video sequences. Selected video sequences included all transfo-
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Figure 7.27: Results of monitoring tracked corners in fifteen frames of the bream sequence starting from frame2 using $\chi^2$-test similarity function. In these images MN corner1 and MN corner2 (MN stands for monitoring corner) are two monitored corners which were selected to show how monitored corners appear in a frame and then they are removed or matched with other corners in next frame or next two frames in the sequence.
Figure 7.28: Results of the multiple-match corner tracker in some frames of the cil sequence. Note that in this sequence, camera has forward motion. Comparison of the left and the right sides of frame0 to frame8 in this sequence shows that due to scaling transformation, corners move from middle of these images toward left, right, up and down sides.
Figure 7.29: Results of the multiple-match corner tracker in some randomly selected frames of the Claire sequence.
Figure 7.30: Results of the multiple-match corner tracker in some randomly selected frames of the hotel sequence
Figure 7.31: Results of corner tracking using single matching combined with monitoring in some frames of the Claire sequence including the paths of tracked corners. The black point at one of the end of the white path in these images shows the previous position of each moving corner in first frame of this sequence. The movement of corners in this sequence is concentrated in the face and the head of Claire.
Figure 7.32: Results of corner tracking using single matching combined with monitoring in the hotel sequence including the paths of tracked corners
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

Figure 7.33: Randomly selected results of the multiple match corner tracker using both the corners of frame0 and the corners of each frame that appear fresh after frame0 for tracking in the bream sequence.

motions, such as translation, scaling, rotation and affine transformation. In all the sequences different lighting and different camera motion existed. The results of the proposed corner tracker, which uses both the corners of first frame and the corners of each frame that will appear newly after frame0 for tracking among the input sequence, in some frames of the bream sequence have been illustrated in Fig.7.33. The bream sequence was selected due to occlusion and non-rigid motion of the bream as the tail or the fin of the bream can move left or right while the whole of the body moves forward. These results can be compared to the results of the multiple-match corner tracker in similar frames of the bream sequence in Fig.7.34 considering only the corners of first frame. In both Fig.7.33 and Fig.7.34, we pointed to four corners numbered 1, 2, 3, and 4 in frame9. These numerated corners are missing corners in frame173 of the Fig.7.34 after disocclusion. While in frame173 of the Fig.7.33, by adding new corners for tracking, these corners are appearing in Fig.7.34. Furthermore, in frame117 the bream turns and one of its fins that was hidden, appears and as it can be seen in frame117 of Fig.7.33 it has a corner. But in frame117 of Fig.7.34, it does not have any corner. Also from frame173, it can be realized that if we have more than one objects in
7.4. Corner Tracking Results, Discussion, and Performance Evaluation

Figure 7.34: Randomly selected results of the multiple match corner tracker using only the corners of frame 0 for tracking in the bream sequence

the video sequence, the proposed tracker can still track all the moving objects among the frames of this video sequence.

7.4.3 Performance Evaluation of the Multiple-match Corner Tracker

The main stage of our retrieval system is to track selected corners of user query forwardly and backwardly using the multiple-match corner tracker in order to determine position of identified similar object as query in each video frame. In this section the performance evaluation of our proposed tracker; the multiple-match tracker in comparison to the Tommasini tracker is presented. In Tommasini et al.’s algorithm feature extraction and tracking are combined but in our tracking algorithm which can be found in Sec.5.3, feature extraction is separate from the tracking algorithm. We are tracking the corners which have been extracted by the CSS corner detector. Then actually these two trackers are completely different but we compared our tracker with Tommasini et al.’s tracker as it is a popular feature tracker. Many other authors have referred to Tommasini et al.’s tracker. Therefore having better results for our tracker in compari-
Chapter 7. Results, Discussion, and Performance Evaluation of Proposed System

son to Tommasini tracker confirms the better performance of our tracker. We applied both of these trackers to a number of real video sequences. However due to having same conditions for both trackers in the experiments, the part of feature extraction in Tommasini tracking algorithm was removed. Afterwards, experiments were performed as follows: first, the multi-scale corner detector automatically extracted a number of corners in first frame of an input video sequence for both trackers based on a selected $c$. Also number of frames in a video sequence was held constant as well. Furthermore, the size of comparing window for removing similar corners in the output of the multi-scale corner detector, the match window, and the value of threshold used for distance criterion in matching process were held constant for all the sequences and for both trackers. Therefore the multiple-match and the Tommasini trackers started with same corners in first frame in an input sequence and same conditions. We also attempted to obtain the best possible results for Tommasini tracker by searching for parameter values that appeared to yield the best results. Some of the multiple-match corner tracker results have been illustrated in section 7.4. In Fig.7.35, the first frame of all tested sequences have been shown. In this figure, for OSU-1 and OSU-2 sequences four frames have been illustrated to show how difficult tasks they are for corner tracking algorithms. More information about these video sequences which have used for our experiments can be found in 7.4.1 as well.

Our criterion for performance evaluation (PE) of corner trackers is defined as the number of tracked corners in each frame ($n_t$) divided by the number of corners in first frame ($n_o$),

$$PE = \frac{n_t}{n_o}$$ (7.4)

PE can vary from 0 to 1. An algorithm is the best corner tracker which has $PE=1$ for all values of $t$. As the results of this comparison, it can be discovered that in any video sequence which has unconstrained motion, non-rigid motion and any motion not anticipated in the Tommasini tracker's motion model (such as sequences in Fig.7.36 and Fig.7.37) Tommasini tracker can not track robustly and as the sequence continues this tracker will lose more corners, while our tracker can continue tracking more robustly. In video sequences such as sequences in Fig.7.38 which have a motion model very close or the same as the Tommasini tracker's motion model, this tracker works more
Figure 7.35: The first frame of children, hotel, tennis, MOVI, bream, heart, Claire, cil, Akiyo, mug, mom-daughter, boxw, OSU-1 and OSU-2 sequences from up-left to down-right. For OSU-1 and OSU-2 sequences four frames have been shown to illustrate how difficult tasks they are for corner tracking algorithms.
Figure 7.36: Performance evaluation comparison between the multiple-match corner tracker and the Tommasini feature tracker in the MOVI, mug, OSU-1, OSU-2, heart, and the trevor sequences.
7.4. Corner Tracking Results, Discussion, and Performance Evaluation

Figure 7.37: Performance evaluation comparison between the multiple-match corner tracker and the Tommasini feature tracker in the bream, children, tennis, and the boxw sequences.
7.5 Proposed Content-based Retrieval System Results and Discussion

We have tested the proposed retrieval system in a wide range of real video databases. Selected video sequences included all transformations and unconstrained motions; such as translation and occlusion in bream and artichoke sequences, rotation in claire and akyio sequences, scaling in cil sequence, affine transformation in hotel sequence and robustly while our tracker loses more corners, but it is still suitable for multimedia application due to having enough corners on query for retrieving. Overall, application of the multiple-match corner tracker showed that this tracker can be used successfully for retrieving moving objects through video databases in a more practical and efficient way based on having sufficient robust corners on moving objects. In fact, it is a more general purpose tracker.

Figure 7.38: Performance evaluation comparison between the multiple-match and the Tommasini trackers in the mom-daughter, hotel, Claire, and the cil sequences.
sudden motions on children, tennis and MOVI sequences. Bream sequence can be used also as an example of non-rigid motion of the object. The tail or the fin of the bream can move left or right while the whole of the body moves forward. Claire and akyio sequences are examples of face and head movements. In the cil sequence, camera has forward motion. Experiments were performed as follows: first we selected a frame among the frames of input sequence. Next by applying the SAC active contour model we drew a snake as close as possible around an interested object in that frame using computer mouse. After that proposed system automatically has continued its process in order to determine all similar objects as query and to demonstrate them to the user. Demonstration of identified similar objects as query has been carried out using both of the bounding box and the least-squares estimation methods. Furthermore in each figure initial and final snakes around user query, all the corners of the underlying frame including query, selected corners in the query, and some of the randomly selected of the retrieved objects for query in other frames have been shown. Note that the final snake is the final iteration output of the SAC algorithm. Actually a user initial snake which has been locked as close as possible to the user query was referred as the final snake. All the results confirm that our proposed content-based video retrieval system, which is based on extracting shots, detecting corners from frames in input sequence and tracking them through video database, is believed to be very efficient and practical especially due to having:

- An easy, accurate and quick user interface
- An accurate and robust corner detector
- An efficient and practical corner tracker

Especially it demonstrates good results in the case of unconstrained motion as its procedure is not based on any motion models or limiting assumptions. Some of the retrieved results using proposed retrieval system have been illustrated in different sequences from Fig.7.39 to Fig.7.54. In Fig.7.42, backward tracking of the akiyo sequence from frame100 to the first frame is quite good but forward tracking due to more changes in the lightening of the background in forward frames, has more changes in the positions
of drawn snake around identified object as query in each frame. Also in Fig.7.48, due to the same problem, tracking corners in both directions are not perfect. Therefore the proposed system still has problems with changes in the lightening of the background of the frames in video sequences. Note that these changes influence the detection of corners and in finding correspondences. However as it can be seen in Fig.7.48, in such situations, still the system can draw a snake very close to the real position of query in each frame. Furthermore a very important problem that may occur is if user draws a very small snake in the selected-frame where there is no corner, then there will be no retrieved results either. Therefore as we mentioned in Chap.1, to prevent such problems, other low level features such as colour, texture or even integrating high-level concepts such as objects and events can be added to the proposed system to make it more useful for multimedia databases retrieval.
Figure 7.39: Some randomly selected of identified similar objects as query in the artichoke sequence including 100 frames. Query (Q.) was selected in artichoke50. The least-squares estimation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.40: Some randomly selected of retrieved objects for query (Q.) in the artichoke sequence including 100 frames. Query was selected in artichoke50. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.41: Some randomly selected of retrieved objects for query in the akiyo sequence including 300 frames. Query (Q.) was selected in artichoke100. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corners stand for underlying and corners respectively.
Figure 7.42: Some randomly selected retrieved objects for query [7.41(a)] in the akiyo sequence including 300 frames using the least-squares estimation.
7.5. Proposed Content-based Retrieval System Results and Discussion

Figure 7.43: Some randomly selected of retrieved objects for query in the bream sequence including 300 frames. Query (Q.) was selected in bream95. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.44: Some randomly selected of retrieved objects for query [7.43(a)] in the bream sequence including 300 frames using the least-squares estimation.
Figure 7.45: Retrieved objects for query in the cil sequence including 10 frames. Query (Q.) was selected in cil0. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.46: Retrieved objects for query [7.45(a)] in the cil sequence including 10 frames using the least-squares estimation.
Figure 7.47: Some randomly selected of retrieved objects for query in the claire sequence including 300 frames. Query (Q.) was selected in claire150. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corners stand for underlying and corners respectively.
Figure 7.48: Some randomly selected of retrieved objects for query [7.47(a)] in the claire sequence including 300 frames using the least-squares estimation.
Figure 7.49: Some randomly selected of retrieved objects for query in the children sequence including 150 frames. Query (Q.) was selected in claire60. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.50: Some randomly selected of retrieved objects for query [7.49(a)] in children sequence including 150 frames using least-squares estimation.
Figure 7.51: Retrieved objects for query in the MOVI sequence including 9 frames. Query (Q.) was selected in MOVI0. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.52: Retrieved objects for query [7.51(a)] in the MOVI sequence including 9 frames using the least-squares estimation.
Figure 7.53: Some randomly selected of retrieved objects for query in the tennis sequence including 30 sudden motion frames. Query (Q.) was selected in tennis15. The bounding box presentation was used for demonstrating the retrieved objects. Furthermore in this figure Underly and corns stand for underlying and corners respectively.
Figure 7.54: Some randomly selected of retrieved objects for query [7.53(a)] in the tennis sequence including 30 frames using the least-squares estimation.
Chapter 8

Conclusions

In this thesis, a novel content-based video retrieval system using shape features was proposed. In the first chapter, before starting to focus deeply on the target of this project, some primary definitions related to the concept of this thesis were described. Afterwards, content-based video retrieval system, content-based itself, and low level visual features were defined. Then the reasons for selecting corners as the most intuitive types of point features on shapes were described. Next content-based video retrieval systems were categorised. Finally the stages of our proposed retrieval system and also our contributions in this field have been pointed out.

In chapter 2, a critical review relevant to key aspects of this project including cut detectors, corner detectors, active contours, feature trackers, and content-based video database retrieval was given.

Since real-time corner tracking algorithm which is used for multimedia applications needs robust corners that can be identified and tracked from frame to frame in video sequences, the multi-scale corner detector was presented in chapter 3. Actually the ECSS corner detector which includes multi-scale edge detector is referred to as the multi-scale corner detector. Therefore by applying Canny edge detector with different scales of smoothing, we considered all changes that may happen in the edge contours of a frame in comparison to its next frame or variation of illumination or moving camera. Then we selected the local maxima of all edge contours of that frame as corner candidates which are extracted using a multi-scale edge detector. Next by removing
the corners which were marked more than once in three outputs of the ECSS corner detector using three different values of $\sigma$, the remaining corners were illustrated as the output of the multi-scale corner detector. The positions of corners in the output of the multi-scale corner detector are more robust and stable in comparison to the output of the ECSS corner detector itself with respect to noise and any transformations. All the multi-scale corner detector experimental results in section 7.2.3 confirmed the existence of many robust and strong corners on the moving objects useful for tracking. The ECSS corner detector with better performance on blunt and rounded corners was proposed in Sec.3.5.

We proposed three active contour models: the ICEAC, the SAC and the ASAC based on reformulating of the internal energy of active contours in the energy minimising active contour models in chapter 4. In the ICEAC, the external energy of active contour at any point is computed using the minimum distance of each point to the underlying image edges. Curvature at any point on active contour is estimated using its two neighbouring points. Application of this new procedure resulted in locking on the interest object very accurately. Although parameters setting does affect the results in general, by following a simple procedure for setting the ICEAC model parameters mentioned in that chapter, setting takes place once. In the SAC model; our main contribution in that chapter, the curvature part of the internal energy of active contour is removed and replaced by CSS technique for smoothing. We smooth the output of each iteration in a discrete multi-stage process until locking on to the underlying image edges occurs at least at one point. The remaining of iterations continue without smoothing until minimum energy is reached and the whole process halts. Therefore in the SAC model, initial snake never cuts the boundary of interest object and never goes inside this object. This improves the stability of the SAC model. Application of this new procedure resulted in locking on the interest object quickly and accurately. The SAC model has only one parameter that affects the internal energy of active contour, therefore it is more independent of model parameter. Also due to using Gaussian filtering with low scale of smoothing, the SAC model is more independent of initial snake as well. In the ASAC model which is an adaptive active contour model, the curvature part of internal energy of active contour is removed and replaced by adaptive CSS filtering with an
adaptive scale, $\sigma$ for smoothing. In this algorithm since the initial $\sigma$ can be selected much higher than the initial $\sigma$ in the SAC active contour model, then final snake can lock to object of interest much quicker only in the single object images. However in complex images still the SAC model is quicker than the ASAC and fast enough to be used as our user interface in the proposed content-based video retrieval system. These three active contour models were tested using many single and multiple objects images. Overall, the results illustrated that these three models can be used in matching and tracking tasks for retrieving multimedia databases.

In chapter 5, first the shortcomings of the Tommasini tracker were discussed in Sec. 5.2. Afterwards, the multiple-match corner tracker was proposed in section 5.3. The multiple-match corner tracker is a robust corner tracker based on two-frame correspondence using the multiple matching algorithm including not the corners of first frame but also all the corners that appear fresh in each frame of the video sequence. In its algorithm despite having problems finding the best correspondence, applying multiple matching together with distance thresholding helps to increase the chance of finding as close as possible at least one match for each tracked corner. The problem of recovering points lost during the tracking was also addressed using the three-frame monitoring process. The three-frame monitoring helps to ensure that the number of tracked corners and their tracked positions among frames become more robust. However in the case of real corners lost, marking corners by monitoring process have been stopped by assigning different flags to the corners marked by the two-frame multiple matching and monitoring process. Including new corners help user to specify an object of interest as query in any frames of input video sequence. We applied the multiple-match corner tracker to many video sequences including different lighting, different camera motion, all transformations, unconstrained and non-smooth motions. All the results were compared to the Tommasini feature tracker's results.

Finally after preparing all the image processing tools that we needed to construct our retrieval system, the proposed content-based video retrieval system was introduced in chapter 6. Therefore in this chapter, a content-based video retrieval system using shape features was proposed. To provide indices, first all the shots in a video sequence were extracted. Then corners of all frames in each video shot were detected applying the
proposed multi-scale corner detector. As a user interface, the SAC active contour model was used to specify one object of interest as a query in one of the frames of each video shot. Then the proposed retrieval system automatically has determined the position of identified similar object for query in each video frame. For this purpose, selected corners of user query were tracked forwardly and/or backwardly using the multiple-match corner tracker. Two approaches of the least-squares estimation and bounding box methods were considered for demonstrating the retrieved results. Applying these two methods does not affect retrieval results and both of them can be used by user. Experiments have been carried out on a wide range of real video databases mentioned in section 7.4.

After explaining all the theories underlying the proposed retrieval system, all the results, discussions and performance evaluations of the proposed retrieval system through its individual stages have been demonstrated in chapter 7. Performance of the ECSS corner detector in comparison to the performances of several popular corner detectors under similarity and affine transformations was evaluated in 7.2.2. For this evaluation new criteria were defined theoretically that do not favour algorithms which find more false corners in input images. Furthermore the new procedure of majority human judgement was proposed and used for creating the ground truth to test the accuracy of corner detectors. The application of these criteria showed that the ECSS algorithm produced a good results with respect to noise and similarity and affine transforms. In section 7.3, three proposed active contour models; the ICEAC, the SAC and the ASAC results were compared with each others and with Amini et al. [2] energy-minimising active contour model in terms of speed and number of iterations. From these results the SAC active contour model which is an accurate and quick active contour model was selected as the best user interface for the proposed retrieval system. Performance evaluation of the multiple-match corner tracker in comparison to the Tommasini tracker was presented in section 7.4. All the results confirmed that the multiple-match corner tracker can be used successfully for retrieving moving objects through video databases in a more practical and efficient way based on having sufficient robust corners on moving objects. In section 7.5, the results of our proposed content-based video retrieval system in a wide range of real video databases have been illustrated. All the results confirm that
the proposed real-time content-based video retrieval system is efficient, easy to use and generally applicable.

8.1 Main Advantages of the Proposed System

The main advantages of the proposed content-based video retrieval system are as following:

- Proposed retrieval system based on applying the multi-scale corner detector has a sufficient number of robust corners in each frame of input sequence.

- Proposed retrieval system based on employing the two-frame multiple matching has at least a match for each tracked corner among frames in input sequence.

- In the proposed retrieval system, false matches are removed using a distance criterion based on a reference point. However the value of threshold is constant.

- In the proposed retrieval system, problem of recovering points lost during tracking is solved using the three-frame based monitoring.

- Since the proposed retrieval system is based on applying a corner tracker which does not make any assumptions or does not use any special motion models in tracking feature points, it is more practical and more efficient for unconstrained and non-smooth motions.

- In the proposed retrieval system, based on the SAC active contour model selection of a query in frames of the input video sequences is easy, quick and attractive for user which has always problem in querying of audio-visual features such as colour, texture, shape and motion in audiovisual feature-based retrieval systems.

- In the proposed retrieval system, retrieved results are demonstrated employing a snake-based model. Therefore user can have a very close communication with the system. This is because she/he has chance to learn from the results how to draw a user snake around object of interest that can have better results.
• In the proposed retrieval system, any kind of video data including any motion model especially non-smooth motion, any transformations, or any lighting can be processed as input sequence and then good and quick results have been given. Therefore the proposed system is more general and efficient.

• In proposed retrieval system, based on our user interface user can access to individual video objects in the video stream.

Finally as the proposed retrieval system can determine the positions of objects similar to the query based on extracting corners from the query and from the frames of input sequence, then if no corners can be found, consequently no similar objects can be found either. This is the only limitation that our system has. Therefore it would be interesting to consider other low-level features combined with corners to retrieve a query among frames. By combining corners with colour, texture and even motion vectors in the video streams not only the mentioned problem of our system but also the shortcomings of systems which are based only on colour, or texture can be solved as well. Furthermore, adding an object recognition method or integrating high-level concepts such as objects and events which are semantic content to this system can generate more promising results.
Appendix A

Similarity Functions Metrics

Corner matching is commonly referred to as the correspondence problem. The problem is how to automatically match corresponding corners from two images, while no incorrect matches are assigned at the same time. The common approach for corners is to take a match window with proper size around detected corner and compare this with a similar match window around each of the candidate corners in the other image. Each comparison yields a score, a measure of similarity. The match is assigned to the corner with the highest matching score.

In our matching process, we have two sets of corners as input to the matcher. For instance, at the start, matcher receives the corners of frame0 and frame1 from output of the multi-scale corner detector. For every corner in frame0, we construct a window match with size of $11 \times 11$ on frame1 centred in the same position of that corner in frame0. Then all corners of frame1 lying in this window are match candidates for that corner in frame0. If more than one corner lie in this window, we need to compute a similarity function for each of them. Afterwards the winner candidate is one with the highest score of similarity. For computation of similarity function, we consider a small match window with size of $3 \times 3$ around each corner of frame0 that has more than one match and also around each of its match candidates in next frame. Each pixel of this window in frame0 is named $i$, and also each pixel of the same size window around each match candidate of that corner in frame1 is named $j$. Our selection among different similarity functions are standard cross-correlation, zero mean cross-correlation, sum of
Appendix A. Similarity Functions Metrics

Squared differences, and $\chi^2$-test. In definition of these functions, $n$ is the size of match window; for example, 3 in our matching process which is used $3 \times 3$ match window.

- **Standard Cross-Correlation;** SCC which yields a score between 0 and 1:

  \[
  C_1 = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} ij}{\sqrt{\sum_{i=0}^{n} i^2 \sum_{j=0}^{n} j^2}} \tag{A.1}
  \]

  Candidate with maximum value of $C_1$ is the best match.

- **Zero Mean Cross-Correlation;** ZM-CC which yields a score between -1 and 1:

  \[
  C_2 = \frac{\sum_{i=0}^{n} \sum_{j=0}^{n} (i - \bar{i})(j - \bar{j})}{\sqrt{\sum_{i=0}^{n} (i^2 - n\bar{i}^2) \sum_{j=0}^{n} (j^2 - n\bar{j}^2)}} \tag{A.2}
  \]

  Where $\bar{i}$ is the mean value of $i$ and computed by $\bar{i} = [(\sum_{i=0}^{n} i)/n]$ and similar for $\bar{j}$. Candidate with maximum value of $C_2$ is the best match.

- **Sum of Squared Differences;** SSD which provides the Euclidean distance between two distributions:

  \[
  C_3 = \sum_{i=0}^{n} \sum_{j=0}^{n} (i - j)^2 \tag{A.3}
  \]

  Candidate with minimum value of $C_3$ is the best match.

- **$\chi^2$-test** which measures similarity between two distributions:

  \[
  C_4 = \sum_{i=0}^{n} \sum_{j=0}^{n} \frac{(i - j)^2}{(i + j)/2} \tag{A.4}
  \]

  Candidate with minimum value of $C_4$ is the best match.

Other similarity functions are Kolmogorov-Smirnov distance, Jeffrey divergence, and Earth Mover distance. Information about these similarity functions and their formulas can be found in [78].
Bibliography


