الخلاصة

مازالت تصنيفات الصور و الفيديو من المشاكل المهمة لفهم محتويات الوسائط المتعددة التي تتطلب تقليل الفجوة بين الأوصاف المعنوية المطلوبة والواصفات المرئية الواطئة المستوى التي يُحصل عليها تلقائياً، في نفس الوقت هي أدوات ثمينة للتطبيقات الأخرى مثل كشف الأجسام، التمييز، وصف المحتويات البصرية، التوليد المعنوي ل(metadata)، الفهرسة، الاسترجاع.

يهدف هذا العمل إلى تقسيم الفيديو إلى عدد من اللقطات باستعمال ثلاث خوارزميات مختلفة، ثم تصنيف كتل صور الفيديو إلى كتل ساكنة (static blocks) وكتل متحركة (dynamic blocks) بالاعتماد على الفرق بين الكتل في صور الفيديو المتلاحقة، يليه استخراج نوعين من الخصائص من الكتل الساكنة والكتل المتحركة لصور الفيديو، الخصائص المتبناة هي:

١. الخصائص الإحصائية (statistical features)
(المتوسط (mean)، الانحراف المعياري (standard deviation)، متوسط الانحراف المطلق (mean absolute deviation)، الانحراف (skewness))

٢. الخصائص النسيجية (textural features)
(طاقة الميل (energy of gradient)، التباين١(contrast 1)، التباين٢(contrast 2)، تعديل لبُعد صورةٍ نمطي هندسي متكرر (modification fractal dimension H))

لقد تم قياس قوة التميز للخصائص المستخرجة باستخدام إما كل خاصية على حدة أو مجموعات من الخصائص (إلى حد ٥ خصائص)، وقد أشارت نتائج الاختبار بان بعض مجموعات هذه الخصائص مفيدة لتميز لقطات الفيديو عن بعضها بنجاح خصوصا إذا عندما تحتوي متتاليات الفيديو (video sequences) المستخرجة من اللقطات على عدد من صور الفيديو اكبر من (٢٥) صورة، و أشارت نتائج الاختبار أيضاً إلى إن كل الخصائص المستخرجة من الكتل الساكنة ماعدا (H) أعطت قوة تميز أعلى من الخصائص المستخرجة من الكتل المتحركة.

و قد تم استخدام خوارزمية أل (K-means) لتصنيف لقطات الفيديو إلى عدد من الأصناف ولقد أشارت نتائج الاختبار إلى إن هذه الخوارزمية أعطت استقرارا جيدا في التصنيف لان معظم متتاليات الفيديو المستخرجة من اللقطات قد صنفت كعضو في نفس الصنف.

Abstract

Image and video classifications, are important problems in multimedia content understand that requires bridging the gap between the target semantic categories, or classes, and the low-level visual descriptors that can be automatically obtained. At the same time, they are valuable tools towards other applications like object detection and recognition, visual content description, semantic metadata generation, indexing and retrieval.

This work aims to segment the video into a number of shots using three different types of algorithms, classify the video frames data into static and dynamic blocks depending on the difference between the blocks of successive frames, extract two types of features from the static and dynamic blocks of the video shots, the adopted features are:

1. The statistical features (mean, standard deviation, mean absolute deviation, skewness).
2. The textural features (energy of gradient, contrast, modification fractal dimension H).

The discrimination power for the extracted features was determined by using each features alone, or combinations of features (up to 5. features).

The test results indicated that some combinations of these features are useful to successfully recognize the video shots from each other especially when the extracted video sequences from any shot consist of video frames more than (٢٥) frame the results also showed that the features extracted from the static blocks (except the H) gave higher

discrimination power than the features extracted form the dynamic blocks.

Finally the K-means clustering algorithm was used to categorize the video shots into a number of classes. The test results indicate that this algorithm shows good stability in classifying the video shots, because most of the extracted sequences from each shot were classified as members to the same class.

Acknowledgment

First of all, my great thanks to Allah who helped me and gave me the ability to achieve this work.

I would like to express my appreciation and my deepest gratefulness to my supervisor Dr. Loay A. George for his guidance, supervision and his efforts during the development of this work.

Grateful thanks for the head of the department Dr. Taha S. Bashaga for his help and support.

My deep gratitude to my lovely family, To the precious, my mother, my father, my brothers mohanad and faseel and my sister hadeel for their love and help during the study years.

Finally thanks to all employees and stuff of the computer science department and to all my friends.

Appendix A

AVI File Structure

A.1 AVI Main Header

In this section the elements of the header section of AVI files are given. The AVI file begins with the main header. This header is identified by the string 'avih' which is consists of (four-character code). The header contains global information for the entire AVI file, such as the number of streams within the file and the width and height of the AVI sequence. The items of the AVI main header are listed in the table (A-1) [Msdn98]:

```
typedef struct {
            DWORD dwMicroSecPerFrame;
            DWORD dwMaxBytesPerSec;
            DWORD dwReserved1;
            DWORD dwFlags;
            DWORD dwTotalFrames;
            DWORD dwInitialFrames;
            DWORD dwStreams;
            DWORD dwSuggestedBufferSize;
            DWORD dwWidth;
            DWORD dwHeight;
            DWORD dwReserved[4];
} MainAVIHeader;
```

Table (A.1) The contents of MainAVIHeader

dwMicroSecPerFrame	Specifies the time interval (in microseconds) between frames. This value indicates the overall timing of the file.
dwMaxBytesPerSec	Specifies the approximate maximum data rate of the file. This value indicates the number of bytes per second the system must handle to present an AVI sequence, as specified by the other parameters contained in the main header and stream header chunks.
dwReserved1	Reserved. Set this to zero.
dwFlags	Contains the flags. The following flags may exists: AVIF_HASINDEX: Indicates the AVI file has an 'idx1' chunk containing an index at the end of the file. For good performance, all AVI files should contain an index. AVIF_MUSTUSEINDEX: Indicates that the index, rather than the physical ordering of the chunks in the file, should be used to determine the order of presentation of the data. For example, you could use this to create a list of frames for editing. AVIF_ISINTERLEAVED: Indicates the AVI file is interleaved. AVIF_WASCAPTUREFILE: Indicates the AVI file is a specially allocated file used for capturing real-time video. Applications should warn the user before writing over a file with this flag set because the user probably defragmented this file. AVIF_COPYRIGHTED: Indicates the AVI file contains copyrighted data and software. When this flag is used, software should not permit the data to be duplicated.
dwTotalFrames	Specifies the total number of frames of data in the file.
dwInitialFrames	Specifies the initial frame for interleaved files. For Noninterleaved files it should set zero. While for interleaved files it should specifies the number of frames in the file prior to the initial frame of the AVI sequence in this member.

Table (A.1) Continue

dwStreams	Specifies the number of streams in the file. For example, a file with audio and video has two streams.
dwSuggestedBufferSize	Specifies the suggested buffer size for reading the file. Generally, this size should be large enough to contain the largest chunk in the file. If set to zero, or if it is too small, the playback software will have to reallocate memory during playback, which will reduce performance. For an interleaved file, the buffer size should be large enough to read an entire record, and not just a chunk.
dwWidth	Specifies the width of the AVI file in pixels.
dwHeight	Specifies the height of the AVI file in pixels.
dwReserved[4]	Reserved. Set this array to zero.

A.2 AVI Stream Headers

The main header is followed by one or more 'strl' chunks. A 'strl' chunk is required for each data stream. These chunks contain information about the streams in the file. Each 'strl' chunk must contain a stream header and stream format chunk. Stream header chunks are identified by the *FOURCC* (four-character code) 'strh', and the stream format chunks are identified by the *FOURCC* 'strf'. In addition to the stream header and stream format chunks, the 'strl' chunk might also contain a stream-header data chunk and a stream name chunk. Stream-header data chunks are identified by the *FOURCC* 'strd'. Stream name chunks are identified by the *FOURCC* 'strn'. The stream header structure contains header information for a single stream of a file, and it specifies the type of data the stream contains, such as audio or video, by means of a *FOURCC* table (A.2) content information about of the stream header [Msdn98]:

```
typedef struct {
        FOURCC fccType;
        FOURCC fccHandler;
        DWORD  dwFlags;
        DWORD  dwPriority;
        DWORD  dwInitialFrames;
        DWORD  dwScale;
        DWORD  dwRate;
        DWORD  dwStart;
        DWORD  dwLength;
        DWORD  dwSuggestedBufferSize;
        DWORD  dwQuality;
        DWORD  dwSampleSize;
        RECT   rcFrame;
      } AVIStreamHeader;
```

Table (A.2) the contents of the AVIStreamHeader

fccType	Contains a **FOURCC** that specifies the type of the data contained in the stream. The following standard AVI values for video and audio are defined: 'vids': Indicates the stream contains video data. The stream format chunk contains a BITMAPINFO structure that can include palette information. 'auds': Indicates the stream contains audio data. The stream format chunk contains a WAVEFORMATEX or PCMWAVEFORMAT structure. 'txts': Indicates the stream contains text data.
fccHandler	Optionally, contains a **FOURCC** that identifies a specific data handler. The data handler is the preferred handler for the stream. For audio and video streams, this specifies the installable compressor or decompressor.

Table (A.2) Continue

dwFlags	Contains any flags for the data stream. The bits in the high-order word of these flags are specific to the type of data contained in the stream. The following standard flags are defined: AVISF_DISABLED: Indicates that this stream should not be enabled by default. AVISF_VIDEO_PALCHANGES: Indicates that this video stream contains palette changes. This flag warns the playback software that it will need to animate the palette.
dwPriority	Specifies priority of a stream type. For example, in a file with multiple audio streams, the one with the highest priority might be the default stream.
dwInitialFrames	Specifies how far audio data is skewed ahead of the video frames in interleaved files. Typically, this is about 0.75 seconds. If you are creating interleaved files, specify the number of frames in the file prior to the initial frame of the AVI sequence in this member.
dwScale	Used with **dwRate** to specify the time scale that this stream will use. Dividing **dwRate** by **dwScale** gives the number of samples per second. For video streams, this rate should be the frame rate. For audio streams, this rate should correspond to the time needed for **nBlockAlign** bytes of audio, which for PCM audio simply reduces to the sample rate.
dwRate	See **dwScale**.
dwStart	Specifies the starting time of the AVI file. The units are defined by the **dwRate** and **dwScale** members in the main file header. Usually, this is zero, but it can specify a delay time for a stream that does not start concurrently with the file.
dwLength	Specifies the length of this stream. The units are defined by the **dwRate** and **dwScale** members of the stream's header.

Table (A.2) Continue

dwSuggestedBufferSize	Specifies how large a buffer should be used to read this stream. Typically, this contains a value corresponding to the largest chunk present in the stream. Using the correct buffer size makes playback more efficient. Use zero if you do not know the correct buffer size.
dwQuality	Specifies an indicator of the quality of the data in the stream. Quality is represented as a number between 0 and 10,000. For compressed data, this typically represents the value of the quality parameter passed to the compression software. If set to –1, drivers use the default quality value.
dwSampleSize	Specifies the size of a single sample of data. This is set to zero if the samples can vary in size. If this number is nonzero, then multiple samples of data can be grouped into a single chunk within the file. If it is zero, each sample of data (such as a video frame) must be in a separate chunk. For video streams, this number is typically zero, although it can be nonzero if all video frames are the same size. For audio streams, this number should be the same as the **nBlockAlign** member of the WAVEFORMATEX structure describing the audio.
rcFrame	Specifies the destination rectangle for a text or video stream within the movie rectangle specified by the **dwWidth** and **dwHeight** members of the AVI main header structure. The **rcFrame** member is typically used in support of multiple video streams. Set this rectangle to the coordinates corresponding to the movie rectangle to update the whole movie rectangle. Units for this member are pixels. The upper-left corner of the destination rectangle is relative to the upper-left corner of the movie rectangle.

The last eight members in table (A.2) describe the playback characteristics of the stream. These factors include the playback rate (*dwScale* and *dwRate*), the starting time of the sequence (*dwStart*), the length of the sequence (*dwLength*), the size of the playback buffer (*dwSuggestedBuffer*), an indicator of the data quality (*dwQuality*), and the sample size (*dwSampleSize*).

Some of the members in the stream header structure are also present in the main header structure. The data in the main header applies to the whole file, while the data in the stream header structure applies only to a stream.

A stream format ('strf') chunk must follow a stream header ('strh') chunk. The stream format chunk describes the format of the data in the stream. For video streams, the information in this chunk is a *BITMAPINFO* structure (including palette information if appropriate).

The 'strl' chunk might also contain an additional stream-header data ('strd') chunk. If used, this chunk follows the stream format chunk. The format and the contents of this chunk are defined by installable compression or decompression drivers. Typically, drivers use this information for configuration. Applications that read and write RIFF files do not need to decode this information. They transfer this data to and from a driver as a memory block.

The optional 'strn' stream name chunk provides a zero-terminated text string describing the stream. The AVI file functions can use this chunk to let applications identify the streams they want to access by their names.

An AVI player associates the stream headers in the LIST 'hdrl' chunk with the stream data in the LIST 'movi' chunk by using the order of the 'strl' chunks. The first 'strl' chunk applies to stream 0; the second applies to stream 1, and so forth. For example, if the first 'strl' chunk describes the wave audio data, the wave audio data is contained in stream 0. Similarly, if the second 'strl' chunk describes video data, then the video data is contained in stream 1.

A.3 Stream Data (LIST 'movi' Chunk)

Following the header information is a LIST 'movi' chunk that contains chunks of the actual data in the streams (that is, the pictures and sounds themselves). The data chunks can reside directly in the LIST 'movi' chunk or they might be grouped into 'rec' chunks. The 'rec' grouping implies that the grouped chunks should be read from disk all at once. This is used only for files specifically interleaved to play from CD-ROM.

Like any RIFF chunk, the data chunks contain a *FOURCC* (four-character code) to identify the chunk type. A *FOURCC* is a 32-bit quantity represented as a sequence of one to four ASCII alphanumeric characters, padded on the right with blank characters. The **FOURCC** that identifies each chunk consists of the stream number and a two-character code that defines the type of information encapsulated in the chunk. For example, a waveform chunk is identified by a two-character code of 'wb'. If a waveform chunk corresponded to the second LIST 'hdrl' stream description, it would have a *FOURCC* of '01wb'.

Appendix B
BMP File Structure

B.1 BITMAPINFOHEADER Structure

The *BITMAPINFOHEADER* structure contains information for the video stream of an AVI RIFF file. Table (B.1) describe the content of *BITMAPINFOHEADER* structure [Msdn98]:

```
typedef struct tagBITMAPINFOHEADER {
        DWORD  biSize;
        LONG   biWidth;
        LONG   biHeight;
        WORD   biPlanes;
        WORD    biBitCount;
        DWORD   biCompression;
        DWORD   biSizeImage;
        LONG    biXPelsPerMeter;
        LONG    biYPelsPerMeter;
        DWORD    biClrUsed;
        DWORD    biClrImportant;
} BITMAPINFOHEADER;
```

Table (B.1) The content of the BITMAPINFOHEADER

biSize	Specifies the number of bytes required by the structure.
biWidth	Specifies the width of the bitmap, in pixels.
biHeight	Specifies the height of the bitmap, in pixels. If **biHeight** is positive, the bitmap is a bottom-up DIB (device-independent bitmap) and its origin is the lower left corner. If **biHeight** is negative, the bitmap is a top-down DIB and its origin is the upper left corner.
biPlanes	Specifies the number of planes for the target device. This value must be set to 1.
biBitCount	Specifies the number of bits per pixel. Some compression formats need this information to properly decode the colors in the pixel.
biCompression	Specifies the type of compression used or requested. Both existing and new compression formats use this member.
biSizeImage	Specifies the size, in bytes, of the image. This can be set to 0 for uncompressed RGB bitmaps.
biXPelsPerMeter	Specifies the horizontal resolution, in pixels per meter, of the target device for the bitmap. An application can use this value to select a bitmap from a resource group that best matches the characteristics of the current device.
biYPelsPerMeter	Specifies the vertical resolution, in pixels per meter, of the target device for the bitmap.
biClrUsed	Specifies the number of color indices in the color table that are actually used by the bitmap. If this value is zero, the bitmap uses the maximum number of colors corresponding to the value of the **biBitCount** member for the compression mode specified by **biCompression**.
biClrImportant	Specifies the number of color indices that are considered important for displaying the bitmap. If this value is zero, all colors are important.

When the value in the *biBitCount* member is set to greater than eight, video drivers can assume bitmaps are true color and they do not use a color table. While if the value in the *biBitCount* member is set to less than or equal to eight, video drivers can assume the bitmap uses a palette or color table defined in the *BITMAPINFO* data structure. This data structure has the following members:

```
typedef struct tagBITMAPINFO {
        BITMAPINFOHEADER bmiHeader;
        RGBQUAD          bmiColors;
} BITMAPINFO;
```

The (BITMAPINFOHEADER), member specifies a BITMAPINFOHEADER structure. The (BITMAPINFO) member specifies an array of RGBQUAD data types that define the colors in the bitmap.

Supervisor Certification

I certify that this thesis was prepared under my supervision at the Department of Computer Science/College of Science/ Al-Nahrain University, by **Hind Ali Al-Kitt** as a partial fulfillment of the requirement for the degree of Master of Science in Computer Science.

Signature**:**
Name**: Dr. Loay A. George**
Title**: Senior Researcher**
Date**: / / 2006**

In view of the available recommendation, I forward this thesis for debate by the examination committee.

Signature**:**
Name**: Dr. Taha S. Bashaga**
Title**: Head of the Department of Computer Science, Al-Nahrain University**
Date**: / / 2006**

Chapter Five
Conclusions and Suggestions

5.1 Conclusions

Form the test results; conducted to investigate the performance of the proposed system the following remarks were derived:

1. The three designed algorithms for shot boundaries detection can exactly detect the boundaries of the shots for all the tested video, which have the (*cut*) boundary type, and it failed in detecting other types of shot boundaries types (like fade, dissolve, wipe).

2. The results of blocks classification algorithm indicate that the number of static and dynamic blocks belong to each frame in the video are different from one frame to another depending on the changes in the pixels data between the current frame and the next frame in the video.

3. The features analysis results indicate that the increase in the number of video frames belong to the video sequences (extracted from the any shot) will improve the recognition efficiency of the adopted features.

4. The results of the power discrimination analysis indicate that all the statistical and textural features (except the H), extracted from the static blocks have a higher discrimination rate than the corresponding features extracted from the dynamic blocks.

5. The results of the power discrimination analysis also indicate that the features (contrast2 and the energy of the horizontal gradient) have the highest rate of discrimination whether it is alone or combined with other features.

6. The results of the power discrimination analysis also indicate that the combinations of the features (mean, standard deviation, mean absolute deviation, contrast2 and the energy of the horizontal gradient) have the highest rate of discrimination than the combinations of other features.

7. The *K-mean* clustering algorithm had classified the shots 98% success when classify it into five clusters, because most of the extracted sequences from each shot were classified as a member in the same shot.

5.2 Suggestions

During the development of the proposed system many suggestions brought in mined to increase the system efficiency, among these suggestions are following:

1. Enhance the shot boundaries detection criteria to be capable to detect the different type of shot boundaries.
2. Using other textural (like run-length based features) and statistical features (like co-occurrence based features).
3. Extracts other features from the video's frames (like motion, shape).
4. Using neural network approach to improve the video discrimination capability.
5. Study the effect of various video compression techniques (like MPEG) on the classification of the video.

Chapter Four
Experimental Results

4.1 Introduction

This chapter is dedicating to displaying the results of the calculated tests to study the efficiency of the adopted image features to classify the video data. The results presented in this chapter are for the following system steps:

1. The classification of static and dynamic blocks of the videos image.
2. The shots boundaries detection by using of the three detection algorithms mentioned in chapter three.
3. Statistical and textural extraction. For both static and dynamic blocks and for the three color components (Red, Blue, Green).
4. Feature analysis for video sequences.
5. Determined power discrimination for the features.
6. The k-mean clustering algorithm results.

4.2 Video Test Samples

As test materials eleven video samples have been utilized. Figure (4.1) present some frames extracted from the eleven video samples. As a prototype the results of video No.6 are given in detail in this chapter. The obtained results for other video sequences indicate similar behavior to the presented prototype results.

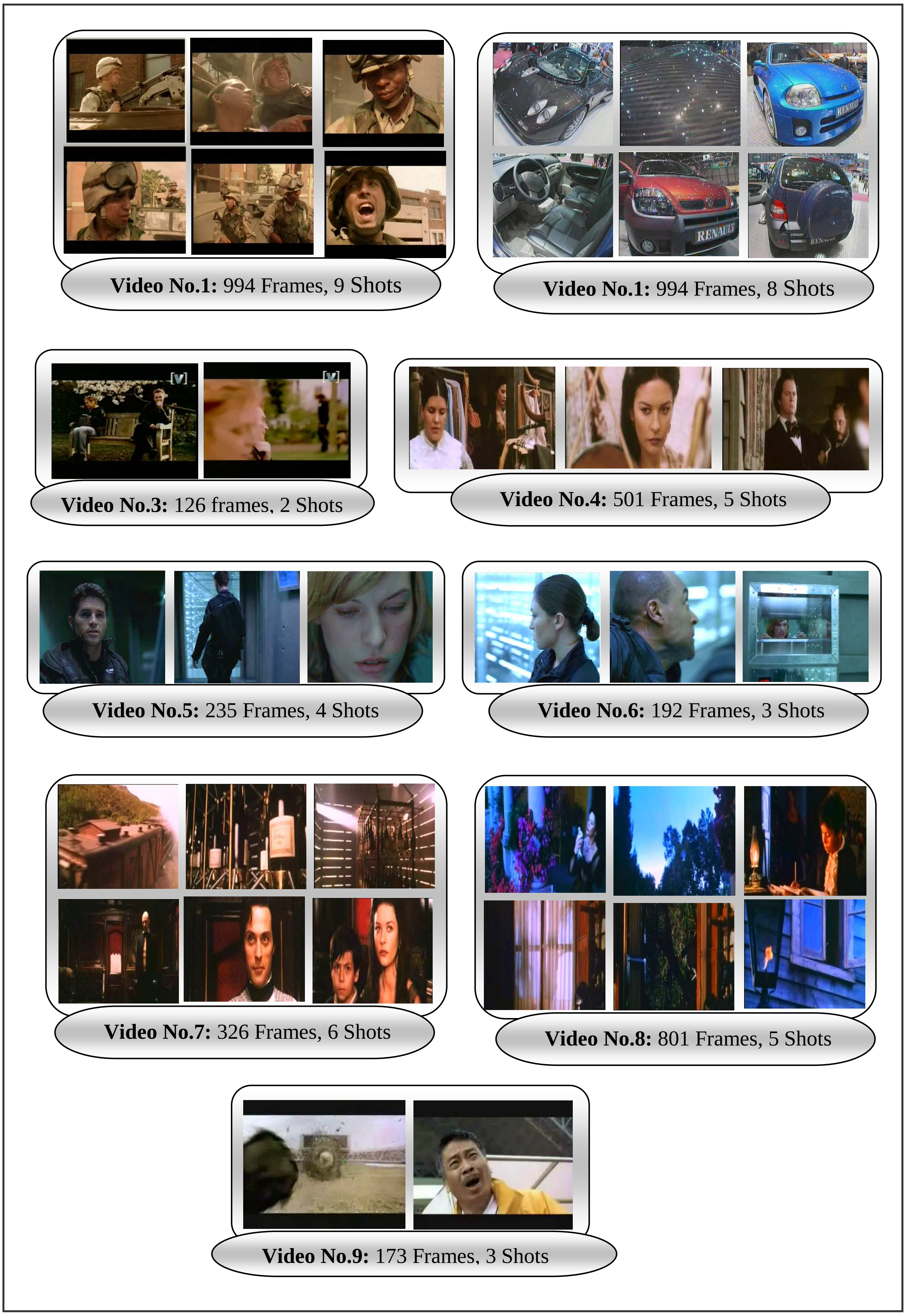
Video No.1: 994 Frames, 9 Shots
Video No.1: 994 Frames, 8 Shots
Video No.3: 126 frames, 2 Shots
Video No.4: 501 Frames, 5 Shots
Video No.5: 235 Frames, 4 Shots
Video No.6: 192 Frames, 3 Shots
Video No.7: 326 Frames, 6 Shots
Video No.8: 801 Frames, 5 Shots
Video No.9: 173 Frames, 3 Shots

Figure (4.1) Video Test Samples

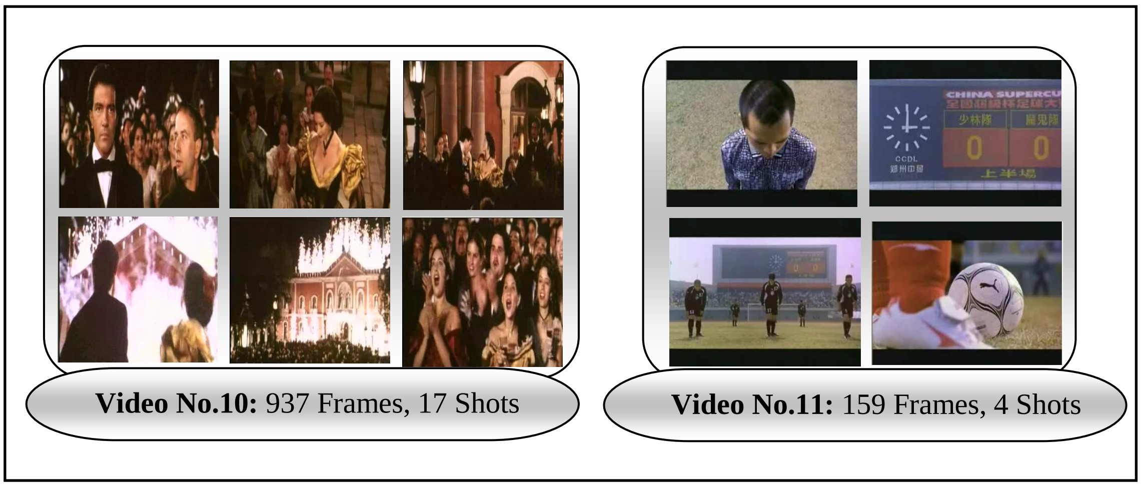
Video No.10: 937 Frames, 17 Shots
Video No.11: 159 Frames, 4 Shots

Figure (4.1) Continue

4.3 The Results of Shot Boundaries Detection

Three algorithms have been applied to detect the shots boundaries, the video samples may have different types of shot boundaries like (*fade, dissolve, wipe and cut*), and so many video sequences have been tested by using these three algorithms in order to detect the shots boundaries which is exist in these video sequences, and the test results indicate that these three algorithms can only detect the (*cut*) boundary type, so the eleven video samples that have been utilized as a test materials are all have (*cut*) boundary type.

The results of the three considered algorithms shows that the sixth video samples consists three shots, the first shot contains (147) frames the second contains (24) frames the third contains (22) frames. Table (4.1) represents the results of applying the first detection method of the shot boundaries which is based on the means of the absolute difference between the color components of the frames, see algorithm (3.2). Tables (4.2-4) show the detection results of applying the second method which is based on using the three overall means of the three difference (Absolute, square, cubic) the difference is between the mean of two successive frames, see algorithm (3.3). Tables (4.5-8) show the detection results of applying the third method which is based on using the three overall means of the three differences (absolute, square, cubic) the difference is between the mean of two successive frames (the mean is computed form the difference between two neighbor pixels in the same frame), see algorithm (3.4).

Table (4.1) The mean of the difference between the frames color components (red, blue, green)

Frame No.	Mean of Difference the frames colors (Red, Blue, Green)
1	625.83
147	75.32
148	467.29
170	70.19
171	307.20
191	77.76

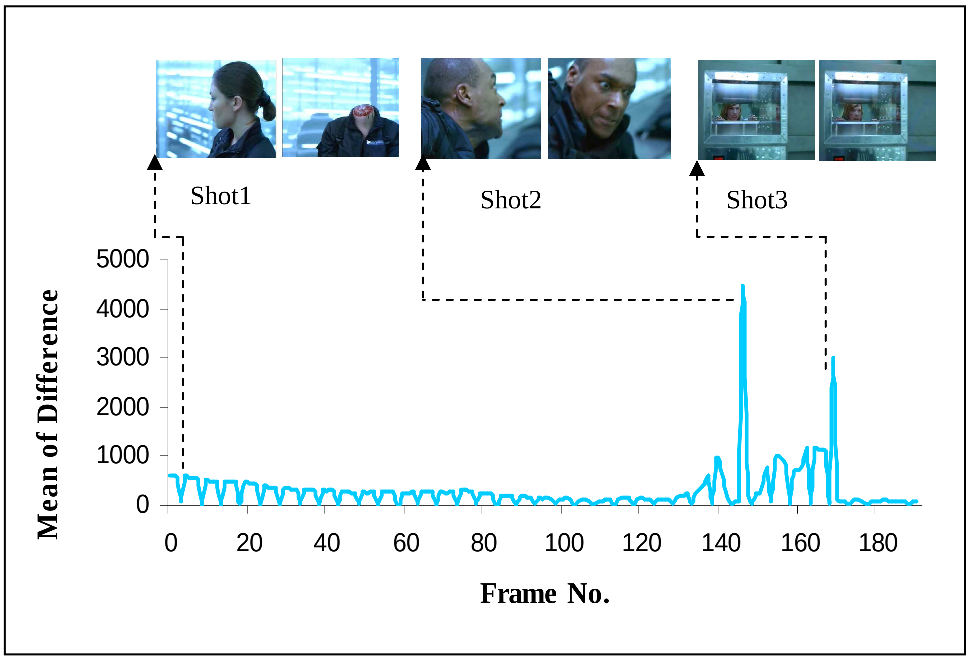

Figure (4.2) Shots detection using the mean of the difference between the frames color components (red, blue, green)

Table (4.2) The results of using the criteria of the absolute differences between the mean of two successive frames

	Absolute difference		
Frame No.	**Red**	**Blue**	**Green**
1	0.761156	0.326468	0.208116
147	0.594648	0.263563	0.435971
148	57.18723	70.56349	69.16737
170	0.152896	0.147165	0.110766
171	13.00987	33.3122	31.44362
191	0.515566	0.573765	0.51379

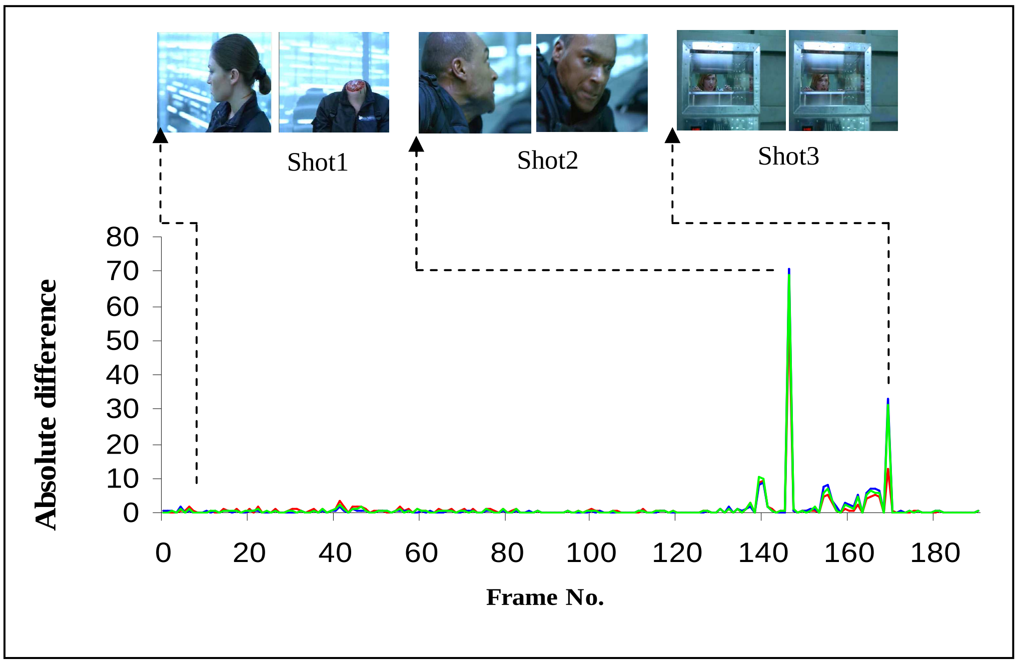

Figure (4.3) Shot detection using the criteria of the absolute differences between the mean of two successive frames

Table (4.3) the results of the using the criteria of the square differences between the mean of two successive frames

Frame No.	Square difference		
	Red	Blue	Green
1	0.579	0.107	0.043
147	0.354	0.069	0.190
148	70.380	79.206	80.125
170	0.023	0.022	0.012
171	98.257	89.703	88.701
191	0.266	0.329	0.264

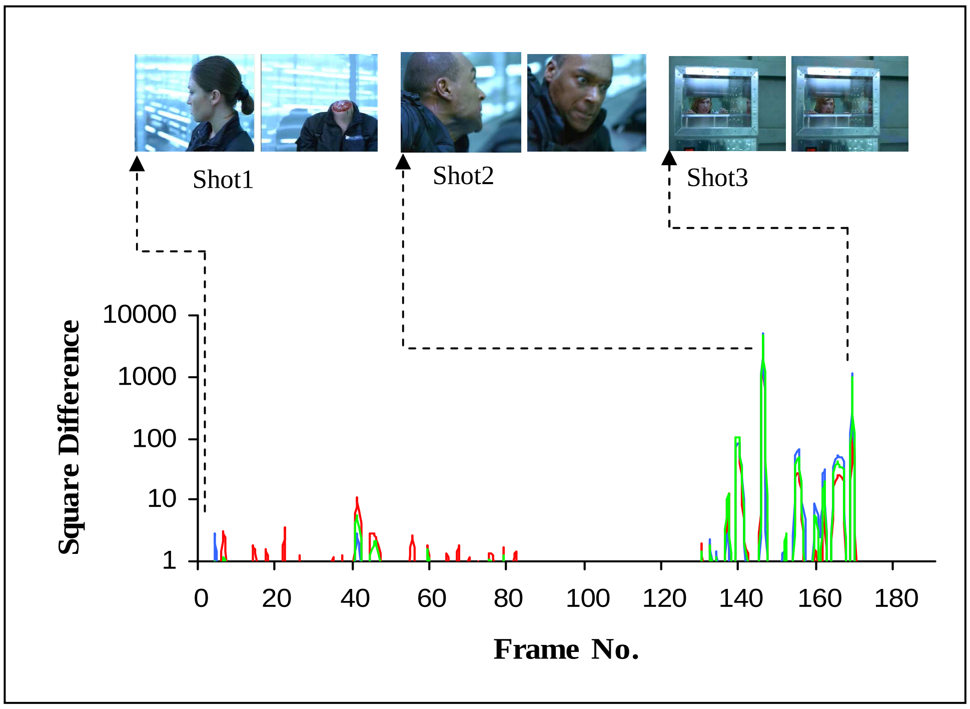

Figure (4.4) Shots detection using the criteria of the square differences between the mean of two successive frames

Table (4.4) the results of using the criteria of the cubic differences between the mean of two successive frames

Frame No.	Cubic difference		
	Red	Blue	Green
1	0.441	0.035	0.009
147	0.210	0.018	0.083
148	123.968	150.108	195.306
170	0.004	0.003	0.001
171	142.010	165.651	185.334
191	0.137	0.189	0.136

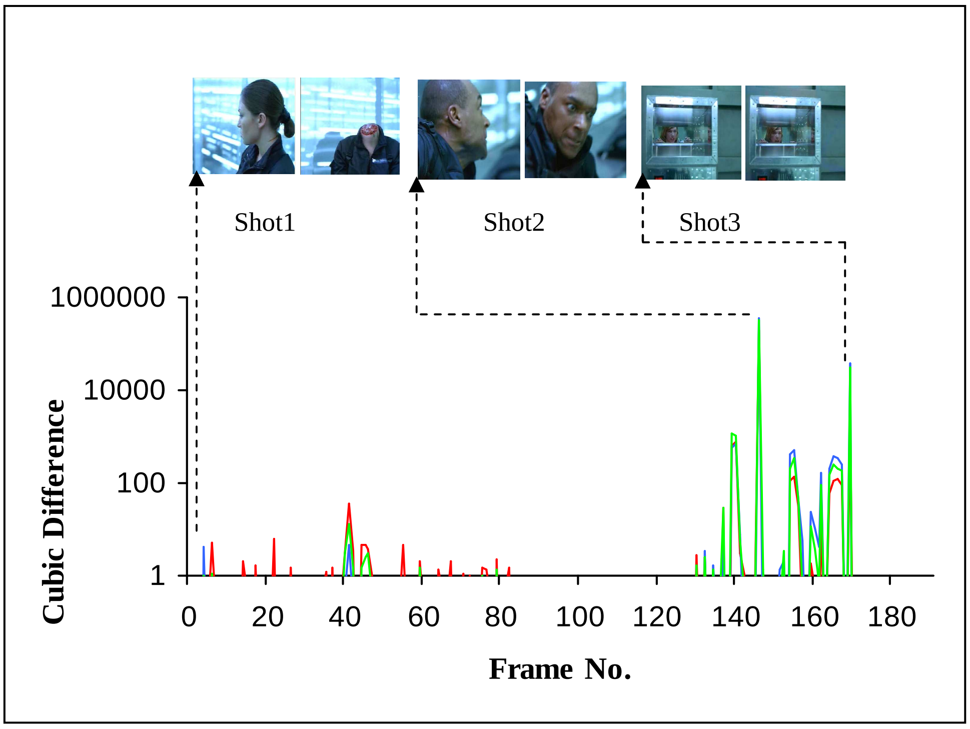

Figure (4.5) Shots detection using the criteria of the cubic differences between the mean of two successive frames

Table (4.5) the results of using the criteria of the absolute differences between the mean of the difference between two neighbored pixels in the frame

	Absolute difference		
Frame No.	**Red**	**Blue**	**Green**
1	0.0512	0.0703	0.0294
147	0.0044	0.0152	0.0103
148	0.3711	0.7790	0.4704
170	0.1793	0.1478	0.1847
171	2.4127	2.1383	2.4521
191	0.0372	0.0261	0.0061

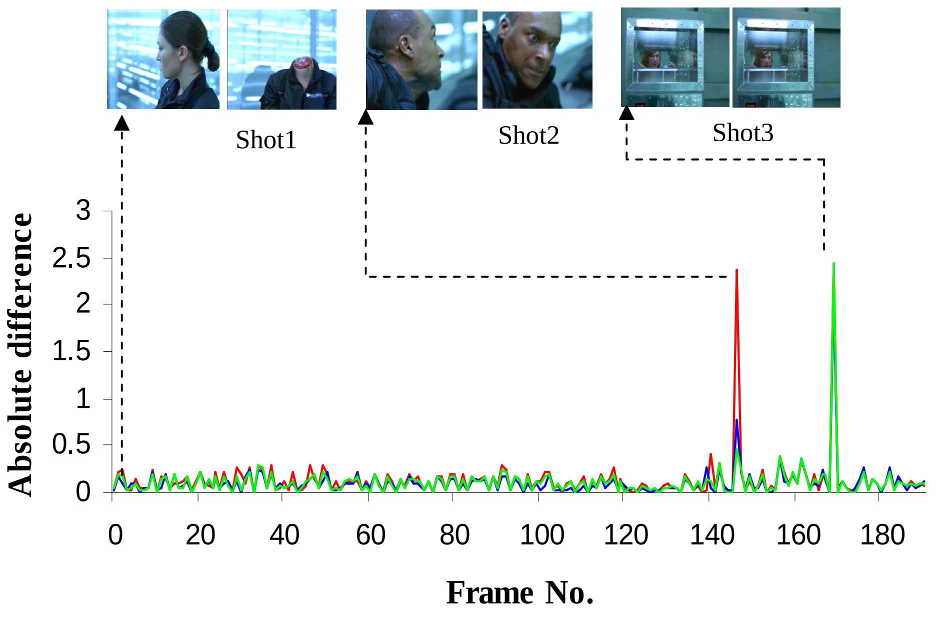

Figure (4.6) Shots detection using the criteria of the absolute differences between the mean of the difference between two neighbored pixels in the frame

Table (4.6) the results of using the criteria of the square differences between the mean of the difference between two neighbored pixels in the frame

	Square difference		
Frame No.	**Red**	**Blue**	**Green**
1	0.002624	0.004938	0.000866
147	0.000020	0.000231	0.000106
148	0.622115	0.606830	0.221279
170	0.032161	0.021853	0.034132
171	5.821236	4.572216	6.012942
191	0.001385	0.000682	0.000038

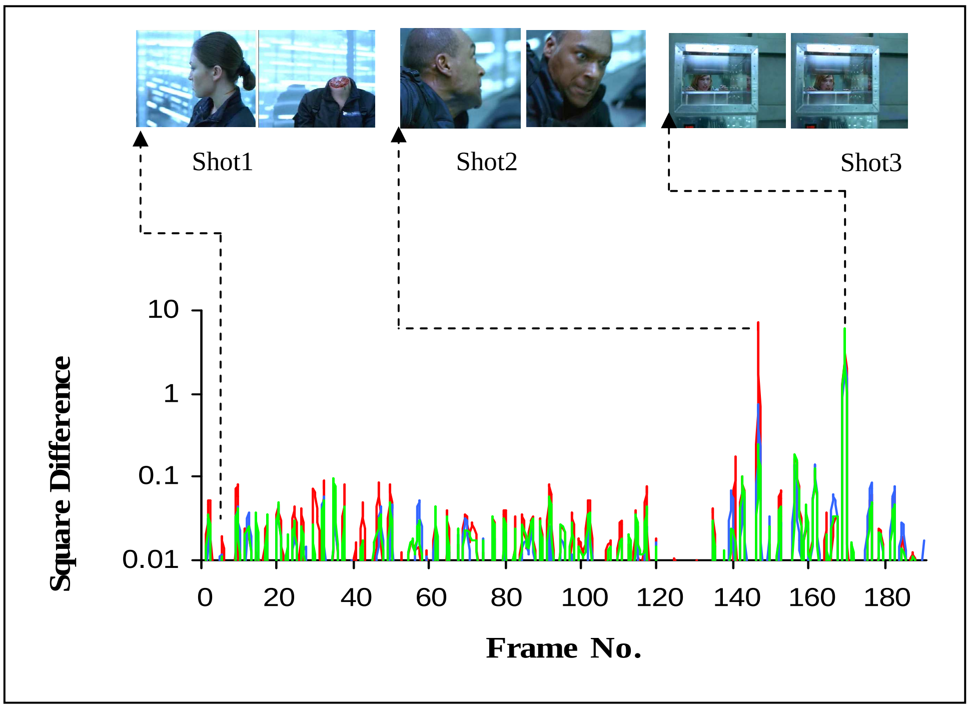

Figure (4.7) Shots detection using the criteria of the square differences between the mean of the difference between two neighbored pixels in the frame

Table (4.7) the results of using the criteria of the cubic differences between the mean of the difference between two neighbored pixels in the frame

	Cubic difference		
Frame No.	**Red**	**Blue**	**Green**
1	0.0001344	0.0003470	0.0000255
147	0.0000088	0.0000035	0.0000011
148	0.3305963	0.4727166	0.1040903
170	0.0057675	0.0032304	0.0063058
171	14.0450334	9.7766510	14.7445177
191	0.0019900	0.0015506	0.0016176

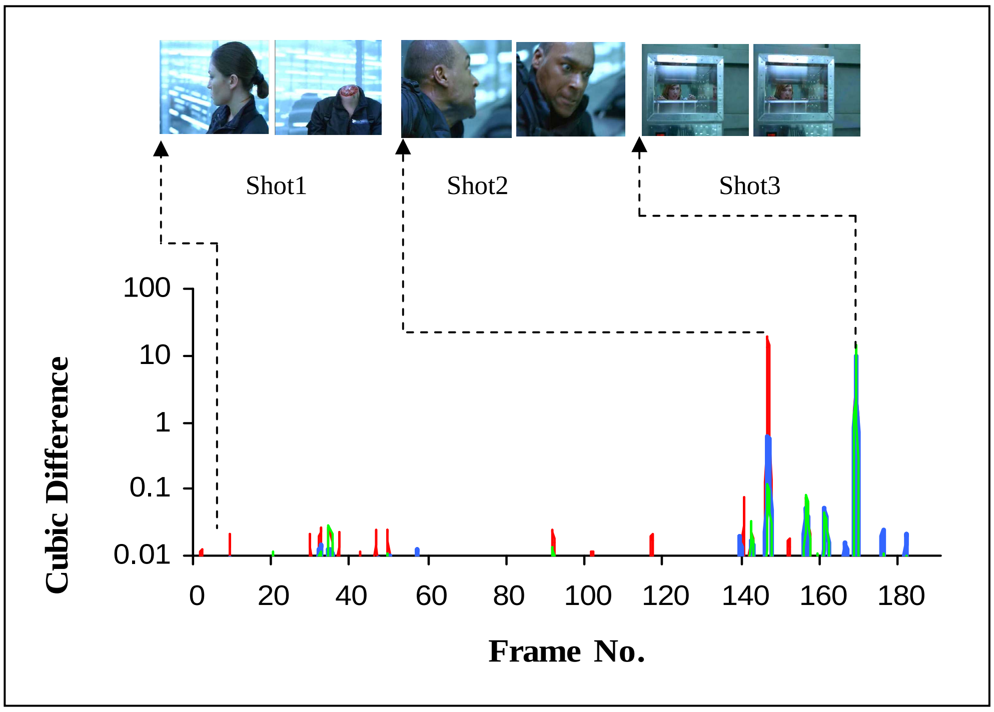

Figure (4.8) Shots detection using the criteria of the cubic differences between the mean of the difference between two neighbored pixels in the frame

4.4 The Results of Static and Dynamic Classification

Table (4.8) shows the results of the block classification for three colors (red, blue, green) for ten frames as a prototype from the sixth video sample, the value threshold was taken equal to 5 and the block size was (4×4).

Table (4.8) the results for the mean of the number of the static and dynamic blocks

Frame No.	Mean of the Static Blocks	Mean of the Dynamic Blocks
1	0.4410	0.5590
2	0.4607	0.5393
3	0.4635	0.5365
4	0.9598	0.0402
5	0.4253	0.5747
6	0.5207	0.4793
7	0.4932	0.5068
8	0.5319	0.4681
9	0.9929	0.0071
10	0.5311	0.4689

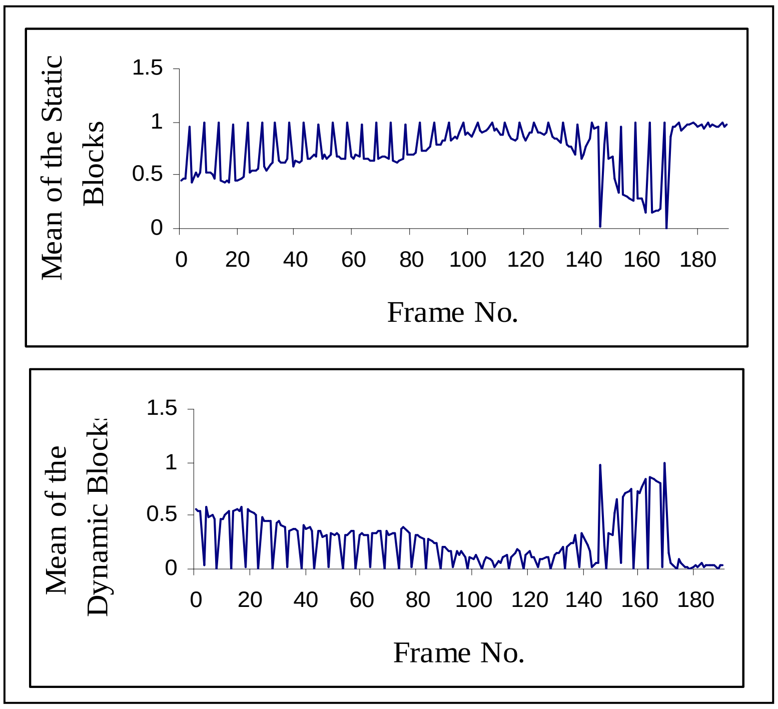

Figure (4.10) The mean of the number of the static and dynamic blocks

4.5 The Results of Features Extraction

Two types of features have been extracted from the video shots (textural, color) the color features calculated form the color histogram which is calculated for three colors components (Red, Blue, Green) from both static and dynamic blocks the following figures display the histogram behavior for the three colors components and for static and dynamic blocks.

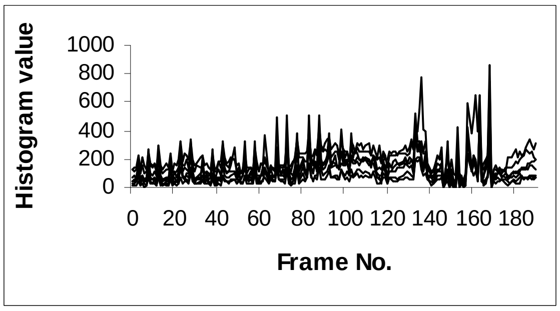

Figure (4.11) The ten histograms for the red component of the static blocks belong to the selected ten frames.

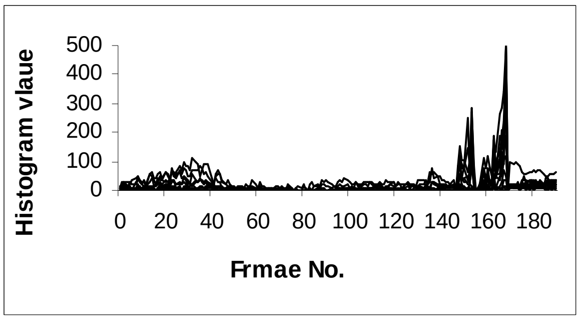

Figure (4.12) The ten histograms for the blue component of the static blocks belong to the selected ten frames.

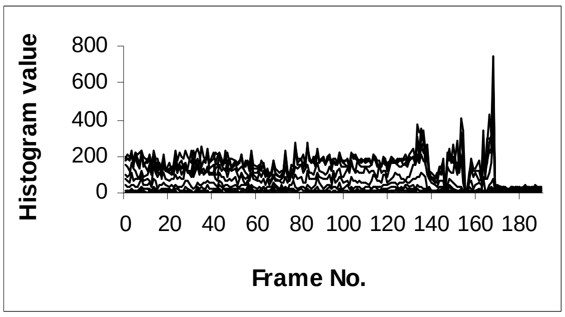

Figure (4.13) The ten histograms for the green component of the static blocks belong to the selected ten frames.

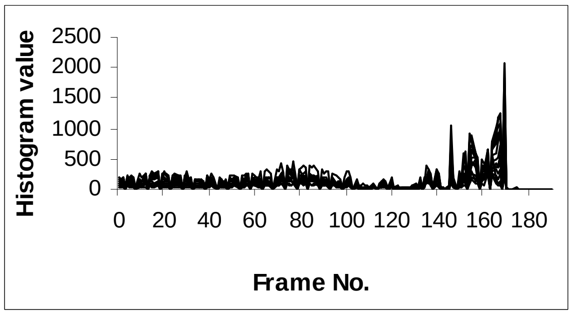

Figure (4.14) The ten histograms for the red component of the dynamic blocks belong to the selected ten frames.

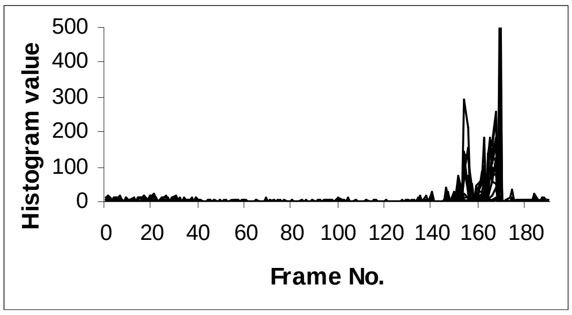

Figure (4.15) The ten histograms for the blue component of the dynamic blocks belong to the selected ten frames.

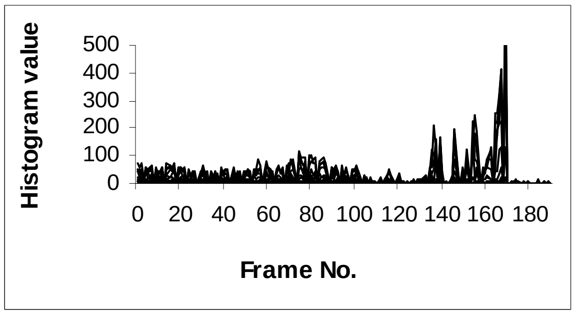

Figure (4.16) The ten histograms for the red component of the dynamic blocks belong to the selected ten frames.

First adopted type of features is the color features which are (mean, standard deviation, mean absolute deviation, skewness) theses features are extracted form both static and dynamic blocks and from the three colors components (red, blue, green) figures (4.17, 19, 21, 23) represent the results of these adopted features extracted from the static blocks and figures (4.18, 20, 22, 24) represent the results of the same features extracted from the dynamic blocks. The results are taken form the sixth video sample which contain three shots.

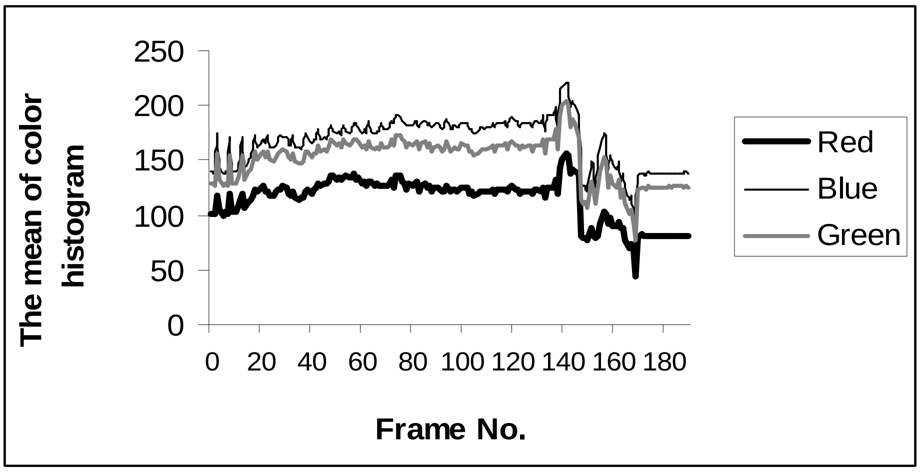

Figure (4.17) The result for mean of color histogram extracted from the static blocks for three color component (red, blue, green).

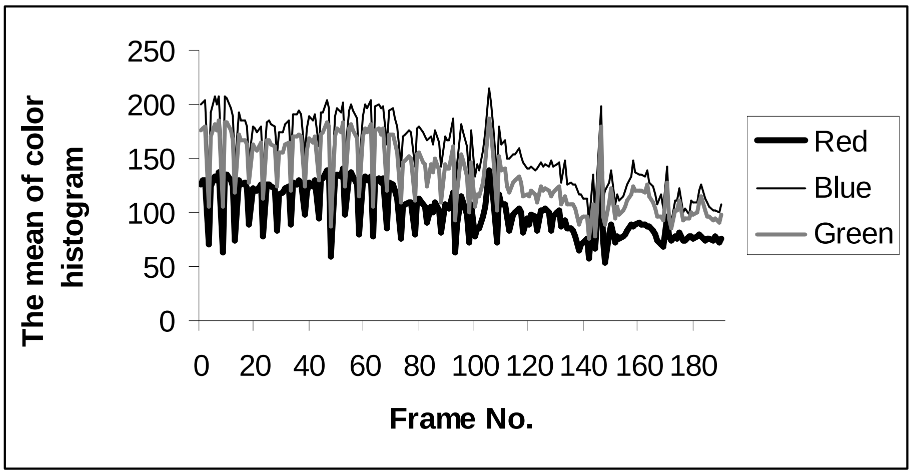

Figure (4.18) The result for mean of color histogram extracted from the dynamic blocks for three color component (red, blue, green).

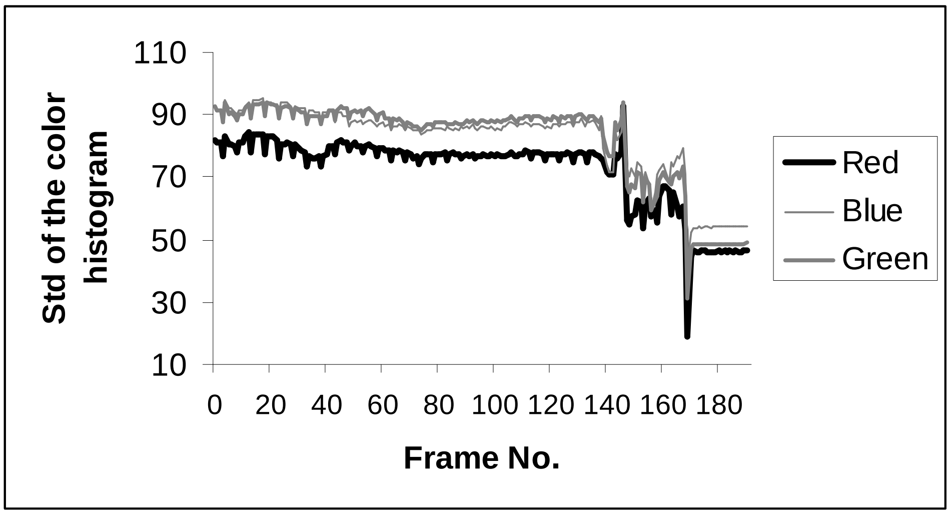

Figure (4.19) The results of the standard deviation of the color histogram extracted from the static blocks for three color component (red, blue, green).

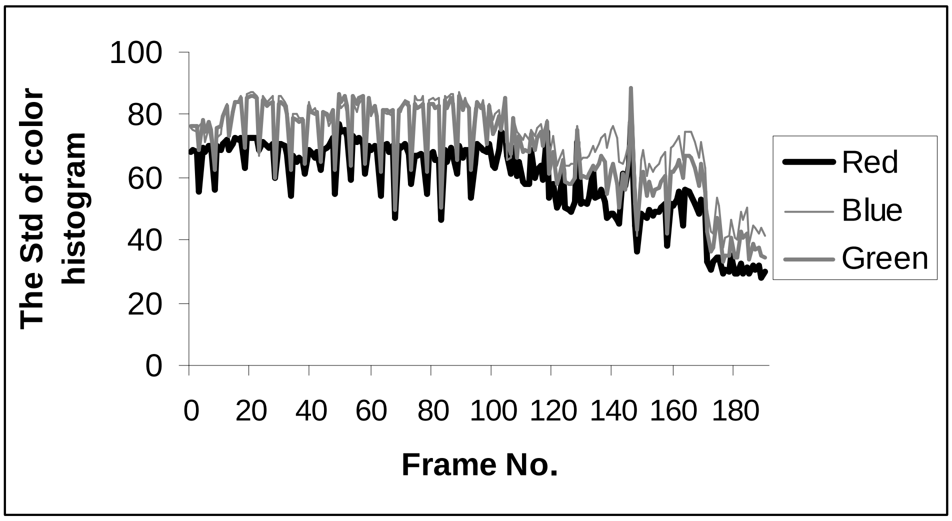

Figure (4.20) The results of the standard deviation of the color histogram extracted from the dynamic blocks for three color component (red, blue, green).

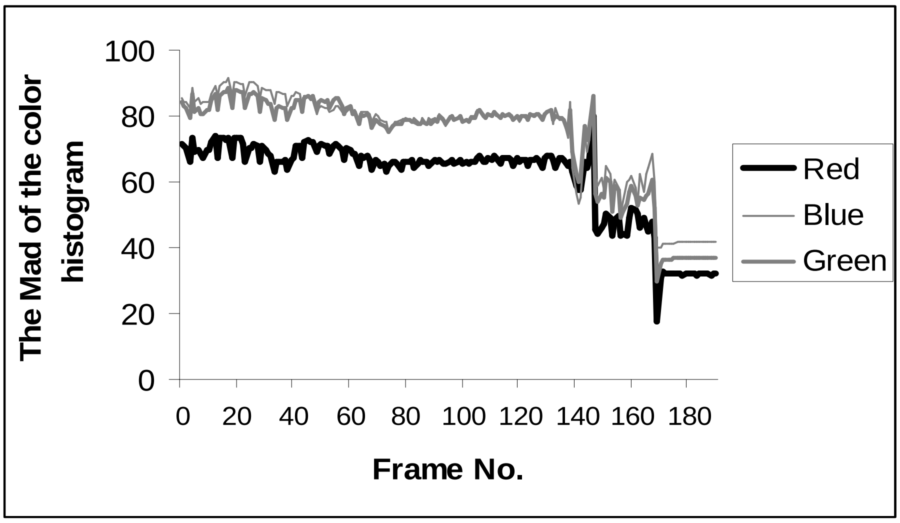

Figure (4.21) The results of mean absolute deviation of the color histogram extracted from the static blocks for three color component (red, blue, green).

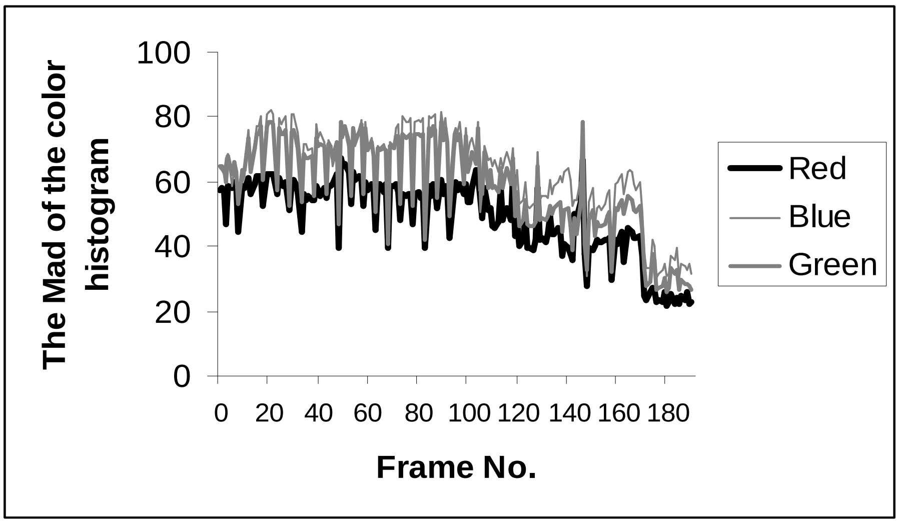

Figure (4.22) The results of mean absolute deviation of the color histogram extracted from the dynamic blocks for three color component (red, blue, green).

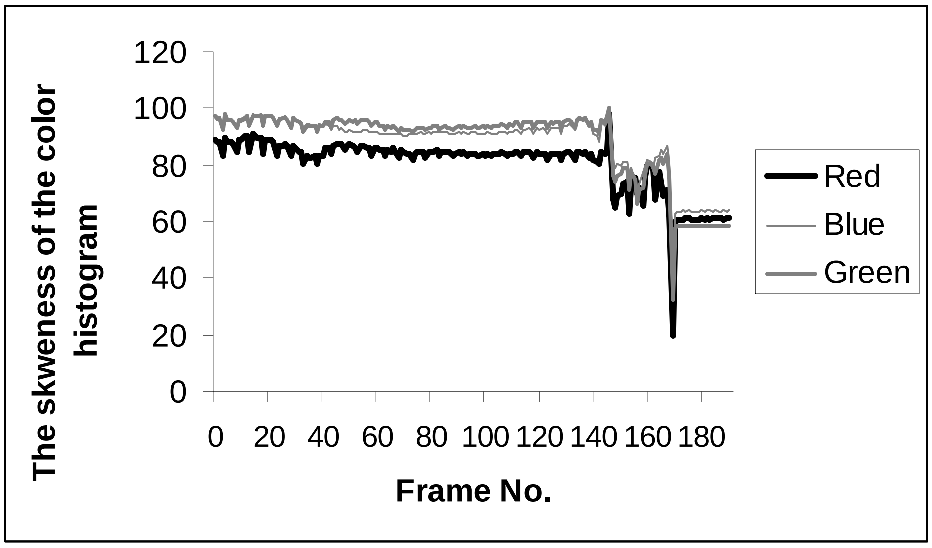

Figure (4.23) The results of skweness of the color histogram extracted from the static blocks for three color component (red, blue, green).

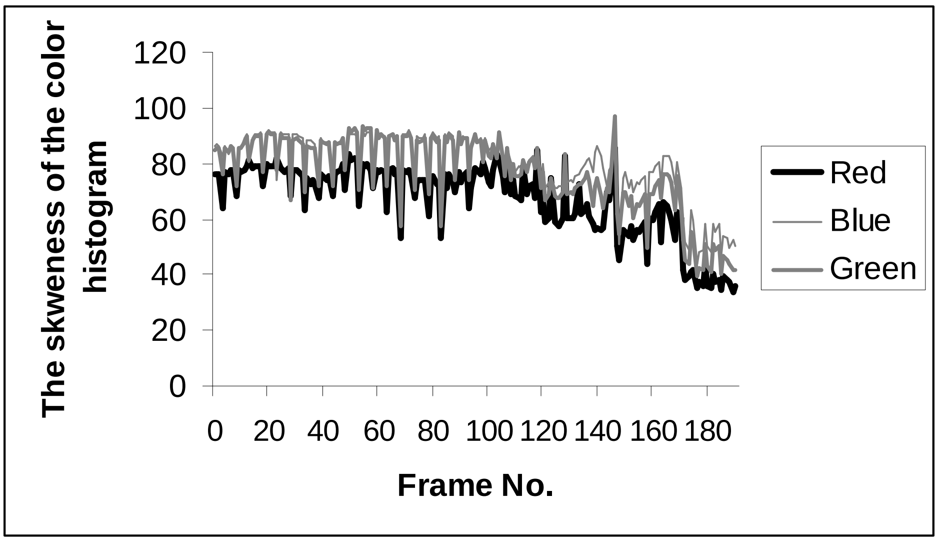

Figure (4.24) The results of skweness of the color histogram extracted from the dynamic blocks for three color component (red, blue, green).

Second adopted type of features is the textural features which are (the average energy of three gradient (horizontal, vertical, diagonal), contrast1, contrast2, modification on fractal dimension(H)) theses features are extracted form both static and dynamic blocks and from the three colors components (red, blue, green) figures (4.25, 27, 29, 31, 33, 35) represent the results of these adopted features extracted from the static blocks and figures (4.26, 28, 30, 32, 34, 36) represent the results of the same features extracted from the dynamic blocks . The results are taken form the sixth video sample which contain three shots.

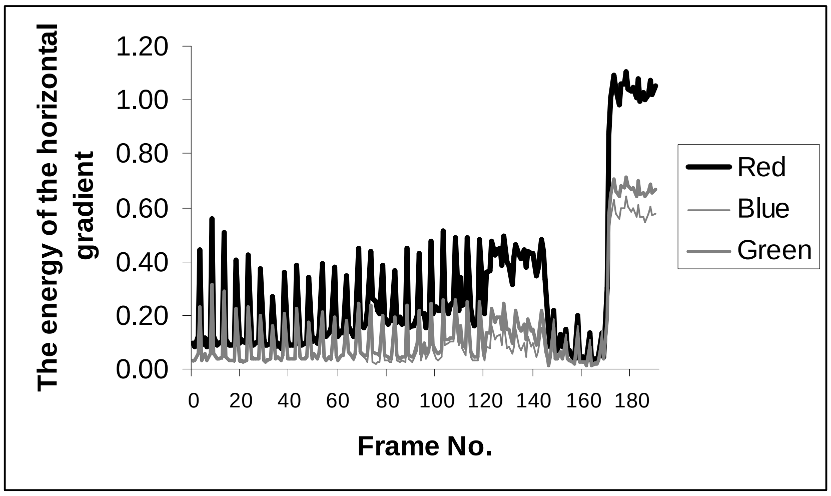

Figure (4.25) The results of energy of the horizontal gradient extracted from the static blocks for three color component (red, blue, green).

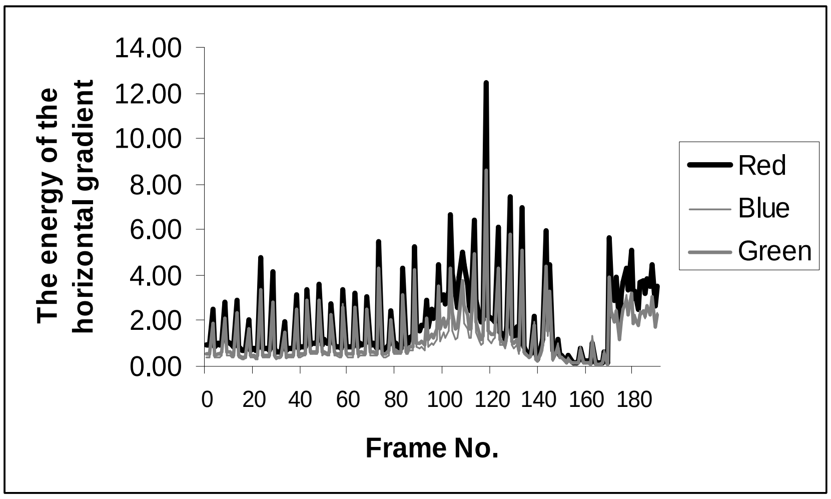

Figure (4.26) The results of energy of the horizontal gradient extracted from the dynamic blocks for three color component (red, blue, green).

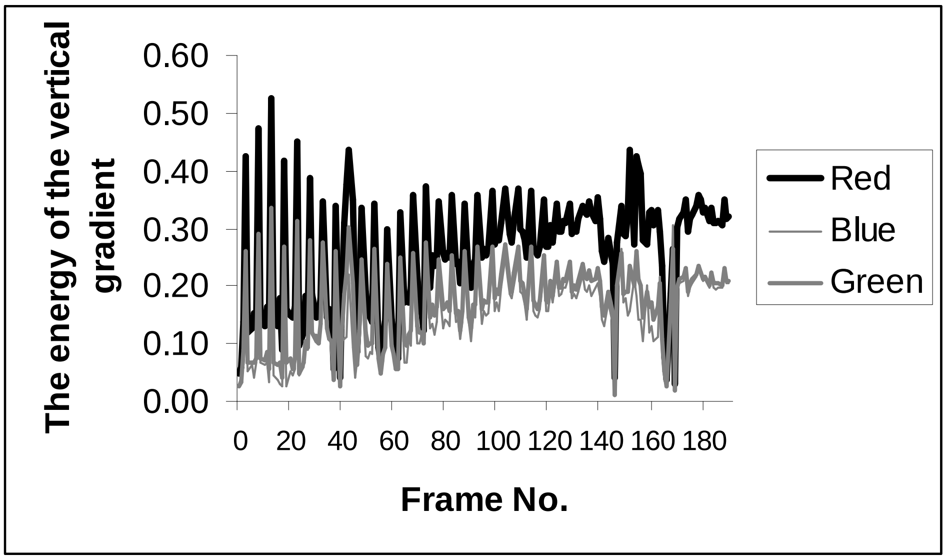

Figure (4.27) The results of energy of the vertical gradient extracted from the static blocks for three color component (red, blue, green).

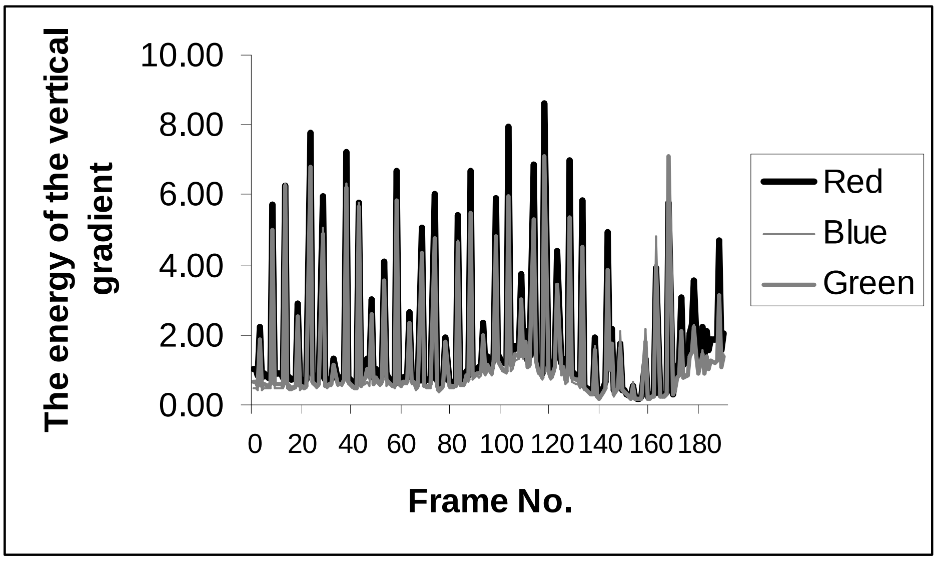

Figure (4.28) The results of energy of the vertical gradient extracted from the dynamic blocks for three color component (red, blue, green).

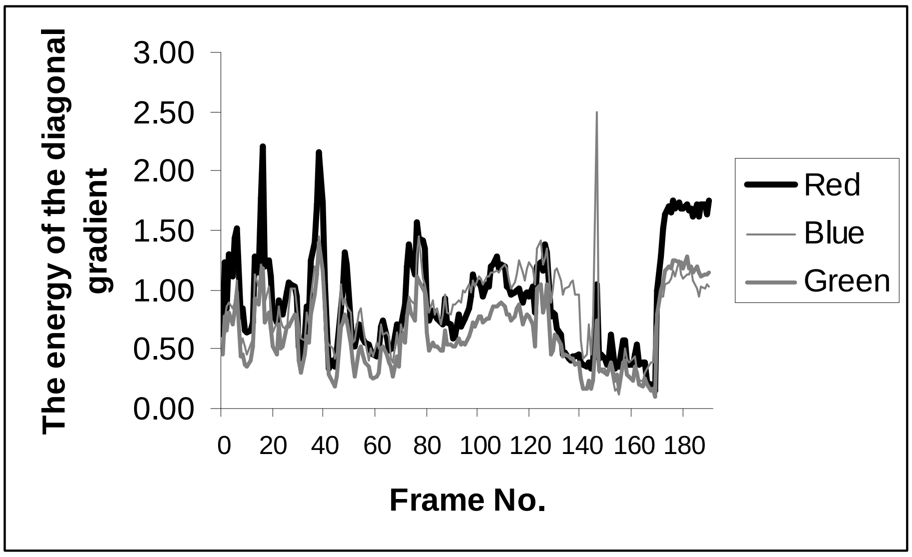

Figure (4.29) The results of energy of the diagonal gradient extracted from the static blocks for three color component (red, blue, green).

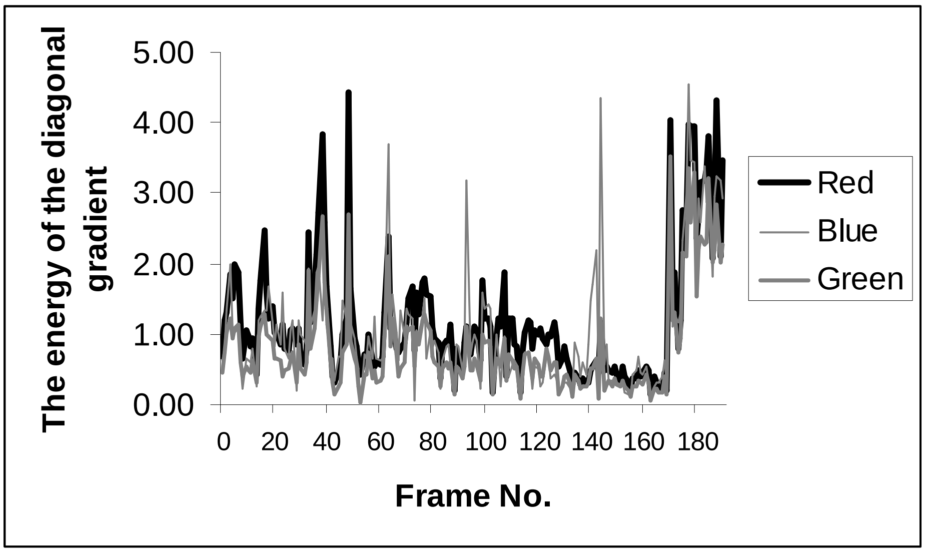

Figure (4.30) The results of energy of the diagonal gradient extracted from the dynamic blocks for three color component (red, blue, green).

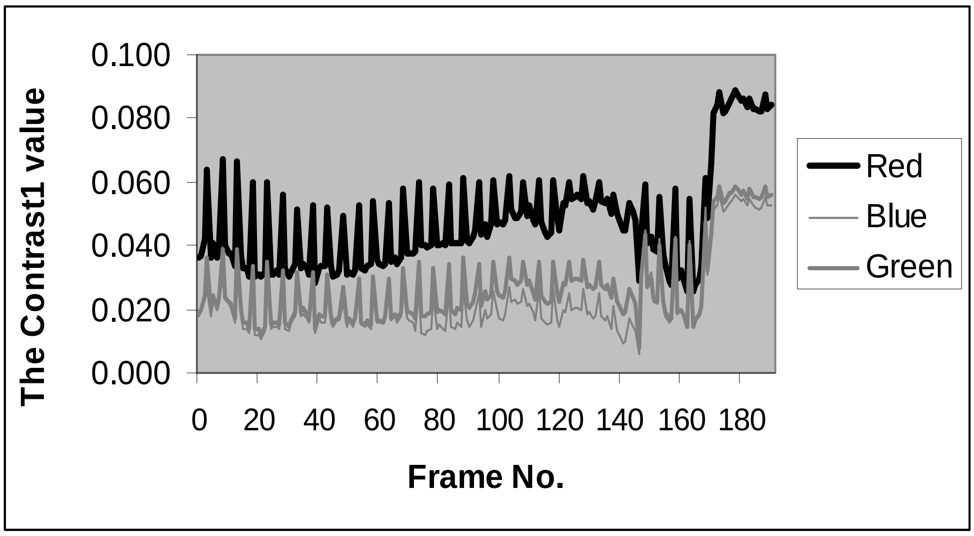

Figure (4.31) The results of contrast1 extracted from the static blocks for three color component (red, blue, green).

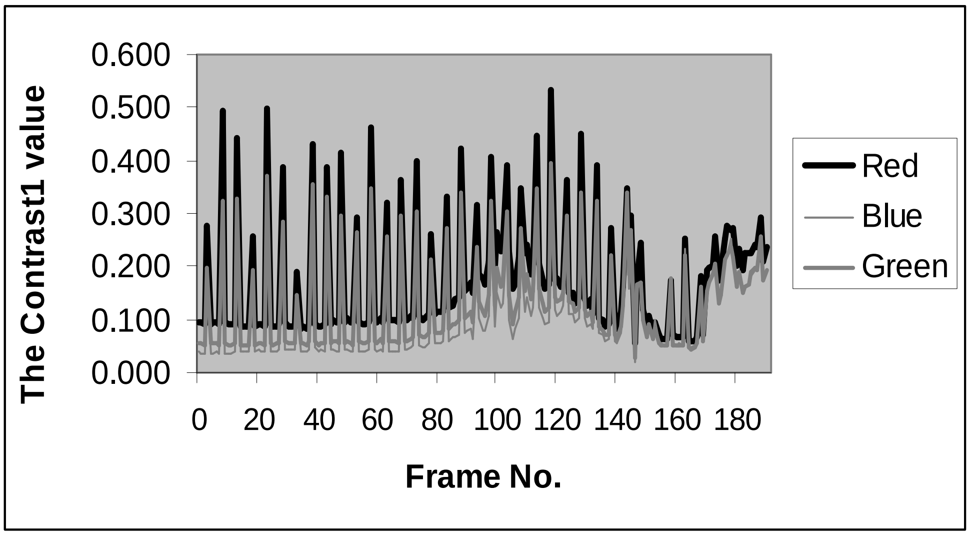

Figure (4.32) The results of contrast1 extracted from the dynamic blocks for three color component (red, blue, green).

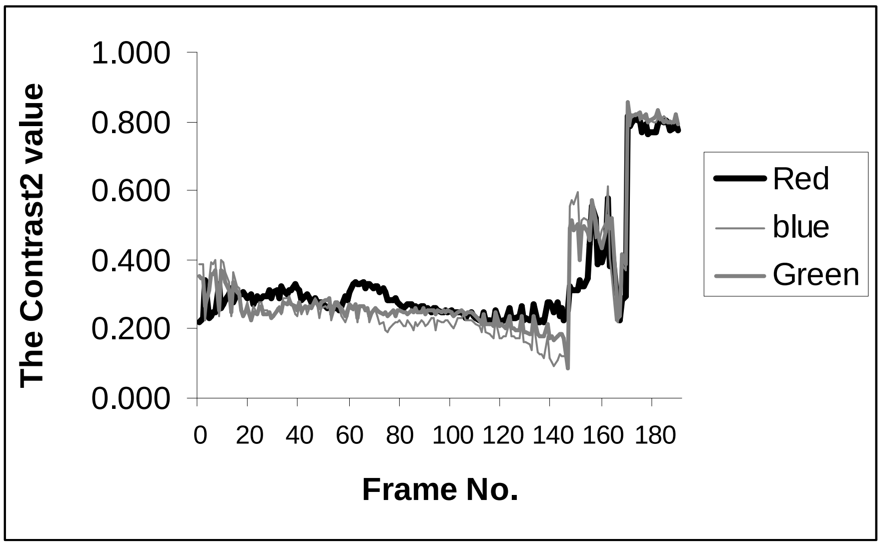

Figure (4.33) The results of contrast2 extracted from the static blocks for three color component (red, blue, green).

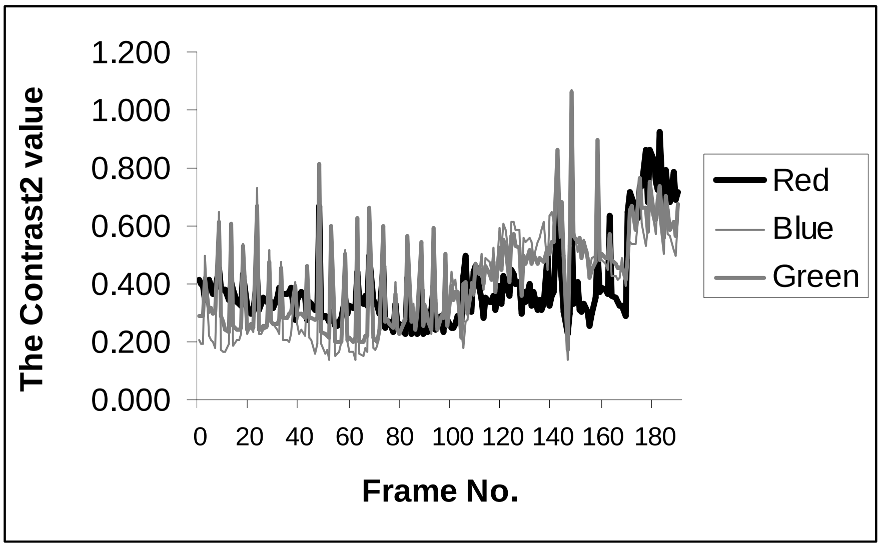

Figure (4.34) The results of contrast2 extracted from the dynamic blocks for three color component (red, blue, green).

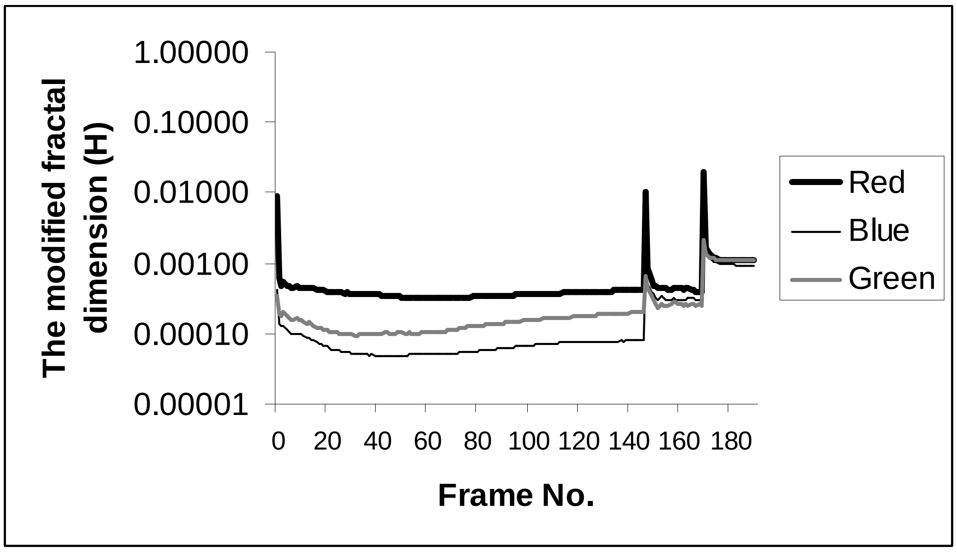

Figure (4.35) The results of modified fractal dimension (H) extracted from the static blocks for three color component (red, blue, green).

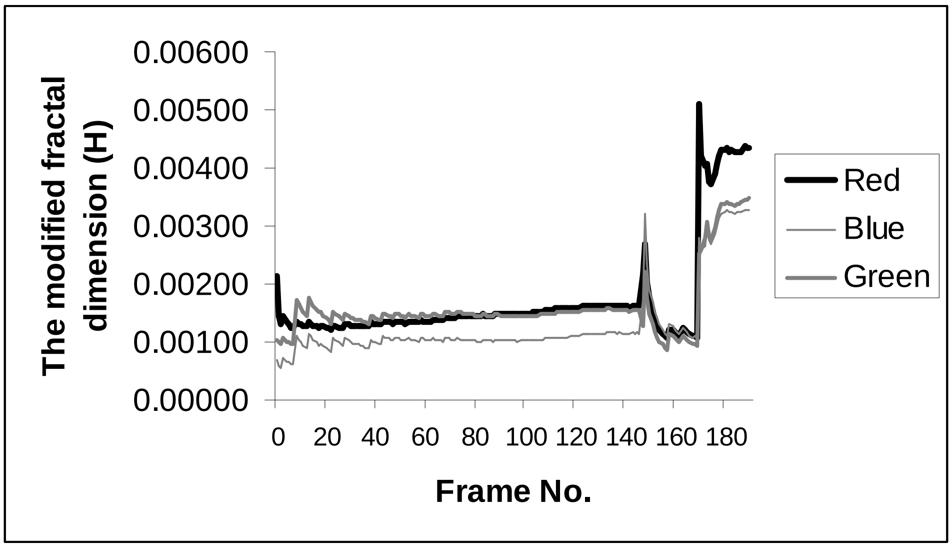

Figure (4.36) The results of modified fractal dimension (H) extracted from the dynamic blocks for three color component (red, blue, green).

4.6 The Results of Feature Analysis

The above ten features have been calculated for all the frames belong to each shot in the eleven tested videos, for each shot in the tested videos the mean of these ten features were determined, then ten video sequences have been taken from each shot and the same ten features have been extracted from each video sequence, the number of frames belongs to each sequence is depending on how many frames in the shot.

A set of tests were conducted to study the effect of the number of the video frames (used to represent each video shot) on the recognition results indicate the increase in the number of frames will improve the recognition efficiency of the adopted features, so each video's shot consist of less than 45 frame is neglected, for this reason in the sixth video sample only first shot was taken because the two other shots have a small number of frames. Table (4.9) show the results of Ten values as a prototype the values is for the mean of the adopted ten features calculated for three randomly selected sequences belong to the first shot.

4.7 The Results of Power Discrimination

Tables (4.10-14) display the rate of the recognition success and frailer of the adopted ten features. As seen from the presented results the first ninth features (mean, standard deviation, skewness, mean absolute deviation, energy of gradient, contrast1, contrast2) show high successful rates than other features; all the obtained results refer to indicate that all the static features except the (*modified fractal dimension*) have higher successful recognition rate than the dynamic features.

Table (4.9) The mean for the ten adopted features calculated for three randomly selected sequences that belong to shot No.1 (from the sixth video sample).

	Dynamic			Static			
Features	**Red**	**Blue**	**Green**	**Red**	**Blue**	**Green**	
Mean	120.20871	192.20363	168.30032	113.02430	155.67870	142.72464	sequence No.1
Std	83.97413	95.22082	94.08603	70.98245	77.88278	78.43591	
Mad	72.56188	88.85220	85.85184	59.41537	66.96887	66.87877	
Skewness	91.43575	99.40249	99.26940	79.98340	87.86890	87.43938	
H-Energy	1.51100	0.83060	1.02048	0.21805	0.09710	0.11462	
V-Energy	2.23824	1.80894	1.91750	0.24298	0.10861	0.13780	
D-Energy	1.02420	0.61878	0.58675	0.92054	0.70582	0.57362	
Contrast-1	0.18613	0.09885	0.12164	0.04489	0.02295	0.02593	
Contrast-2	0.40323	0.35704	0.37254	0.29327	0.29230	0.33916	
H	0.00191	0.00134	0.00210	0.00150	0.00014	0.00017	
Mean	136.70397	195.24983	171.67958	171.09918	187.10038	125.78592	sequence No.2
Std	83.94187	93.25120	95.17006	71.57174	82.98902	82.47166	
Mad	72.88498	87.13978	86.56016	60.13697	72.96849	71.00647	
Skewness	91.29477	98.33491	100.81112	80.22058	92.22312	91.31607	
H-Energy	1.58371	1.00762	1.10951	0.21605	0.08205	0.09376	
V-Energy	1.91067	1.58760	1.63680	0.19290	0.11702	0.13226	
D-Energy	1.09999	1.31977	0.85354	0.62953	0.62599	0.42133	
Contrast-1	0.19413	0.11010	0.13499	0.04516	0.02105	0.02360	
Contrast-2	0.37167	0.27818	0.32410	0.33553	0.27735	0.28534	
H	0.00247	0.00275	0.00291	0.00225	0.00012	0.00027	
Mean	137.82109	198.45694	182.62041	111.25058	164.79339	138.60219	sequence No.3
Std	86.40482	97.26434	99.64530	61.12988	75.35320	70.24007	
Mad	74.64823	89.76535	89.69606	49.26038	63.58346	57.18453	
Skewness	94.01401	103.78550	106.13080	71.10556	83.79155	80.55727	
H-Energy	2.40583	1.45500	1.73911	0.41121	0.10847	0.16685	
V-Energy	1.73967	1.22726	1.41940	0.32033	0.20031	0.21742	
D-Energy	1.19700	1.31719	0.88988	1.11768	0.60739	0.66620	
Contrast-1	0.20483	0.13836	0.16873	0.05963	0.02251	0.03154	
Contrast-2	0.44230	0.60149	0.55002	0.27065	0.21431	0.24116	
H	0.00278	0.00196	0.00220	0.00277	0.00017	0.00044	

Table (4.10) The best five values of the discrimination power of each feature alone.

Feature No.	Successful Rate %	Failure Rate %
9	64.79	35.21
5	51.04	48.96
1	49.38	50.62
3	47.08	52.92
2	43.75	56.25

Table (4.21) The best discrimination power values (≥%80) of the combination of two features.

Feature No.		Successful Rate %	Failure Rate %
5	9	85.83	14.17
2	7	83.54	16.46
8	9	82.08	17.92
6	9	81.67	18.33
3	7	81.25	18.75
1	9	81.04	18.96
3	5	80.83	19.17
1	5	80.00	20.00
3	9	80.00	20.00

Table (4.12) The best discrimination power values (≥%91) of the combination of three features.

Feature No.			Successful Rate %	Failure Rate %
3	5	9	94.79	5.21
2	5	9	93.13	6.87
5	6	9	92.92	7.08
3	7	9	92.50	7.50
5	8	9	92.50	7.50
1	5	6	91.88	8.12
1	6	9	91.46	8.54
2	7	9	91.46	8.54
1	5	9	91.04	8.96
1	7	9	91.04	8.96

Table (4.13) The best discrimination power values (≥%94) of the combination of four features.

Feature No.				Successful Rate %	Failure Rate %
3	**5**	**8**	**9**	96.04	3.96
3	**5**	**7**	**9**	95.83	4.17
1	**6**	**7**	**9**	95.42	4.58
1	**5**	**7**	**9**	95.21	4.79
3	**5**	**6**	**9**	95.21	4.79
2	**5**	**8**	**9**	95.00	5.00
5	**6**	**8**	**9**	95.00	5.00
3	**7**	**8**	**9**	94.79	5.21
1	**5**	**6**	**7**	94.17	5.83
1	**5**	**8**	**9**	94.17	5.83
1	**7**	**8**	**9**	94.17	5.83

Table (4.14) The best discrimination power values (≥%95) of the combination of five features.

Feature No.					Successful Rate %	Failure Rate %
1	**3**	**5**	**8**	**9**	96.04	3.96
2	**3**	**5**	**8**	**9**	96.04	3.96
1	**3**	**5**	**7**	**9**	95.83	4.17
2	**3**	**5**	**7**	**9**	95.83	4.17
1	**3**	**5**	**6**	**9**	95.21	4.79
2	**3**	**5**	**6**	**9**	95.21	4.79
1	**2**	**5**	**8**	**9**	95.00	5.00
1	**5**	**6**	**8**	**9**	95.00	5.00
2	**5**	**6**	**8**	**9**	95.00	5.00
3	**5**	**6**	**8**	**9**	95.00	5.00
4	**5**	**6**	**8**	**9**	95.00	5.00

4.8 The Results of K-Means Algorithm

Table (4.15) present the results of applying K-means clustering method, the k-mean method used the features that give a high recognition success rate, these features are the first nine features for the three color components (red, blue, green) in the features vector (i.e., mean, standard deviation, skewness, mean absolute deviation, energy of gradient, contrast1, contrast2), they are the features extracted from the static blocks except the (modified fractal dimension), the number of training video sequences is 490 patterns (i.e. ten sequences are randomly selected from each of the 49 segmented shots, which are in turn gathered from the eleven tested video samples).

The number of clusters that used in the K-means algorithm is five clusters, because the shots in the considered video samples can be categorized into five types, depending on the events inside the shots.

The results given in table (4.15) shows that most of the ten sequences that belong to certain shot were clustered within one cluster, except the sequences belong to shots (6, 12, 41, 43, 44) where %80 of the sequences belong to each of these shots have been clustered in the same cluster.

Table (4.16) shows the classification results when the number of clusters was taken four. It is obvious that the failure in classification rate was increase.

Table (4.15) The results of K-means algorithm for five clusters.

Shot No.	Class No. 1	Class No. 2	Class No. 3	Class No. 4	Class No.5
1	0	0	0	100%	0
2	100%	0	0	0	0
3	100%	0	0	0	0
4	100%	0	0	0	0
5	0	0	100%	0	0
6	20%	0	0	80%	0
7	100%	0	0	0	0
8	100%	0	0	0	0
9	100%	0	0	0	0
10	100%	0	0	0	0
11	0	0	100%	0	0
12	10%	0	0	90%	0
13	0	0	100%	0	0
14	0	0	0	100%	0
15	0	0	100%	0	0
16	0	0	100%	0	0
17	0	0	100%	0	0
18	0	0	100%	0	0
19	100%	0	0	0	0
20	0	0	100%	0	0
21	0	0	100%	0	0
22	0	0	100%	0	0
23	0	0	100%	0	0
24	0	0	100%	0	0
25	100%	0	0	0	0
26	0	0	0	100%	0
27	0	100%	0	0	0
28	100%	0	0	0	0
29	100%	0	0	0	0
30	100%	0	0	0	0
31	100%	0	0	0	0
32	100%	0	0	0	0
33	0	100%	0	0	0
34	0	100%	0	0	0
35	0	100%	0	0	0
36	0	100%	0	0	0
37	0	0	100%	0	0
38	0	0	0	0	100%
39	0	0	100%	0	0
40	0	0	0	100%	0
41	20%	0	0	80%	0
42	0	0	0	0	100%
43	20%	0	80%	0	0
44	30%	0	70%	0	0
45	0	0	100%	0	0
46	0	0	100%	0	0
47	0	0	0	100%	0
48	0	0	0	0	100%
49	0	0	0	100%	0
Overall successful	98%	100%	99%	99%	100%

Table (4.16) The results of K-means algorithm for four clusters.

Shot No.	Class No. 1	Class No. 2	Class No. 3	Class No. 4
1	100%	0	0	0
2	0	100%	0	0
3	0	100%	0	0
4	0	100%	0	0
5	0	0	0	100%
6	50%	50%	0	0
7	0	100%	0	0
8	0	100%	0	0
9	0	100%	0	0
10	0	100%	0	0
11	0	0	0	100%
12	70%	30%	0	0
13	0	0	0	100%
14	100%	0	0	0
15	0	0	0	100%
16	0	0	0	100%
17	0	0	0	100%
18	0	0	0	100%
19	0	100%	0	0
20	0	0	0	100%
21	0	0	0	100%
22	0	0	0	100%
23	0	0	0	100%
24	0	0	0	100%
25	0	100%	0	0
26	100%	0	0	0
27	0	0	100%	0
28	0	100%	0	0
29	0	100%	0	0
30	0	100%	0	0
31	0	100%	0	0
32	0	100%	0	0
33	0	0	100%	0
34	0	0	100%	0
35	0	0	100%	0
36	0	0	100%	0
37	0	0	0	100%
38	100%	0	0	0
39	0	0	0	100%
40	100%	0	0	0
41	80%	20%	0	0
42	40%	0	0	60%
43	0	20%	0	80%
44	0	30%	0	70%
45	0	0	0	100%
46	0	0	0	100%
47	100%	0	0	0
48	80%	0	0	20%
49	100%	0	0	0
Overall successful	96%	97%	100%	97%

Chapter One
General Introduction

1.1 Data Mining

Substantial progress in the field of data mining research has been witnessed in the last few years [Han98]. The Data mining DM (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the identified relationships. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Data might be one of the most valuable assets of your corporation, but only if you know how to reveal valuable knowledge hidden in raw data. Data mining allows you to extract diamonds of knowledge from your historical data and predict outcomes of future situations. It will help you optimize your business decisions, increase the value of each customer and communication, and improve satisfaction of customer with your services **[**Data05].

Data mining software analyzes relationships and patterns in stored transaction data based on open-ended user queries. Several types of analytical software are available: statistical, machine learning, and neural networks.

Generally, four types of relationships are sought [What01]:

1. **Classes**: Stored data is used to locate data in predetermined groups. For example, a restaurant chain could mine customer purchase data to

determine when customers visit the restaurant and what they typically order? This information could be used to increase traffic by having daily specials.

2. **Clusters**: Data items are grouped according to logical relationships or consumer preferences. For example, data can be mined to identify market segments or consumer affinities.
3. **Associations**: Data can be mined to identify associations. The beer-diaper example is an example of associative mining.
4. **Sequential patterns**: Data is mined to anticipate behavior patterns and trends. For example, an outdoor equipment retailer could predict the likelihood of a backpack being purchased based on a consumer's purchase of sleeping bags and hiking shoes.

Data mining consists of five major elements [What01]:

1. *Extract, transform, and load* transaction data onto the data warehouse system.
2. *Store and manage the data* in a multidimensional database system.
3. *Provide data access* to business analysts and information technology professionals.
4. *Analyze the data* by data mining software.
5. *Present the data* in a useful format, such as a graph or table.

1.2 Growth of Data Mining

During the last years the data mining issues have been rapidly developed, the reasons behind this development are [Data05]:

1. **Growing Data Volume:** The main reason for necessity of automated computer systems for intelligent data analysis is the enormous volume of existing and newly appearing data that require processing. The amount of data accumulated each day by various business, scientific, and governmental organizations around the world is daunting. According to information from GTE research center, only scientific organizations store each day about 1 TB (terabyte) of new information. And it is well known that academic world is by far not the leading supplier of new data. It becomes impossible for human analysts to cope with such overwhelming amounts of data.
2. **Low Cost of Machine Learning**: One additional benefit of using automated data mining systems is that this process has a much lower cost than hiring an army of highly trained professional statisticians. While data mining does not eliminate human participation in solving the task completely, it significantly simplifies the job and allows an analyst who is not professional in statistics and programming to manage the process of extracting knowledge from data**.**

1.3 Multimedia Data Mining

Multimedia data mining is a subfield of data mining that deals with the extraction of implicit knowledge, multimedia relationship, or other patterns not explicitly stored in multimedia data base [Han98].

Multimedia data (image, audio and video) has been the major focus for many researchers around the world. Many techniques for representing, storing, indexing and retrieving multimedia data have been proposed [Zaia00].

The computer industry has seen a large growth in technology access, storage and processing fields. This combined with the fact that there are a lot of data to be processed, and they paved the way for analyzing and mining data to derive potentially useful information. Various fields ranging from commercial to military want to analyze data in an efficient and fast manner.

Particularly in the area of Multimedia data, images have the stronghold. However there is a general agreement that sufficient tools are not available for analysis of images. One of the issues is the effective identification of features in the images (color, shape and texture) and the other one is extracting them. One of the difficult tasks is how to know the image domain and obtaining a priori knowledge of what information is required from the image. This is one of the reasons behind making the image mining process cannot be completely automated [Fosc01].

1.4 Literature Survey

Several researches in the multimedia data mining field had been done; the following are some of these research:

1. **Zaiane, et al [Zaia98]** they had designed and developed a multimedia data mining system prototype called (multimediaminer), the system include the construction of multimedia data cube which facilitated multiple dimensional analysis of multimedia data, the system is primarily based on the visual content, and the mined of multiple kinds of knowledge, including summarization, comparison, classification, association and clustering.

2. **Zaiane, et al [Zaia99]** they had designed and implemented MultiMediaMiner, a system prototype to mine high-level multimedia information and knowledge from large multimedia repositories like the WWW. WordNet, a semantic network for English, was used to clean and transform sets of keywords extracted from Web pages to index

multimedia objects contained in these pages. WordNet was also enriched and used to generate concept hierarchies necessary for interactive information retrieval and the construction of multi-dimensional data cubes for multimedia data mining with MultiMediaMiner.

3. **Zaïane, et al [Zaia00]** they studied methods that mined content-based associations with recurrent items and with spatial relationships that extracted from large visual data repositories. A progressive resolution refinement approach is proposed in which frequent item-sets at rough resolution levels are mined, and progressively finer resolutions are mined only on the candidate frequent item-sets derived from mining rough resolution levels.

4. **Foschi, et al [Fosc01]** they proposed system that extracted patterns and derived knowledge from large collections of images, dealt mainly with identification and extraction of unique features for particular domain. The aim was to identify the best features and thereby extract relevant information from the images. They tried various methods of extraction; the extracted features and the used extraction techniques have been evaluated. The experimental results showed that the adopted features are sufficient to identify the patterns of the images. The extracted features were evaluated for goodness and tested on tested images.

5. **Zhang, et al [Zhan01]** they had presented a research that highlighted the need for image mining in view of the rapidly growing amounts of image data, they pointed out the unique characteristics of image databases that brought a whole new set of challenging and interesting research issues to be resolved. Also they examined two frameworks for image mining: function-driven and information-driven image mining frameworks. They

discussed some techniques that are frequently used in the early works in image mining, namely, object recognition, image retrieval, image indexing, image classification and clustering, association rule mining and neural network.

6. **Oh and Bandi [Band02]** they proposed general framework for real time video data mining to be applied to the raw videos (like Traffic videos, surveillance videos, etc.). The system was designed to perform the following tasks, grouping of input frames into segments, discovering unknown knowledge and detecting interesting patterns (like motion, object, colors, etc.), the researchers focused on motion as a feature and the last task was the clustering of segments using multi-level hierarchical clustering approach.

7. **Datcu and Seidel** [**Datc02]** they presented a system for image information mining based on modeling the causalities, which link the image-signal contents to the objects and structures lay within the regions of interest for the users. The basic idea was to split the process of information representation into the stages: image feature extraction, unsupervised grouping in a large number of clusters, data reduction by parametric modeling the clusters, and supervised learning of user semantics. (Where instead of being programmed, the system is trained by a set of examples). The proposed system was a prototyped for inclusion in a new generation of intelligent satellite ground segment systems, and several other applications.

8. **Ciucu, et al [Ciuc02]** they described an application of a scale space clustering algorithm (melting) for exploration of image information content. Clustering by melting considers the feature space as a thermo

dynamical ensemble and groups the data by minimizing the free energy, using the temperature as a scale parameter. They developed clustering by melting for multidimensional data; they proposed and demonstrated a solution for the initialization of the algorithm.

9. **Oh, et al [Band03]** they extended their previous work **[Band02]**, to further address the issue such as how to mine video data, in other words how to extract previously unknown knowledge and detect interesting patterns. To extract motions, they used a criteria based on the accumulation of quantized pixel differences among all frames in a video segment. They studied how to cluster those segmented pieces by using the features: the amount and the location of motions. Also they investigated an algorithm to find whether a segment has normal or abnormal events and computes *the degree of abnormality* of a segment, which represents to what extent a segment is distant from to the existing segments.

10. **Zhu and Wu [Zhu04]** they proposed an association-based video summarization scheme that mined sequential associations from video data for summary creation. The detected shots of the video were clustered into visually distinct groups, and then constructed a sequential sequence by integrating the temporal order and cluster type of each shot. An association mining scheme is designed to mine sequentially associated clusters from the sequence, and these clusters are selected as summary candidates. The system generated the corresponding summary by selecting representative frames from candidate clusters and assembled them according to their original temporal order.

1.5 Aim of Thesis

The aim of this project is summarized by the following targets:

1. Partition the video data into static and dynamic blocks according to the level of variability of the pixels belong to each block (across the video sequences).
2. Segment the video into shots (collection of frames) and find the boundaries of each shot.
3. Extract two types of features textural and color features form the static and dynamic blocks.
4. Study the discrimination power of the extracted features (textural and color) to recognize video sequences from each other.
5. Utilize the k-mean clustering method to classify the video sequences.
6. establish the program required to perform the following tasks:

    a. Find the shots boundaries of the video.
    b. Segment each frame to static and dynamic blocks.
    c. Extract the features.
    d. Analysis the discrimination power of the adopted features set.
    e. Apply the K-Mean algorithm to classify the video sequences.

1.6 Chapters overview

In this section, the contents of the individual chapters of this thesis are briefly reviewed:

1. Chapter two "Theoretical Background" consists of all the methodology deals with the video classification, details of features extraction (color and textural features) and the clustering of the video sequences.

2. Chapter three "The Proposed System" presents the proposed system design steps, Like the detection of shots boundaries, the extraction of the features (color and textural) from the video frames, the finding of the power discrimination for these features and the use of the K-mean clustering algorithm to cluster the video sequences.

3. Chapter four "Experimental Results" this chapter display the experimental results of the work.

4. Chapter five "Conclusion and Future Work" introduce the conclusion of the experimental results of this work with som recommendation for the future work.

Chapter Three
Proposed System

3.1 Introduction

This chapter is concerned with the description of the designed and implemented video classification system. The description will include the following stages of the proposed system:

decomposition of the colors of each frame in the video, detection of the scenes changes (shots), classify the blocks of the frames into dynamic and static, then extract the features from the shots of AVI file, analyze the extracted features and find the discrimination power of each extracted features and as a last step apply the K-Means algorithm to establish the video classification system.

The programming language Microsoft Visual Basic-0.6 had been used to establish the required programs of this project.

3.2 The System Model

Figure (3.1) illustrates the phases of the proposed video. As it's shown in the figure, the system model consists of three phases.

The first phase contains the load of the video stream and decompose the colors component of each frame into three colors (Red, Green, Blue) then detect the shots boundaries, the last step is classifying the blocks of the frames into static and dynamic blocks; the output from this phase are the dynamic and the static blocks for the three colors (Red, Green, Blue) for each frame in the shot.

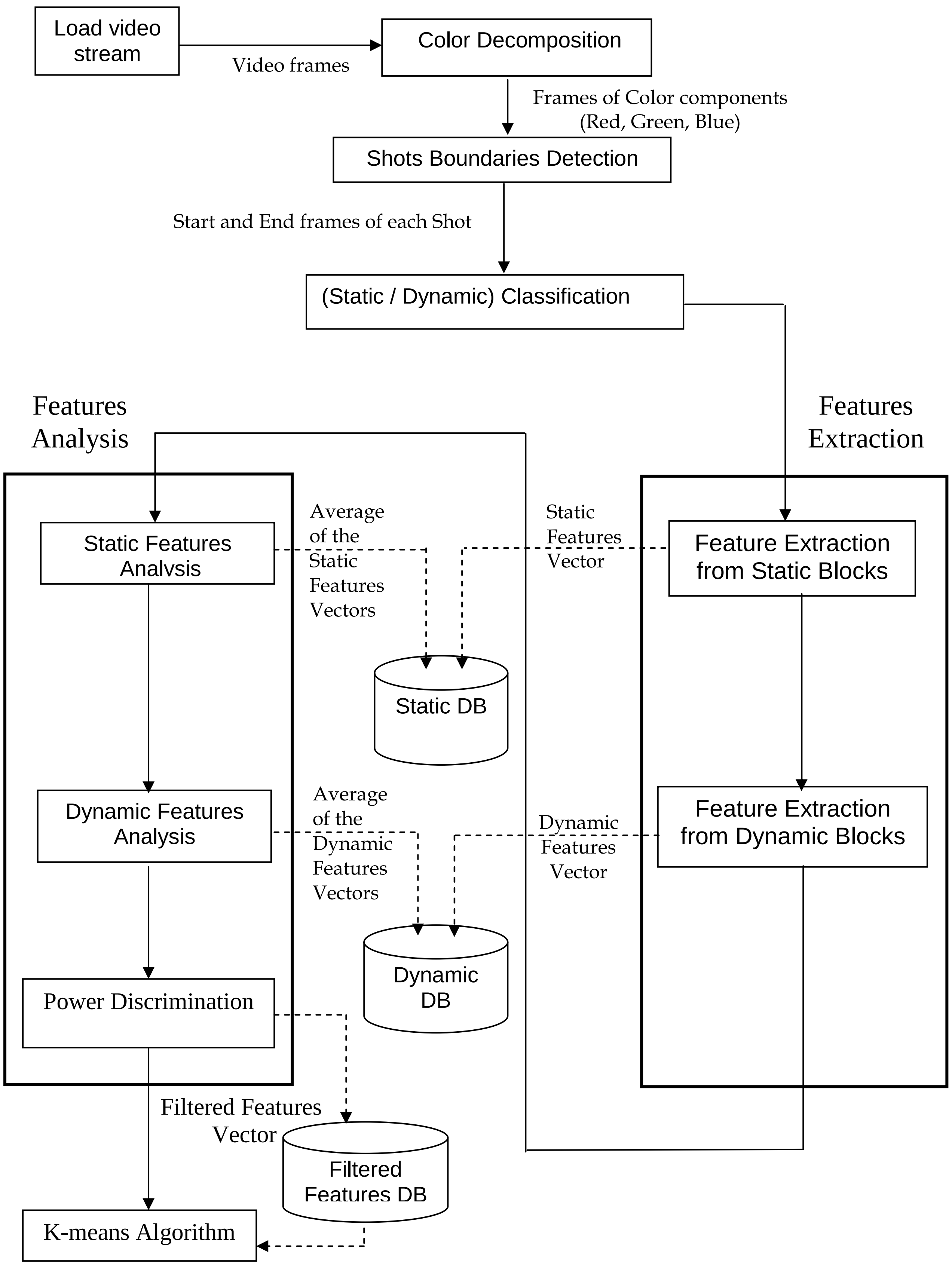
Load video stream
Video frames
Color Decomposition
Frames of Color components (Red, Green, Blue)
Shots Boundaries Detection
Start and End frames of each Shot
(Static / Dynamic) Classification
Features Analysis
Features Extraction
Static Features Analysis
Average of the Static Features Vectors
Static Features Vector
Feature Extraction from Static Blocks
Static DB
Dynamic Features Analysis
Average of the Dynamic Features Vectors
Dynamic Features Vector
Feature Extraction from Dynamic Blocks
Dynamic DB
Power Discrimination
Filtered Features Vector
Filtered Features DB
K-means Algorithm

Figure (3.1) The system model

The second phase, use the output of the first phase to extract ten features for both static and dynamic blocks for the three color components (Red, Green, Blue) belong to each shot, then analysis these features by computing the same ten features for some selected number of frames belong to each shot in the video. Then find the power discrimination for these ten features, the power was first determined for each single feature alone, then for combination of features (starting from two to five); the output is a vector of filtered features that have high discrimination power.

The third and last phase is the clustering phase; the k-means algorithm has been used to cluster the videos shots.

Eleven AVI video had been used as test sample to implement this project, and more than 49 shots had been detected, the shots that have small number of frames have been neglected, so only 49 shots have been used in our research work.

3.2.1 Load Video stream

Every AVI file contains audio stream and video stream, the audio stream contains information about the audio part of the AVI file and the video stream contains information about the video part of the AVI file (like total number of frames, frame width, frame height). In our work the audio stream had separated from the video stream by using (*VCDCutter V- 4.04*) program, then only the video stream data which had been used in the implementation of the project.

3.2.2 Color Decomposition

The video stream data which is contains a sequence of bitmap frames. The bitmap image consists of two parts header and data, each pixel in the image data have three color components (Red, Blue and Green), in code list (3.1) these three components (Red, Blue and Green) have separated and put in the buffer (Color-array).

Code List (3.1) Decompose Color Component of Video Frames

Input:

```
    Videono is the video number.
    Image  is a record contains the frame color (Red, Blue and Green) information
```

Output:

```
    Frames_file (Video No., Frame No.) is a buffer contains the array Color_array
      which holds the data of the color components (Red, Blue and Green).
```

Steps:

```
For each frame in the video Xi
    For each pixel in the image X, Y
        Color_array (Xi, 0, X, Y) ← Image (x, y).R
        Color_array (Xi, 1, X, Y) ← Image (x, y).B
        Color_array (Xi, 2, X, Y) ← Image (x, y).G
    End loop X, Y
    Save in buffer named Frame_files (Videono) the array Color_array.
End loop Xi
```

3.2.3 Shots Boundaries Detection

In order to detect the video shots boundaries three methods have been considered, these three algorithms depending on two major elements; the first one is the mean for all the frames in the video, which is calculated its value varies in each algorithm. The second element is the relative threshold value which is varies from video to another.

To detect a shot, the relative threshold value is multiplied by the overall mean of the frames in the video to get the absolute threshold value. Then a compare between the absolute threshold with the mean of each frame, if the mean of the tested frame is larger than the absolute threshold then a new shot is detected.

The three methods gave the same results so we considered the first the result of the first method only in our work.

Code list (3.2) illustrates the steps of the first implemented method to allocate the shots boundaries (i.e., the start and the end frames numbers); in this Code List the mean is calculated for the difference between pixels in two successive frames.

Code List (3.2) Shots Boundaries Detection-1

Input:

```
    Videono is the video number.
    Th is a predefined relative threshold for each video.
    Tf is the total number of frames in the video.
    Le is the block size.
```

Output:

```
    Shot (Video No.) Is a buffer contains the record Shots_Record which holds the start
        and end frames numbers of a shot.
```

Steps:

```
  From buffer named Frame_file (Videno) the output of code list (3.1) gets the array
      named Color_array.

  For each frame in the video Xi
     Count ← 0
     For each block in the frame Ix, Iy
        Xs ← Ix × Le, Xe ← Xs + Le - 1
        Ys ← Iy × Le, Ye ← Ys + Le – 1
        Difference ← 0
        For each color (Red, Blue and Green) C
           For X ← Xs to Xe
              For Y ← Ys to Ye
```

Continue

```
        Difference ← Difference + $|Color\_array(Xi, C, X, Y) - Color\_array(Xi + 1, C, X, Y)|$
       End loop Y
     End loop X
   End loop C
   Mean_Difference (Xi) ← Mean_Difference (Xi) + Difference
   Count ← Count + 1
  End loop Iy, Ix
  Mean_Difference (Xi) ← Mean_Difference (Xi) / Count
End loop Xi
For each frame in the video Xi
  Mean ← Mean + Mean_Difference (Xi)
End loop Xi
Mean ← Mean / Tf
B ← 0
For each frame in the video Xi
   If Mean_ Difference (Xi) > Mean × Th Then
      Shots_Record (Count).Start ← B
      Shots_Record (Count).End ← Xi
      B ← Xi + 1, Count ← Count + 1
   End If
 End loop Xi
 Shots_Record (Count).End ← Tf
 Shots_Record (Count).Start ← B
 Save in buffer named Shots (Videono) the record Shots_Record.
```

In code list (3.3) three kind of the overall means have been calculated, the first is for the absolute difference between the mean of two successive frames, the second is for the square difference between the mean of two successive frames, the last one is cubic absolute difference between the mean of two successive frames, and it is found that the three computed type of means gave a peak at the same position (frame) approximately, there for only one of them can be used as a criteria to allocate the shots boundaries, or the sum of the three kinds of the mean of difference could be utilized to define the absolute threshold value by multiplying the sum of the total means by the relative threshold value.

Code List (3.3) Shots Boundaries Detection-2

Input:

Videono is the video number.

Th is a given threshold for each video.

Tf is the total number of frames in the video.

Output:

Shot2 (Video No.) is a buffer contains the record Shots2_Record which holds the start and end frames numbers of a shot.

Steps:

From buffer named Frame_file (Videno) the output of code list (3.1) gets the array Color_array.

```
For each frame in the video Xi
    For each color (Red, Blue, Green) C
        For each pixel in the frame X, Y
            $\text{Mean}(C, Xi) \leftarrow \text{Mean}(C, Xi) + \text{Color\_array}(Xi, C, X, Y)$
        End loop X, Y
        $\text{Mean}(C, Xi) \leftarrow \text{Mean}(C, Xi) / (H \times W)$
    End loop C
End loop Xi

For each frame in the video Xi
    For each color (Red, Blue, Green) C
        $\text{Difference1}(C, Xi) \leftarrow |\text{Mean}(C, Xi+1) - \text{Mean}(C, Xi)|$
        $\text{Difference2}(C, Xi) \leftarrow (\text{Mean}(C, Xi+1) - \text{Mean}(C, Xi))^2$
        $\text{Difference3}(C, Xi) \leftarrow |\text{Mean}(C, Xi+1) - \text{Mean}(C, Xi)|^3$
```

Continue

```
        Mean_Diff1(C) ← Mean_Diff1 (C) + Difference1(C, Xi)
        Mean_Diff2(C) ← Mean_Diff2 (C) + Difference2(C, Xi)
        Mean_Diff3(C) ← Mean_Diff3 (C) + Difference3(C, Xi)
    End loop C
End loop Xi

For each color (Red, Blue, Green) C
    Mean_Diff1(C) ← Mean_Diff1(C)/ Tf
    Mean_Diff2(C) ← Mean_Diff2 (C) / Tf
    Mean_Diff3(C) ← Mean_Diff3 (C) / Tf
End loop C

B ← 0
For each frame in the video Xi
    If Difference1 (0, Xi) > Mean_Diff1 (0) * Th and
       Difference1 (1, Xi) > Mean_Diff1 (1) * Th and
       Difference1 (2, Xi) > Mean_Diff1 (2) * Th Then
          Shots_Record (Count).Start ← B
          Shots_Record (Count).End ← Xi
          B ← Xi + 1, Count ← Count + 1
    End If
End loop Xi
Shots_Record (Count).End ← Tf
Shots_Record (Count).Start ← B
Save in buffer named Shot2 (Videono) the record Shots_Record.
```

In code list (3.4) three overall means had been calculated one for the absolute difference other for the square difference and the last one for the cubic absolute difference, the above difference is computed between the mean of two successive frames, the mean of the frames is computed from the difference between two neighbor pixels in the frame. and it is found that the three computed type of means gave a peak at the same position (frame) approximately, there for only one of them can be used as a criteria to allocate the shots boundaries, or the sum of the three kinds of the mean of difference could be utilized to define the absolute threshold value by multiplying the sum of the total means by the relative threshold value.

Code List (3.4) Shots boundaries Detection-3

```
Input:
    Videono is the video number.
    Th is a given threshold for each video.
    Tf is the total number of frames in the video.
Output:
    Shot3 (Video No.) is a buffer contains the record Shots3_Record which holds the start
        and end frames numbers of a shot.
Steps:
    From buffer named Frame_file (Videno) the output of code list (3.1) gets the array
        Color_array.
    For each frame in the video Xi
        Pixel_Diff ← 0
        For each color (Red, Blue, Green) C
            For each pixel in the frame X, Y
                Pixel_Diff ← $|Color\_array(Xi, C, X, Y+1) - Color\_array(Xi, C, X, Y)|$
                Mean_pixel_Diff(C, Xi) ← Mean_pixel_Diff (C, Xi) + Pixel_Diff
            End loop X, Y
        End loop C
    End loop Xi
    Mean_pixel_Diff(C, Xi) ← Mean_pixel_Diff (C, Xi) / (H-1 × W)
    For each frame in the video Xi
        For each color (Red, Blue, Green) C
            Difference1(C, Xi) ← $|Mean\_pixel\_Diff(C, Xi+1) - Mean\_pixel\_Diff(C, Xi)|$
            Difference2 (C, Xi) ← $(Mean\_pixel\_Diff(C, Xi+1) - Mean\_pixel\_Diff(C, Xi))^2$
            Difference3 (C, Xi) ← $|Mean\_pixel\_Diff(C, Xi+1) - Mean\_pixel\_Diff(C, Xi)|^3$
```

Continue

```
    Mean_Diff1 (C) ← Mean_Diff1 (C) + Difference1(C, Xi)
    Mean_Diff2 (C) ← Mean_Diff2 (C) + Difference2(C, Xi)
    Mean_Diff3 (C) ← Mean_Diff3 (C) + Difference3(C, Xi)
  End loop C
End loop Xi
For each color (Red, Blue, Green) C
    Mean_Diff1(C) ← Mean_Diff1(C)/ Tf
    Mean_Diff2(C) ← Mean_Diff2 (C) / Tf
    Mean_Diff3(C) ← Mean_Diff3 (C) / Tf
End loop C
B ← 0
For each frame in the video Xi
    If Difference1 (0, Xi) > Mean_Diff1 (0, Xi) * Th And
       Difference1 (1, Xi) > Mean_Diff1 (1, Xi) * Th And
       Difference1 (2, Xi) > Mean_Diff1 (2, Xi) * Th Then
          Shots_Record (Count).Start ← B
          Shots_Record (Count).End ← Xi
          B ← Xi + 1, Count ← Count + 1
    End If
End loop Xi
Shots_Record (Count).End ← Tf
Shots_Record (Count).Start ← B
Save in buffer named Shot3 (Videono) the record Shots_Record.
```

3.2.4 Static and Dynamic Classification

The blocks belong to each frame had been classified into static and dynamic blocks, by taking the difference between the two successive frames, then this difference is compared with the calculated threshold, if the difference is larger than the threshold then the block is considered as a dynamic block otherwise it is a static block, code list (3.5) shows the detail of this classification process.

Code list (3.5) Classification of Static and Dynamic Blocks

Input:

Videono is the video number.

Le is the length of the block.

Th is a given threshold.

Output:

Static_Dynamic_records (Videono) is a buffer contains the records Static_record, Dynamic_record they hold the information about the static blocks and information about the dynamic blocks.

Steps:

```
Threshold = $le \times le \times 3 \times Th$
From buffer named Frame_file (Videno) the output of code list (3.1) gets
        the array named Color_array.
For each frame in the shot Xi
   $S \leftarrow 0$
   $D \leftarrow 0$
   For each block in the frame Ix, Iy
       $Xs \leftarrow Ix \times Le$
       $Xe \leftarrow Xs + Le - 1$
       $Ys \leftarrow Iy \times Le$
       $Ye \leftarrow Ys + Le - 1$
       $Difference \leftarrow 0$
       For each color (Red, Blue and Green) C
          For $X \leftarrow Xs$ to $Xe$
             For $Y \leftarrow Ys$ to $Ye$
```

```
              Difference ← Difference + $|Color\_array(Xi, C, X, Y) - Color\_array2(Xi+1, C, X, Y)|$
             End loop Y
           End loop X
         End loop C

         If Difference < Threshold Then
              S ← S + 1
              For each color (Red, Blue and Green) C
                 Static_record (C, Xi).Scount(S).Xss ← Xs
                 Static_record (C, Xi).Scount(S).Xes ← Xe
                 Static_record (C, Xi).Scount(S).Yss ← Ys
                 Static_record (C, Xi).Scount(S).Yes ← Ye
              End loop C
           Else
               D ← D + 1
               For each color (Red, Blue and Green) C
                  Dynamic_record (C, Xi).Dcount (D).Xs ← Xs
                  Dynamic_record (C, Xi).Dcount (D).Xe ← Xe
                  Dynamic_record (C, Xi).Dcount (D).Ys ← Xs
                  Dynamic_record (C, Xi).Dcount (D).Ye ← Xe
               End loop C
           End if
       End loop Ix, Iy

      For each color (Red, Blue and Green) C
           Static_record (C, Xi).Scounter ← S
           Dynamic_record (C, Xi).Dcounter ← D
      End loop C
   End loop Xi
```

3.2.5 Feature Extraction

First it should be clear that the feature extraction stage was applied three colors (red, blue and green) on both the static and the dynamic blocks, two types of features have adopted in our work the histogram features and the textures features, as mentioned in chapter two. The steps of determining the color histogram are listed in code list (3.6).

Code List (3.6) Calculate the Color Histogram

```
Input:
      Videono is the video number.
Output:
      Hist_S is an array contains color histogram values for the static blocks.
      Hist_D is an array contains color histogram values for the dynamic blocks.
Steps:
   Form buffer named Shot (Videono) the output of code list (3.2) gets the record
     Shots_Record.
   For each shot in the video Si
     Bs ← Shots_Record (Si).Start
     Es ← Shots_Record (Si).End

     From buffer named Frame_files (Videono) the output of code list (3.1) gets the array
         named Color_array.
     From buffer named Static_Dynamic_Records (Videono) the output of code list (5.3)
         gets the records Static_record and Dynamic_record.

     For Xi ← Bs to Es
       For each color (Red, Blue and Green)
          Sc ← Static_record (C, Xi).Scounter
          Dc ← Dynamic_record (C, Xi).Dcounter
          For S ← 1 to Sc
            Xs ← Static_record (C, Xi).Scount(S).Xs
            Xe ← Static_record (C, Xi).Scount(S).Xe
            Ys ← Static_record (C, Xi).Scount(S).Ys
            Ye ← Static_record (C, Xi).Scount(S).Ye
```

Continue

```
            For X ← Xs to Xe
              For y ← Ys to Ye
                Hist_S (C, Color_array (Xi, C, X, Y)) ← Hist_S (Xi, C, Color_array(C, X, Y)) + 1
              End loop Y
            End loop X
          End loop S
          For D ← 1 to Dc
              Xs ← Dynamic_record (C, Xi).Dcount (D).Xs
              Xe ← Dynamic_record (C, Xi).Dcount (D).Xe
              Ys ← Dynamic_record (C, Xi).Dcount (D).Ys
              Ye ← Dynamic_record (C, Xi).Dcount (D).Ye
              For X ← Xs to Xe
                 For y ← Ys to Ye
                     Co ← Color_array (Xi,C, X, Y)
                     Hist_D (Xi, C, Co) ← Hist_D (Xi, C, Co) + 1
                 End loop Y
              End loop X
          End loop D
        End loop C
      End loop Xi
    End loop Si
```

Forms the output of Code List (3.6), the following features were determined:

1. Mean of the color histogram, calculated by using equation (2.6). Code list (3.7) presents of steps of determining the Mean of the color histogram values.

Code list (3.7) Calculate the Mean for the Color Histogram

Input:

Videono is the video number.

Output:

Mean_S is an array contains the mean values for the static blocks.

Mean_D is an array contains the mean values for the static blocks.

Steps:

```
Form buffer named Shots (Videono) the output of code list (3.2) gets the record
   Shots_Record.

For each shot in the video Si
   Bs ← Shots_Record (Si).Start
   Es ← Shots_Record (Si).End
   For Xi ← Bs to Es
     Sum_S ← 0, Sum_D ← 0
     Call Calculate the Color Histogram "code list (3.6)".
     For each color (Red, Blue and Green) C
        For Xc ← 0 to 255
           Sum_S ← Sum_S + Hist_S(C, Xc)
           Sum_D ← Sum_D + Hist_D(C, Xc)
           Mean_S (C, Xi) ← Mean_S (C, Xi) + (Xc × Hist_S (C, Xc))
           Mean_D (C, Xi) ← Mean_D (C, Xi) + (Xc × Hist_D (C, Xc))
        End loop Xc
        Mean_S (C, Xi) ← Mean_S (C, Xi) / Sum_S
        Mean_D (C, Xi) ← Mean_D (C, Xi) / Sum_D
     End loop C
   End loop Xi
End loop Si
```

2. Standard Deviation of the color histogram, calculated by using equation (2.7). Code list (3.8) presents of steps of determining the value of Standard deviation of color histogram.

Code List (3.8) Calculate Standard Deviation of the Color Histogram

Input:

Videono is the video number.

Output:

Std_S is an array contains the standard deviation values for the static blocks.

Std_D is an array contains the standard deviation values for the dynamic blocks.

Steps:

```
Form buffer named Shots (Videono) the output of code list (3.2) gets the record
   Shots_Record.

For each shot in the video Si
  Bs ← Shots_Record (Si).Start
  Es ← Shots_Record (Si).End
  For Xi ← Bs to Es
     Sum_S ← 0, Sum_D ← 0

     Call Calculate the Color Histogram "code list (3.6)".
     Call Calculate the Mean for the Color Histogram "code list (3.7)".

     For each color (Red, Blue and Green) C
       For Xc ← 0 to 255
         Sum_S ← Sum_S + Hist_S(C, Xc)
         Sum_D ← Sum_D + Hist_D(C, Xc)
         $Std\_S (C, Xi) \leftarrow Std\_S (C, Xi) + (Xc - Mean\_S(C, Xi))^2 \times Hist\_S (C, Xc)$
         $Std\_D (C, Xi) \leftarrow Std\_D (C, Xi) + (Xc - Mean\_D(C, Xi))^2 \times Hist\_D (C, Xc)$
       End loop Xc
       $Std\_S (C, Xi) \leftarrow \sqrt{( Std\_S(C, Xi) / Sum\_S)}$
       $Std\_S (C, Xi) \leftarrow \sqrt{( Std\_S(C, Xi) / Sum\_S)}$
     End loop C
  End loop Xi
End Si
```

3. Mean Absolute Deviation, is calculated by using equation (2.8). Code list (3.9) presents of steps of determining the value of the mean absolute deviation.

Code List (3.9) Calculate Mean Absolute Deviation

Input:

Videono is the video number.

Output:

Mad_S is an array contains the mean absolute deviation values for the static blocks.

Mad_D is an array contains the mean absolute deviation values for the dynamic blocks.

Steps:

```
Form buffer named Shots (Videono) the output of code list (3.2) gets the record
    Shots_Record.

For each shot in the video Si
  Bs ← Shots_Record (Si).Start
  Es ← Shots_Record (Si).End
  For xi ← BS to Es
    Sum_S ← 0, Sum_D ← 0

    Call Calculate the Color Histogram "code list (3.6)".
    Call Calculate the Mean for the Color Histogram "code list (3.7)".

    For each color (Red, Blue and Green) C
      For Xc ← 0 to 255
        Sum_S ← Sum_S + Hist_S(C, Xc)
        Sum_D ← Sum_D + Hist_D(C, Xc)
        $mad\_S (C, Xi) \leftarrow mad\_S (C, Xi) + |Xc - Mean\_S(C, Xi)| \times Hist\_S (C, Xc)$
        $mad\_D (C, Xi) \leftarrow mad\_D (C, Xi) + |Xc - Mean\_D(C, Xi)| \times Hist\_D (C, Xc)$
      End loop Xc
      Mad_S (C, Xi) ← Mad_S (C, Xi) / Sum_S
      Mad_D (C, Xi) ← Mad_D (C, Xi) / Sum_D
    End loop C
  End loop Xi
End loop Si
```

4. Skewness is calculated by using equation (2.9). The skewness is the third order moment which is defined as the measure asymmetry about the mean. Code list (10.3) presents of steps of determining the values of the skewness.

Code List (3.10) Calculate the Skewness

```
Input:
    Videono is the video number.
Output:
    Sk_S is an array contains the skewness values for the static blocks.
    Sk_D is an array contains the skewness values for the dynamic blocks.
Steps:
    Form buffer named Shots (Videono) the output of code list (3.2) gets the record
        Shots_Record.
    For each shot in the video Si
      Bs ← Shots_Record (Si).Start
      Es ← Shots_Record (Si).End
      For Xi ← Bs to Es
        Sum_S ← 0, Sum_D ← 0
        Call Calculate the Color Histogram "code list (3.6)".
        Call Calculate the Mean for the Color Histogram "code list (3.7)".
        For each color (Red, Blue and Green) C
          For Xc ← 0 to 255
            Sum_S ← Sum_S + Hist_S(C, Xc)
            Sum_D ← Sum_D + Hist_D(C, Xc)
            $Sk\_S(C, Xi) \leftarrow Sk\_S(C, Xi) + |Xc - Mean\_S(C, Xi)|^3 \times Hist\_S(C, Xc)$
            $Sk\_D(C, Xi) \leftarrow Sk\_D(C, Xi) + |Xc - Mean\_D(C, Xi)|^3 \times Hist\_D(C, Xc)$
          End loop Xc
          $Sk\_S(C, Xi) \leftarrow \sqrt[3]{Sk\_S(C, Xi) / Sum\_S}$
          $Sk\_D(C, Xi) \leftarrow \sqrt[3]{Sk\_D(C, Xi) / Sum\_D}$
        End loop C
      End loop Xi
```

All the above features are determined by using the color histogram of the image, the following textural features have been determined:

1. Energy, the average energy of the gradient image was calculated by using equation (2.4). Three directions of image differencing have been determined (horizontal, vertical and diagonal) and considered as the gradient components. This determination was done on each block in the frame (both the static and dynamic blocks) and for the three color components (Red, Blue, Green); the average energy of the gradient image will result from the summation of the average energy of gradient of the blocks divided by the number of the blocks. Code list (3.11) illustrates the steps of determining the values of average energy.

Code List (3.11) Calculate the Average Energy

Input:

Videono is the video number.

Output:

Ene_S is an array contains the average energy values for the static blocks.

Ene_D is an array contains the average energy values for the dynamic blocks

Steps:

```
Form buffer named Shots (Videono) the output of code list (3.2) gets the record
   Shots_Record.
From buffer named Frame_files (Videono) the output of code list (3.1) gets the array
      Color_array.
From buffer named Static_Dynamic_Records (Videono) the output of code list (3.5)
   gets the records Static_record and Dynamic_record.

For each shot in the video Si
   Bs ← Shots_Record (Si).Start
   Es ← Shots_Record (Si).End
   For Xi ← Bs to Es
     For each color (Red, Blue and Green) C
       Sc ← Static_record (C, Xi).Scounter
       Dc ← Dynamic_record (C, Xi).Dcounter
```

Continue

```
          For S ← 1 to Sc
             Xs ← Static_record (C, Xi).Scount(S).Xs
             Xe ← Static_record (C, Xi).Scount(S).Xe
             Ys ← Static_record (C, Xi).Scount(S).Ys
             Ye ← Static_record (C, Xi).Scount(S).Ye
             Call Determine the average energy of the HVD gradient of the sub-blocks "Code
                 List (3.12)".
             Ene_S(C, Xi, 0) ← Ene_S (C, Xi, 0) + Energy_h
             Ene_S(C, Xi, 1) ← Ene_S (C, Xi, 1) + Energy_v
             Ene_S(C, Xi, 2) ← Ene_S (C, Xi, 2) + Energy_d
          End loop S
          Ene_S(C, Xi, 0) ← Ene_S (C, Xi, 0) / Sc
          Ene_S(C, Xi, 1) ← Ene_S (C, Xi, 1) / Sc
          Ene_S(C, Xi, 2) ← Ene_S (C, Xi, 2) / Sc
          For D ← 1 to Dc
               Xs ← Dynamic_record (C, Xi).Dcount (D).Xs
               Xe ← Dynamic _record (C, Xi).Dcount (D).Xe
               Ys ← Dynamic _record (C, Xi).Dcount (D).Ys
               Ye ← Dynamic _record (C, Xi).Dcount (D).Ye
               Call Determine the average energy of the HVD gradient of the sub-blocks
                   "Code List (3.12)".
               Ene_D(C, Xi, 0) ← Ene_D (C, Xi, 0) + Energy_h
               Ene_D(C, Xi, 1) ← Ene_D (C, Xi, 1) + Energy_v
               Ene_D(C, Xi, 2) ← Ene_D (C, Xi, 2) + Energy_d
         End loop D
         Ene_D(C, Xi, 0) ← Ene_D (C, Xi, 0) / Dc
         Ene_D(C, Xi, 1) ← Ene_D (C, Xi, 1) / Dc
         Ene_D(C, Xi, 2) ← Ene_D (C, Xi, 2) / Dc
    End loop C
  End loop Xi
End loop Si
```

Continue

Code list (3.12) is established to determine the average energies of the vertical, horizontal and diagonal components of the gradient, these average energies are defined as follows:

$$E_H = \frac{1}{3H(W-1)}\sum_{y=0}^{H-1}\sum_{x=0}^{W-2}\sum_{c=0}^{2}(A(c,x,y)-A(c,x+1,y))^2 \qquad (3.1)$$

$$E_V = \frac{1}{3W(H-1)}\sum_{y=0}^{H-2}\sum_{x=0}^{W-1}\sum_{c=0}^{2}(A(c,x,y)-A(c,x,y+1))^2 \qquad (3.2)$$

$$E_D = \frac{1}{3(H-1)(W-1)}\sum_{y=0}^{H-2}\sum_{x=0}^{W-2}\sum_{c=0}^{2}(A(c,x,y)-A(c,x+1,y+1))^2 \qquad (3.3)$$

The above average energies are computed for the static and dynamic sub-blocks.

Code List (3.12) Determine the average energy of the HVD gradient of the sub-blocks

Input:

Xs is the x-coordinate of the left side of the sub-block.
Xe is the x-coordinate of the right side of the sub-block.
Ys is the y-coordinate of the left side of the sub-block.
Ye is the y-coordinate of the right side of the sub-block.
Color_array is an array contains color component data for each frame.
Xi is the frame index.

Output:

Energy_h is an array contains the average energy values for horizontal gradient in the sub-blocks.
Energy_v is an array contains the average energy values for vertical gradient in the sub-blocks.
Energy_d is an array contains the average energy values for diagonal gradient in the sub-blocks.

Steps:

```
Sum1 ← 0
Sum2 ← 0
Energy_h ← 0
Energy_v ← 0
Energy_d ← 0
For each color (Red, Blue and Green) C
  For Y ← Ys to Ye – 1
    For X ← Xs to Xe
      Ene_h ← Ene_h + $(Color\_array (Xi, C, X, Y + 1) – Color\_array (Xi, C, X, Y))^2$
      Sum1← Sum1 + 1
    End loop Y
  End loop X
  For Y ← Ys to Ye
   For X ← Xs to Xe - 1
     Ene_v ← Ene_v + $(Color\_array (Xi, C, X + 1, Y) – Color\_array (Xi, C, X, Y))^2$
   End loop X
 End loop Y
 For Y ← Ys to Ye – 1
   For X ← Xs to Xe - 1
     Ene_d ← Ene_d + $(Color\_array (Xi, C, X + 1, Y + 1) – Color\_array (Xi, C, X, Y))^2$
     Sum1← Sum1 + 1
   End loop X
 End loop Y
 Energy_h ← Ene_h/Sum1
 Energy_v ← Ene_v/Sum1
 Energy_d ← Ene_d /Sum2
End loop C
```

2. Contrast-1, calculated by using equation (2.2). In order to compute the contrast the standard deviation of each block (both static and dynamic) is computed for three color component (Red, Blue, Green) and then take the average of these values and consider as the standard deviation of the frame in the same way the average of the block's mean is computed for each frame and the contrast is computed by dividing the average of the standard deviation over the average of the mean for each frame. Code list (3.12) illustrates the implementation step to determined Contrast-1.

Code List (3.13) Calculate the Contrast-1

```
Input:
        Videono is the video number.

Output:
        Cont _S is an array contains the contrast values for the static blocks.
        Cont_D is an array contains the contrast values for the dynamic blocks.

Steps:
       Form buffer named Shots (Videono) the output of code list (3.2) gets the record
          named Shots_Record.
       From buffer named Frame_files (Videono) the output of code list (3.1) gets the array
             named Color_array.
       From buffer named Static_Dynamic_Records (Videono) the output from the code list
             (5.3) gets the records Static_record and Dynamic_record

     For each shot in the video Si
          Bs ← Shots_Record (Si).Start
          Es ← Shots_Record (Si).End
```

Continue

```
For Xi ← Bs to Es
  For each color (Red, Blue and Green) C
    Xe ← Static_record (C, Xi).Scount(S).Xe
    Ys ← Static_record (C, Xi).Scount(S).Ys
    Ye ← Static_record (C, Xi).Scount(S).Ye
    Mean (Xi, C) ← 0
    Sc ← Static_record (C, Xi).Scounter
    Dc ← Dynamic_record (C, Xi).Dcounter
    For S ← 1 to Sc
       Xs ← Static_record (C, Xi).Scount(S).Xs
       Call Determined the Mean and Stander for sub-blocks "Code List (3.14)".
       Mean_S (C, Xi) ← Mean_S (C, Xi) + M_block (C, S)
       Mean_Std_S (C, Xi) ← Mean_Std_S (C, Xi) + Std_Block (C, S)
    End loop S
    Mean_S (C, Xi) ← Mean_S (C, Xi) /Sc
    Mean_Std_S (C, Xi) ← Mean_Std_S (C, Xi) / Sc
    Cont_S (C, xi) ← Mean_Std_S (C, Xi) / Mean_S (C, Xi)
     For D ← 1 to Dc
        Xs ← Dynamic_record (C, Xi).Dcount (D).Xs
        Xe ← Dynamic _record (C, Xi).Dcount (D).Xe
        Ys ← Dynamic _record (C, Xi).Dcount (D).Ys
        Ye ← Dynamic _record (C, Xi).Dcount (D).Ye
        Call Determined the Mean and Stander for sub-blocks "Code List (3.14)".
        Mean_D (Xi) ← Mean_D (C, Xi) + M_block (C, D)
        Mean_Std_D (C, Xi) ← Mean_Std_D (C, Xi) + Std_block (C, D)
     End loop D
```

Continue

```
        Mean_D (C, Xi) ← Mean_D (C, Xi) / Dc
        Mean_Std_D (C, Xi) ← Mean_Std_D (C, Xi) / Dc
        Cont_D (C, xi) ← Mean_Std_D (C, Xi) / Mean_D (C, Xi)
      End loop C
    End loop Xi
  End loop Si
```

Code list (3.14), is established to determine the mean and the standard deviation for each sub-block in the image (whether it is static or dynamic).

Code List (3.14) Determined the Mean and Stander for sub-blocks

Input:

Xs is the x-coordinate of the left side of the sub-block

Xe is the x-coordinate of the right side of the sub-block

Ys is the y-coordinate of the left side of the sub-block

Ye is the y-coordinate of the right side of the sub-block

Color_array is an array contains color component data for each frame

Bi is the block index.

Xi is the frame index.

Output:

Std_block it is an array contains the standard deviation values for the sub-blocks.

M_block it is an array contains the mean values for the sub-blocks.

Steps:

```
For each color (Red, Blue and Green) C
  For X ← Xs to Xe
    For Y ← Ys to Ye
      Counter (C) ← Counter (C) + 1
```

Continue

```
      M_block (C, Bi) ← M_block (C, Bi) + Color_arrary (Xi, C, X, Y)
    End loop Y
  End loop X
  M_block (C, Bi) ← M_block (C, Bi) / Counter (C)
  For X ← Xs to Xe
    For Y ← Ys to Ye
      Std_block (C, Bi) ← Std_block (C, Bi) + (Color_arrary (Xi, C, X, Y) – M_block(C, Bi)) ^ 2
    End loop Y
  End loop
  Std_block (C, Bi) ← $\sqrt{\text{Std\_block (C, Bi) / Counter(C)}}$
End loop C
```

3. Contrast-2, it is calculated by using equation (2.3). In order to compute the contrast minimum pixel value and the maximum pixel value from each block (both static and dynamic) in the frame is computed and the mid value for each frame is computed from the average of the minimum and maximum values of the blocks of each frame. The contrast of each frame is computed by dividing the summation of the points that are smaller of the mid value over the summation of the points that are larger than the mid value. Code list (3.15) shows the implementation steps to determine the Contrast-2.

Code List (3.15) Calculate Contrast-2

Input:

Videono is the video number.

Output:

Cont _S is an array contains the contrast values for the static blocks.

Cont_D is an array contains the contrast values for the dynamic blocks

Steps:

```
    Form buffer named Shots (Videono) the output of code list (3.2) gets the record
      Shots_Record.
```

Continue

```
From buffer named Frame_files (Videono) the output of code list (3.1) gets the
    array named Color_array.
From buffer named Static_dynamic_records (Videono) the output of code list (5.3)
    gets the records Static_record and Dynamic_record.

For each shot in the video Si
   Bs ← Shot_Record (Si).Start
   Es ← Shot_Record (Si).End
   For Xi ← Bs to Es
     For each color (Red, Blue and Green) C
       Sc ← Static_record (C, Xi).Scounter
       Dc ← Dynamic_record (C, Xi).Dcounter
       For S ← 1 to Sc
          Xs ← Static_record (C, Xi).Scount(S).Xs
          Xe ← Static_record (C, Xi).Scount(S).Xe
          Ys ← Static_record (C, Xi).Scount(S).Ys
          Ye ← Static_record (C, Xi).Scount(S).Ye
          Call Determined the Min and Max of the Sub-blocks "code List (3.17)".
          M_Min_S (C, Xi) ← M_Min_S (C, Xi) + Min (C, S)
          M_Max_S (C, Xi) ← M_Max_S (C, Xi) + Max (C, S)
          M_Min_S (C, Xi) ← M_Min_S (C, Xi) /Sc
          M_Max_S (C, Xi) ← M_Max_S (C, Xi) / Sc
         Median_S (C) ← (M_Min_S (C, Xi) + M_Max_S (C, Xi)) / 2
         For X ← Xs to Xe
           For Y ← Ys to Ye
               If Color_array (Xi, C, X, Y) > Meadin_S (C) Then
                   Small(C) ← Small(C) + Color_array (Xi, C, X, Y)
               Else
```

Continue

```
                    Large(C) ← Large(C) + Color_array (Xi, C, X, Y)
                  End if
                End loop Y
              End loop X
            End loop S
        Cont_S (C, Xi) ← Smal_S(C) / Large_S(C)
        For D ← 1 to Dc
          Xs ← Dynamic_record (C, Xi).Dcount (D).Xs
          Xe ← Dynamic _record (C, Xi).Dcount (D).Xe
          Ys ← Dynamic _record (C, Xi).Dcount (D).Ys
          Ye ← Dynamic _record (C, Xi).Dcount (D).Ye
          Call Determined the Min and Max of the Sub-blocks "code List (3.17)".
          M_Min_D (C, Xi) ← M_Min_D (C, Xi) + Min (C, D)
          M_Max_D (C, Xi) ← M_Max_D (C, Xi) + Max (C, D)
          M_Min_D (C, Xi) ← M_Min_D (C, Xi) / Dc
          M_Max_D (C, Xi) ← M_Max_D (C, Xi) / Dc
          Median_D (C) ← (M_Min_D (C, Xi) + M_Max_D (C, Xi)) / 2
          Large (C) ← 0, Small (C) ← 0
          For X ← Xs to Xe
            For Y ← Ys to Ye
              If Color_array (Xi, C, X, Y) > Meadin_D (C) Then
                Small(C) ← Small(C) + Color_array (Xi, C, X, Y)
              Else
                Large(C) ← Large(C) + Color_array (Xi, C, X, Y)
              End if
            End loop Y
          End loop X
        End loop D
        Cont_D (C, Xi) ← Smal_D(C) / Large_D(C)
      End loop C
  End loop Xi
End loop Si
```

Code list (3.16) computes the minimum and maximum values for each block in the frame, whether it is static and dynamic blocks.

Code List (3.17) Determined the Min and Max of the Sub-blocks

Input:

Xs is the x-coordinate of the left side of the sub-block.

Xe is the x-coordinate of the right side of the sub-block.

Ys is the y-coordinate of the left side of the sub-block.

Ye is the y-coordinate of the right side of the sub-block.

Color_array is an array contains color component data for each frame.

Bi is the block index.

Xi is the frame index.

Output:

Min it is array of the Minimum values.

Max it is array for the Maximum values.

Steps:

```
For each color (Red, Blue and Green) C
  Min (C, Index) ← Color_array (Xi, C, Xs, Ys)
  Max (C, Index) ← Color_array (Xi, C, Xs, Ys)

  For X ← Xs+1 to Xe
    For Y ← Ys to Ye
            If Min (C, Index) > Color_arrary (C, X, Y) Then
        Min (C, Index) ← Color_arrary (C, X, Y)
      End if
      If Max (C, Index) < Color_arrary (C, X, Y) Then
        Max (C, Index) ← Color_arrary (C, X, Y)
      End if
     End loop Y
   End loop X
  End loop C
```

4. Modification of Fractal Dimension: one of the important textural feature is fractal dimension, in this research work a modification (simplified) method for determination of fractal dimension was implemented, the modification is proposed to handle the high computational complexity of the traditional methods for determination of fractal implies the following steps:
   1. Scan the blocks (static/dynamic) and find the mean ($\overline{g}$) and standard deviation (σ) of the pixels values
   2. determined the following parameters:

      $$M_1 = \overline{g} - (\alpha \times \sigma) \qquad (3.4)$$

      $$M_2 = \overline{g} + (\alpha \times \sigma) \qquad (3.5)$$

      Where α was taken (3).

      And apply the following conditions:

      If $M_1 < 0$ then $M_1=0$.

      If $M_2 >255$ then $M_2=255$.
   3. determined the following slope value:

      $$S = \frac{M_2 - M_1}{N_p} \qquad (3.6)$$

      Where N_p is the number of threshold values (T_i) used to categorize the block's points into dark or bright points.
   4. for each threshold value (T_1……..T_{Np})
      a. Set the counter n_i= 0.
      b. Scan all pixels of the blocks (static/dynamic), and for the case where each scanned pixel has a value larger than T_i increment n_i by 1.

      End for
   5. then determined the estimated (approximate) fractal dimension parameter buy using the following equation:

$$H = \frac{N_p \sum_{i=1}^{N_p} i \log(n_i) - \sum_{i=1}^{N_p} i \sum_{i=1}^{N_p} \log(n_i)}{N_p \sum i^2 - \left(\sum i\right)^2} \qquad (3.7)$$

Code list (3.18) shows how to determine the modification of the fractal dimension (H).

Code List (3.18) Calculate modification of the fractal dimension (H)

Input:

Videono is the video number.

Np is the number of point.

Output:

Slop_S is an array that contains the slope values for the static blocks.

Slop_D is an array that contains the slope values for the static blocks.

Steps:

```
Form buffer named Shots (Videono) the output of code list (3.2) gets the record
  Shots_Record.
From buffer named Frame_files (Videono) the output of code list (3.1) gets the
  array named Color_array.
From buffer named Static_dynamic_records (Videono) the output of code list
  (5.3) gets the records Static_record and Dynamic_record.

For each scenes in the video Si
   Bs ← Shot_Record (Si).Start
   Es ← Shot_Record (Si).End
   For Xi ← Bs to Es
      For each color (Red, Blue and Green) C
         Sc ← Static_record (C, Xi).Scounter
         Dc ← Dynamic_record (C, Xi).Dcounter
```

Continue

```
For S ← 1 to Sc
    Xs ← Static_record (C, Xi).Scount(S).Xs
    Xe ← Static_record (C, Xi).Scount(S).Xe
    Ys ← Static_record (C, Xi).Scount(S).Ys
    Ye ← Static_record (C, Xi).Scount(S).Ye
    Call Determined the Mean and Stander for sub-blocks "Code List (3.14)".
     Mean_S (C, Xi) ← Mean_S (C, Xi) + M_block (C, S)
     Mean_Std_S (C, Xi) ← Mean_Std_S (C, Xi) + Std_Block (C, S)
     Mean_S (C, Xi) ← Mean_S (C, Xi) /Sc
     Mean_Std_S (C, Xi) ← Mean_Std_S (C, Xi) / Sc
     Min_S(C, Xi) ← Mean_S (C, Xi) - 3 × Mean_ Std_S (C, Xi)
     Max_S(C, Xi) ← Mean_S (C, Xi) + 3 × Mean_ Std_S (C, Xi)
     If Min_S(C, Xi) < 0 Then
         Min_S(C, Xi) ← 0
     End If
     If Max_S(C, Xi) > 255 Then
         Max_S(C, Xi) ← 255
     End If
       For I ← 1 to Np
           Po_S = Min_S(C, Xi) + (Max_S(C, Xi) – Min_S(C, Xi)) × I / Np
           For X ← Xs toXe
             For Y ← Ys to Ye
                    If Color_array (Xi, C, X, Y) ≤ Po_S   Then
                        PCount_S (C, Xi, I) ← PCount_S (C, Xi, I) + 1
                    End If
             End loop Y
           End loop X
       End loop I
     End loop S
```

Continue

```
        For D ← 1 to Dc
            Xs ← Dynamic_record (C, Xi).Dcount (D).Xs
            Xe ← Dynamic _record (C, Xi).Dcount (D).Xe
            Ys ← Dynamic _record (C, Xi).Dcount (D).Ys
            Ye ← Dynamic _record (C, Xi).Dcount (D).Ye
            Call Determined the Mean and Stander for sub-blocks "Code List (3.14)".
            Mean_D (Xi) ← Mean_D (C, Xi) + M_block (C, D)
            Mean_Std_D (C, Xi) ← Mean_Std_D (C, Xi) + Std_block (C, D)
            Mean_D (C, Xi) ← Mean_D (C, Xi) / Dc
            Mean_Std_D (C, Xi) ← Mean_Std_D (C, Xi) / Dc
            Min_D(C, Xi) ← Mean_D (C, Xi) - 3 × Mean_ Std_D (C, Xi)
            Max_D(C, Xi) ← Mean_D (C, Xi) + 3 × Mean_ Std_D (C, Xi)
            If Min_D (C, Xi) < 0 Then
               Min_D (C, Xi) ← 0
            End If
            If Max_D(C, Xi) > 255 Then
              Max_D(C, Xi) ← 255
            End If
            For I ← 1 to Np
              Po_D = Min_D(C, Xi) + (Max_D(C, Xi) – Min_D(C, Xi)) × I / Np
              For X ← Xs toXe
                 For Y ← Ys to Ye
                    If Color_array (Xi, C, X, Y) ≤ Po_D Then
                       PCount_D (C, Xi, I) ← PCount_D (C, Xi, I) + 1
                    End If
                 End loop Y
               End loop X
             End loop I
        End loop D
   End loop C
End loop Xi
```

Continue

```
   For Xi ← Bs to Es
     For each color (Red, Blue and Green) C
       K ← 0
       For I ← 1 to Np
         If PCount_S (C, Xi, I) > 0 Then
           Sum_I ← Sum_I + I
           Sqr_I ← Sqr_I + $(I)^2$
           Ig ← Ig + I × log (PCount_S(C, Xi, I))
           Cg ← Cg + log (PCount_S(C, Xi, I))
           K ← K + 1
         End If
       End loop I
       If K > 0 Then
         $Slop\_S(C, Xi) \leftarrow \frac{Np \times Cg - Sum\_I \times Ig}{Np \times Sqr\_I - (Sum\_I)^2}$
       End If
       Sum_I ← 0, Sqr_I ←0
       Ig ← 0, Cg ← 0
       K ← 0
       For I ← 1 to Np
         If PCount_D (C, Xi, I) > 0 Then
           Sum_I ← Sum_I + I
           Sqr_I ← Sqr_I + $(I)^2$
           Ig ← Ig + I × log (PCount_D(C, Xi, I))
           Cg ← Cg + log (PCount_D(C, Xi, I))
           K ← K + 1
         End If
       End loop I
       If K > 0 Then
         $Slop\_D(C, Xi) \leftarrow \frac{Np \times Cg - Sum\_I \times Ig}{Np \times Sqr\_I - (Sum\_I)^2}$
       End If
     End loop C
   End loop Xi
 End loop Si
```

3.2.6 Features Analysis

The first step of features analysis is to neglect all shots that have small number of frames; in our work all the shots that have less than 45 frames have been neglected. The next step is to select a number that should be less than the total number of frames in a specific shot, this number refers to the number of frames in each video sequence, Ten video sequence have been taken from each shot, in every sequence the list of frames that belongs to that sequence is determined by the start frame, end frame and the total number of frames in the specific shot. The same ten features will be recalculated for the list of frames in every sequence of the shot. The last step is to compute the average of each feature in each sequence, the results will put in matrix named pattern, it contains the average of each feature of the entire 49 shots and the indices of the matrix are the sequence number and the feature number. It should be known that all the above steps are applied on the three colors (red, blue and green) and for both static and dynamic blocks.

3.2.7 Power Discrimination

In order to reduce the computation time taken in the K-Means algorithm, and to improve the classification results, the discrimination power of the adopted features must be calculated;

This procedure use three matrices as a major input, first one is the pattern matrix which is mentioned in the features analysis, the second matrix is the M_Template matrix (the matrix contains the averages of the features calculated for all the frames in the entire 49 shots, and indexed by shot number and feature number), the third matrix is the Std_Template matrix (the matrix contains the standard deviation of the features calculated for all the frames in the entire 49 shots, and indexed by shot number and feature number), all the above matrices are for the three color components (Red, Blue, Green) and both have 20 features first ten features are for the static features and the last ten are for the dynamic features.

two types of distance measure had been used the first depends on the Euclidean distance and the second depending on the city block distance, the results of the two measures were approximately similar, therefore only one of them is consider in the next steps of the work.

The procedure of the power discrimination will include five steps, in the first step every feature is tested alone, the second step is the test of combination of two features, the third step is the test to the combination of three features, the fourth step is test to the combination of four features, the fifth step is test to the combination of five features, only the test of the single feature and the test of the combination of two features are mentioned in the code list (3.19).

Code List (3.19) Determined the Power Discrimination

```
Input:
    Pattern is an array contains means of features that extracted from the video sequence.
    M_Template is an array contains means of features that extracted from the shot's frames.
    Std _Template is an array contains standard deviation of features that extracted from the
        shot's frames.
    Index is an array indexed by the video sequence number and contains the shot number.
    Pn is the number of patterns.
    Sn is the number of shots.
    Fn is the number of features.
Output:
    Featues_Power_file1 is a buffer contains two records Feature_Record.Success and
        Feature_Record.Failure, they hold the high power discrimination and low power
        discrimination rates.
    Featues_Power_file2 is a buffer contains two records Feature_Record.Success2 and
        Feature_Record.Failure2 they hold the high power discrimination and low power
        discrimination rates.
```

Continue

Steps:

```
    For fi ← 1 to Fn
      Fo r P ← 1 to Pn
      Min ← 99999999999#
        For S ← 1 to Sn
          Diff ← (Abs (Pattern (P, Fi) – M_Template (S, Fi)) / Std_ Template (S, Fi))
          If Min > Diff Then
              Min ← Diff
              Sm ← S
          End if
        End loop S
        If Sm ← Index (p) Then
            Success (Fi) ← Success (Fi) + 1
        Else
            Failure (Fi) ← Failure (Fi) + 1
        End If
      End loop p
    End loop fi
    Feature_Record.Success (Fi) ← 100 * Success (Fi) / Xr
    Feature_Record.Failure (Fi) ← 100 * Failure (Fi)/ Xr
    For I ← 1 to Nf - 1
      For J ← I + 1 to Fn
        For P ← 1 to Xr
        Mn ← 99999999999#
        For Si ← 1 to Sn
          Diff ← 0
          Diff ← (Abs Pattern (P, I) – M_Template (S, I)) / Std_Template(S, I)) +
                    (Abs Pattern (P, J) – M_Template (S, J)) / Std_Template (S, J))
          If Mn > diff Then
              Mn ← Diff
              Sm ← S
          End if
        End loop S
        If Sm ← Index (P) Then
```

Continue

```
            Success2 (I, J) ← Success2 (I, J) + 1
        Else
            Failure2 (I, J) ← Failure2 (I, J) + 1
        End If
      End loop P
    End loop J
  End loop I
  Feature_Record.Success2 (I, J) ← 100 * Success2 (I, J) / Xr
  Feature_Record.Failure2 (I, J) ← 100 * Failure2 (I, J)/ Xr

  Save in buffered named Featues_Power_file1 the records Feature_Record.Success
    and   Feature_Record.Failure.
  Save in buffered named Featues_Power_file2 the records Feature_Record.Success2
    and   Feature_Record.Failure2.
```

3.2.8 K-means Algorithm

The last step of the work is the clustering, the algorithm used in the clustering is the *K-Means* algorithm, and the input to the algorithm will be only the features (for read, blue and green) that give high rate of success after applied Code List (3.19). Code List (3.20) shows how the *K-mean algorithm* has been implemented.

Code List (3.20) Implement the K-Means Algorithm

```
Input:
    Cntr is an array contains the centriods.
    Pn is the number of the videos sequence.
    Cn is the number of centriode.
    Fn is the number of features.
    Feature is an array contains the features that have high rate of power discrimination.
Output:
    Success array contains the successful rate for the features.
    Failure array contains the failure rate for the features.
Steps:
    Flag ← 1
```

Continue

```
While Flag ← 1
    Cc ← 0, Cm ← 0
    Max = -99999999999#
    For Pi ← 1 to Pn
      Min = 99999999999#
      For Ci ← 1 to Cn
        Diff ← 0
         For Fi ← 1 to Fn
           Diff ← Diff + (Cntr (Ci, Fi) – Feature (Pi, Fi)) $^2$
         End loop Fi
         If Min > $\sqrt{Diff}$ Then
            Min ← $\sqrt{Diff}$
            Cc ← Ci
         End If
      End loop Ci
      Count (Cc) ← Count (Cc) + 1
      Clust (Cc).Counter (Count (Cc)) ← Pi
       If Max < Count (Cc) Then
          Max ← Count (Cc)
           Cm ← Cc
       End If
    End loop Pi
    For Ci ← 1 to Cn
      For C ← 1 to Count (Ci)
        Pi ← Clust (Ci).Counter (C)
        For Fi ← 1 to Fn
          Scntr (Fi) ← Scntr (Fi) + Feature (Pi, Fi)
        End loop Fi
      End loop C
      For Fi ← 1 to Fn
        Ocntr (Ci, Fi) ← Ocntr (Ci, Fi)
        Cntr (Ci, Fi) ← Scntr (Fi)
```

Continue

```
                Scntr (Fi) ← 0
               Cntr(Ci, Fi) ← Cntr(Ci, Fi) / Count(Ci)
           End loop Fi
        End loop Ci
        Flag2 ← 0
        For Ci ← 1 to Cn
           If Count (Ci) ← 0 Then
               Flag2 ← 1, Flag ← 1
               For Fi ← 1 to Fn
                 Cntr (Ci, Fi) ← Cntr (Cm, Fi) × 0.95
                 Cntr (Cm, Fi) ← Cntr (Cm, Fi) × 1.05
               End loop Fi
           End If
        End loop Ci
        If Flag2 ← 0 Then
           Flag ← 0
           For Ci ← 1 to Cn
             For Fi ← 1 to Fn
                If (Cntr (Ci, Fi) – Ocntr (Ci, Fi)) ≤ 0.00001 Then
                Else
                   Flag ← 1
                End If
             End loop Fi
           End loop Ci
        End If
   End while loop
```

Chapter Two
Theoretical Background

2.1 Introduction

Fields ranging from commercial to military are needed to analyze data in an efficient and fast manner. And due to the digitization of data and advances in technology, it has become extremely easy to obtain and store large quantities of data, particularly multimedia data (video, image, audio). Multimedia data mining is a sub field of data mining that deal with the extraction of implicit knowledge, multimedia relationship, or other patterns, not explicitly stored in multimedia database. Feature selection and extraction is the pre-processing step of multimedia data mining. Obviously this is a critical step in the entire scenario of multimedia data mining.

2.2 Video Segmentation

Video, whether digital or analog, consists of a series of individual frames displayed at a constant rate (the effect of which is to give the illusion of motion) along with an associated audio track **[**Smea99**]**. generally, there are three types of videos; the produced, the raw, and the medical video. The examples of produced video are movies, news videos, dramas...etc. And, those of raw video are traffic videos, surveillance videos…etc. Ultra sound videos including echocardiogram can be an example of the medical videos; these different types of videos need to be treated differently to achieve these missing parts due to their different characteristics [Band02].

To allow any kind of content-based navigation of video, the material has first to be broken up into constituent elements and structured. For video, these elements are shots and scenes. A shot is defined as the video resulting from a continuous recording by a single camera. A scene is made up of multiple shots, while a television broadcast of a program consists of a collection of scenes. For studio broadcasts (for example the news transmitted live), it is fairly easy to break the program up as the boundaries between shots so these boundaries are hard. However, many television programs and most films use special post-production techniques to soften the boundaries, thus making them easier for the human eye, but more difficult to detect automatically [Smea99].

There are four major types of boundaries between shots [Gorm99, Smea00]:

1. **Cut**: This is a hard boundary and occurs when there is a complete change of shot over a span of two consecutive frames. This is commonly used in live or in studio transmissions.
2. **Fade**: There are two types of fade, a fade-out and a fade-in. A fade-out occurs when the picture gradually fades to a dot or black screen, while a fade-in occurs when the picture is gradually displayed from a black screen. Both these effects occur over a few frames, e.g. 12 frames for a half-second fade-out.
3. **Dissolve**: This is the simultaneous occurrence of a fade-out and a fade-in, the two frames being superimposed on each other over a fixed duration of, say, 1/2 second (12 frames). This can be used in live in-studio transmissions.

4. **Wipe**: This effect is like a virtual line going across the screen clearing one picture as it brings in another, again occurring over a few frames. It was particularly common in early TV (such as the *Batman* series, but it still used).

Each of these post-production and live techniques makes the automatic detection of shot boundaries in video a non-trivial task.

A number of techniques have been tried in shot boundary detection with varying degrees of success. Some of these techniques are pixel differences between adjacent frames, color histograms method which compares the intensity or color histograms between adjacent frames, and the edges detection in adjacent frames. Each of these techniques is known to work well for different transition types, e.g. frame comparison based on colors works well on cuts, but not on fades or dissolves, while edge detection handles wipes and dissolves quite well [Smea00, Zhon00].

In general, the most widely used basic unit in produced videos (i.e., movies, news videos) is a **shot** which is defined as collections of frames recorded from a single camera operation. Raw videos are usually recorded from a single fixed camera or multiple cameras with very limited camera motion without any camera on-off. Therefore, the concept of the shot is not relevant since whole video would be a shot by the above definition [Band02].

2.3 Scene Changes Detection

Shot based indexing techniques have been widely used to organize video data. Scene change detection is the most commonly used method to segment image sequences into coherent units for video indexing. A shot is a sequence of contiguous frames that are recorded from a camera.

There is usually one continuous action within a shot, with no major change of scene content. However, there are still many different changes in a video (e.g. object motion, lighting change and camera motion), it is a nontrivial task to accurately detect scene changes. Furthermore, the cinematic techniques used between scenes, such as dissolves, fade and wipes, produce gradual scene changes that are harder to detect. Scene cut detection algorithms have been studied since the early 90’s [Zhon00]. The basic method is to measure the pixel difference frame-to-frame in terms of intensity or color. The number of changed pixels is counted and if the number exceeds a certain percentage, a scene cut is detected. This method is not robust due to the camera and object motions that can cause large pixel value differences. Color histograms have been used to overcome the problem, as color distributions in successive frames are not significantly affected by camera or object motions. Assume H_i is an N-bin color histogram extracted from frame i, the frame difference is defined as [Fern03]:

$$D_i = \sum_{j=1}^{n} \left| H_i(j) - H_{i+1}(j) \right| \tag{2.1}$$

If D_i is larger than a given threshold, a scene cut is detected at the frame i+1.

Although color histogram difference is good for direct scene changes, gradual transitions such as fade-in, fade-out, dissolve and wipe cannot be accurately detected in the same way.

The edge detection method is used to solve this problem; this method is based on detecting edges in two adjacent images and comparing them. By detecting the appearance of intensity edges in a frame that are far away from the intensity edges in the previous frame, it should be possible to detect and classify the four different types of shot breaks [Smea99, Gorm99].

2.4 Motion Estimation

A lot of information can be extracted from time varying sequences of images, often more easily than from static images. For example, camouflaged objects are only easily seen when they move. Moreover, the relative sizes and position of objects are more easily determined when the objects move. The analysis of visual motion divides into two stages:

1. The measurement of the motion
2. The use of motion data to segment the scene into distinct objects, and to extract information, about the shape and motion of the objects.

There are two types of motion to consider: movement in the scene with a static camera, and movement of the camera. Since motion is relative anyway, these types of motion should be the same.

However, this is not always the case, since if the scene moves relative to the illumination, shadow effects need to be dealt with. Also, specularities can cause relative motion within the scene [Comp87].

The Motion estimation has been wildly used in many applications of video processing since it provides the most essential information for an image sequence; Motion estimation is defined as the process which generates the motion vectors that determine the differences between the blocks of the current frame and the blocks of the previous frame [Kuan03]. One of the most common methods of motion estimation is the Block Matching (BM); the block matching is a standard technique for encoding motion in video sequences. It aims at detecting the motion between two images in a block-wise sense. The blocks are usually defined by dividing the image frame into non-overlapping square parts [Gyao03, Watk94, Bolt06]. This method work on a sequence of frames, the current frame is predicted from a previous frame known as reference frame. The current frame is divided into macro blocks, typically 16 x 16 pixels in size. This choice of size is a good trade-off between accuracy and computational cost. However, motion estimation techniques may choose different block sizes, and may vary the size of the blocks within a given frame. Each macro block is compared to a macro block in the reference frame using some error measure, if there is no motion between fields, there will be high correlation between the pixel values. However, in the case of motion, the same or similar pixel values will be elsewhere and it will be necessary to search for them by moving the search block to all possible locations in the search area, and the best matching macro block is selected. A vector denoting the displacement of the macro block in the reference frame with respect to the macro block in the current frame is determined. This vector is known as motion vector [Moti00].

Different error measures can bee used for motion estimation. Among others, the sum of absolute differences (SAD) and the minimum squared error (MSE) are commonly used [Bane00].

2.5 Image Mining

Image mining deals with extraction of implicit knowledge, image data relationship or other patterns are not explicitly stored in images; image mining methodology uses ideas from computer vision, image processing, image retrieval, data mining, machine learning, databases and AI [Zhan01]. The fundamental challenge in image mining is to determine how low-level, pixel representation contained in an image or an image sequence can be effectively and efficiently processed to identify high-level spatial objects and relationships; typical image mining process involves preprocessing, transformations and feature extraction, mining (to discover significant patterns out of extracted features), evaluation and interpretation and establishment of the final knowledge. Various techniques have been utilized to image mining; they include object recognition, learning, clustering and classification. For example, Association rule mining is a well known data mining technique that aims to find interesting patterns in very large databases. Some preliminary work has been done to apply association rule mining on sets of images to find interesting patterns [Mali05].

Clearly, image mining is different from low-level computer vision and image processing techniques because the focus of image mining is in extraction of patterns from large collection of images, whereas the focus of computer vision and image processing techniques is in understanding and/or extracting specific features from a single image. While there seems to be some overlaps between image mining and content-based retrieval (both are dealing with large collection of images). The content-base retrieval requires the image search engine to find the set of images from a given image collection that is similar to the given query image [Haup02]; the image mining goes beyond the problem of retrieving relevant images. In image mining, the goal is the discovery of image patterns that are significant in a given collection of images. Perhaps, the most common misconception of image mining is that image mining is nothing more than just applying existing data mining algorithms on images; this is certainly not true because there are important differences between relational databases versus image databases [Zhan01]:

1. *Absolute versus relative values*. In relational databases, the data values are semantically meaningful. For example, age is 35 is well understood. However, in image databases, the data values themselves may not be significant unless the context supports them. For example, a grey scale value of 46 could appear darker than a grey scale value of 87 if the surrounding context pixels values are all very bright.

2. *Spatial information (Independent versus dependent position)*. Another important difference between relational databases and image databases is that the implicit spatial information is critical for interpretation of image contents but there is no such requirement in relational databases. As a result, image miners try to overcome this problem by extracting position-independent features from images first before attempting to mine useful patterns from the images.

3. *Unique versus multiple interpretations*. A third important difference deals with image characteristics of having multiple interpretations for the same visual patterns. The traditional data mining algorithm of associating a pattern to a class (interpretation) will not work well here. A new class of discovery algorithms is needed to cater to the special needs in mining useful patterns from images.

2.6 Color Image

Color is a property of light that is determined by its wavelength or by its composition as a blend of several wavelengths. The range of visible wavelengths of light is known as the *visible spectrum*, or simply the *spectrum*. The term *color* may also refer to a property of objects or materials, determined by which wavelengths of light they reflect, transmit, or emit. Typically only features of the composition of light that are detectable by humans are included, so color may also be considered as a psychological phenomenon [Colo06].

It is possible to construct almost all visible colors by combining the three primary colors (red, green and blue), because the human eye has only three different color receptors, each of them sensible to one of the three colors. Different combinations in the stimulation of the receptors enable the human eye to distinguish approximately *350000* colors [Colo03].

RGB color model is an additive model in which red, green and blue (often used in additive light models) are combined in various ways to reproduce other colors. The name of the model and the abbreviation "RGB" come from the three primary colors: Red, Green and Blue. These three colors should not be confused with the primary pigments of red, blue and yellow, known in the art world as "primary colors" [Rgb06].

Color image is a digital image that includes color information for each pixel. Color image can be modeled as three-band monochrome image data, where each band of data corresponds to different color. The actual information stored in the digital image data is the brightness information in each spectral band. When the image is to be presented the corresponding brightness information is displayed on the screen by picture elements that emit light energy corresponding to that particular color. Any typical color image is represented as red, green, blue, or RGB images. Using the 8-bit monochrome standard as a model, the corresponding color would have 24 bits/pixel, where 8-bits for each of the three color bands (red, green, blue). Figure (2.1) illustrates the typical RGB color image. Figure (2.2) illustrates that, in addition to referring to a row or column as a vector, we can refer to the red, green, blue value as a color pixel vector (R, G, B)[Umba98].

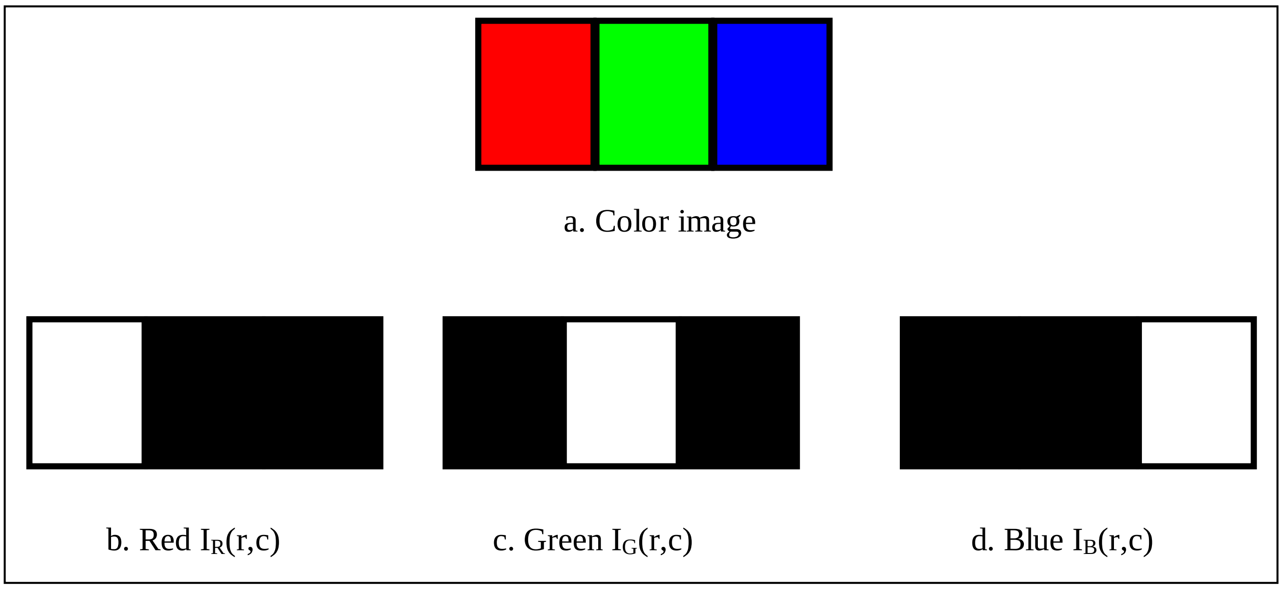

Figure (2.1) A typical RGB color image can be thought as three separate images: $I_R(r,c)$, $I_G(r,c)$ and $I_B(r,c)$ [Umba98].

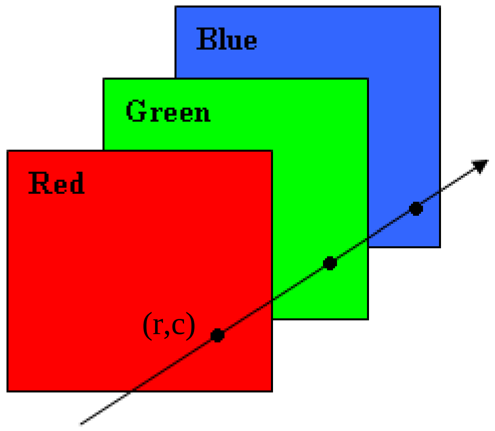

Figure (2.2) A color pixel vector consists of the red, green, blue pixel values (R, G, B) at one given row/column pixel coordinate (r,c) [Umba98].

2.7 Color and Textural Features

Feature (content) extraction is the basis of content-based image retrieval. In a broad sense, features may include both text-based features (key words, annotations) and visual features (color, texture, shape, faces). The features can be further classified as general features and domain-specific features. The former include color, texture, and shape features while the latter is application-dependent and may include, for example, features related to geometry of human faces and finger prints. Because of perception subjectivity, there is no single best representation for a given feature. For any given feature there exist multiple representations which characterize the feature from different perspectives [Rui99]. Among the general types of features are the following two important types:

1. Texture feature refers to the visual patterns that have properties of homogeneity that do not result from the presence of only a single color or intensity. It is an innate property of virtually all surfaces, including clouds, trees, bricks, hair, and fabric. It contains important information about the structural arrangement of surfaces and their relationship to the surrounding environment [Rui99]. The statistical methods for feature extraction are one of the early methods proposed in the literatures; they used to detect texels and the relationship among them. Some statistical quantities (like entropy, correlation, energy, contrast) have been utilized to describe the statistical behavior of the textural image [Wazi99].

   Energy tells us something about how the gray levels are distributed; it is usually determined by using the following:

$$\text{Energy} = \sum_{(r,c)} \left(D\left(I(r,c)\right)\right)^2 \qquad (2.2)$$

   Where D is the difference between two pixels value and I (r, c) is the pixel value.

Contrast is very important measure in the image processing which often determined the quality of an image [Abdu96]; Different mathematical definitions for contrast were appeared in the literatures the most popular definitions are: Contrast, is the difference in visual properties that makes an object (or its representation in an image) distinguishable from other objects and from the background. Other definition of contrast is the relative difference in intensity between an image point and its surroundings.

High contrast means the difference is great; low contrast, means the difference is little (e.g., image is mostly made up of gray areas, lacking white and/or black areas). In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view [Cont06].

The following equations represent how to compute the contrast:

$$C_1 = \frac{\sigma}{m} \tag{2.3}$$

$$C_2 = \frac{\sum_{(r,c)\in B_1} I(r,c)}{\sum_{(r,c)\in B_2} I(r,c)} \tag{2.4}$$

Where m is the mean, σ is the standard deviation, B_1 is the set of pixels whose values are larger than the median of the block, B_2 is the set of pixels whose values are smallest than the median of the block and I (r, c) is the pixel value.

2. Color features are most widely used as visual features in image retrieval. They relatively robust to background complication and independent of image size and orientation, the color histogram are one of the most important color features. Besides the color histogram (which is the most commonly used color feature representation), several other color feature representations have been applied in image retrieval, including color moments and color sets [Rui99].

2.8 Histogram Feature

Color histogram of an image is produced first by dividing the colors in the image into a number of bins, and counting the number of image pixels in each bin. The idea was proposed by Michael Swain and Dana Ballard in 1991 and is primarily used in situations where speed of processing is a factor in the choice of algorithm. Color histograms are a flexible constructs that can be built from images in various color spaces, whether RGB, or any other color space of any dimension [Rui99].

The most popularly used features are the color histogram features because the color histogram is computationally efficient and generally insensitive to small changes in camera position [Zhan02]. The histogram is a statistically based feature, where it is used as a model of the probability distribution of the gray levels [Umba98]. First order statistical features are extracted from the histograms of the three color channels (RGB) and the grey level histogram [Mari04].

These statistical features describe the gray level histogram without considering spatial independence [Mien02]. The first-order histogram probability is [Umba98]:

$$P(g) = \frac{n(g)}{M} \tag{2.5}$$

Where M is the number of pixels in the image or sub image (if the entire image is under consideration then M=N^2 for an N×N image), and n(g) is the number of pixels at gray-level g. Some of the features based on the first-order histogram probability are: Mean, Variance, Median, Skewness, Kurtosis, and Energy [Mien02].

The mean is the average values, so it tells us something about the general brightness of the image. The mean can be defined as [Umba98]:

$$\overline{g} = \sum_{g=0}^{L-1} gP(g) \tag{2.6}$$

The standard deviation which is also known as the square root of the variance tells us something about the contrast. And it is defined as follows [Umba98]:

$$\sigma_g = \sqrt{\sum_{g=0}^{L-1} (g - \overline{g})^2 P(g)} \tag{2.7}$$

The mean absolute deviation MAD is the average of the difference between pixels values and the average value [Umba98]:

$$\text{MAD} = \sum_{g=0}^{L-1} \left| (g - \overline{g}) P(g) \right| \tag{2.8}$$

The shape of the boundary segment can be described quantitatively by using the moments; the *nth* moment of *g* about its mean can be defined as [Gonz92]:

$$M_n(r) = \sum_{g=0}^{L-1} \left| (g - \overline{g})^n \right| P(g) \tag{2.9}$$

The second moment measures the spread of the curve about the mean value of *g* and the third moment measures its symmetry with reference to the mean. Both moment representations may used simultaneously to describe a boundary segment [Gonz92].

Since the color images consist of three color planes (red, blue, green), so it can be treated as three gray-scale images. This approach allows us to use any of the previously defined histogram features for three times, one for each color component.

2.9 Clustering

Clustering is unsupervised learning of a hidden data concept, it implies the division of data into groups of similar objects. Each group, called cluster, consists of objects that are similar between themselves and dissimilar to objects of other groups. Clustering differs from classification in that there is no target variable for clustering. The clustering task does not try to classify, estimate, or predict the value of a target variable. Instead, clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters, where the similarity of the records within the cluster is maximized and the similarity to records outside the cluster is minimized [Laro04].

Clustering is one of the most important tasks performed in Data Mining applications. A clustering algorithm attempts to find natural groups of components (or data) based on some similarity. The clustering algorithm also finds the centroid (like center of mass or center of gravity) of a group of data sets. To determine cluster membership, most algorithms evaluate a distance between a point and the cluster centroids. The output from a clustering algorithm is a statistical description of the clusters, centroids and the number of components in each cluster. There is more than one way to measure a distance. The most commonly used distance is the Euclidean measure, generally, the distance between the two points (in the feature space) is taken as a common metric to assess the similarity among the components of a population.

The Euclidian distance measure between two points's p= (p_1, p_2...) and q = (q_1, q_2...) is [Ciuc02]:

$$d = \sqrt{\sum_{t=1}^{k} (p_t - q_t)^2} \qquad (2.10)$$

Various clustering concepts have been appeared in the literature; they can be grouped into two classes according to the type of the partitioning structure imposed on the data:

1. **Hierarchical clustering:** the hierarchical approach produces a nested series of partitions consisting of clusters either disjoint or included one into the other [Peng04]. In hierarchical clustering the input data are not partitioned into the desired number of classes in a single step. Instead, a series of successive partitions of data are performed until the final number of clusters is obtained [Cuic02]. An example of hierarchical clustering algorithms is [Andr02]:

a. *Agglomerative:* algorithms start with each object being a separate cluster itself, and successively merge groups according to a distance measure. The clustering may stop when all objects are in a single group or at any other point the user wants.
b. *Divisive* algorithms follow the opposite strategy. They start with one group of all objects and successively split groups into smaller ones, until each object falls in one cluster, or as desired. Divisive approaches divide the data objects in disjoint groups at every step, and follow the same Patterns until all objects fall into a separate cluster.

In general, hierarchical algorithms can not provide optimal partitions for their criterion.

2. **Nonhierarchical clustering (partitional clustering) [Andr02]:** partitional clustering algorithm constructs partitions of the data, where each cluster optimizes a clustering criterion, such as the minimization of the *sum of squared distance from the mean* within each cluster.

One of the issues with such algorithms is their high complexity, as some of them exhaustively enumerate all possible groupings and try to find the global optimum. Even for a small number of objects, the number of partitions is huge. That's why; common solutions start with an initial, usually random, partition and proceed with its refinement. A better practice would be to run the partitional algorithm for different sets of initial ***k*** points (considered as representatives), and investigate whether all solutions lead to the same final partition. Partitional Clustering algorithms try to locally improve a certain criterion. First, they compute the values of the similarity or distance, they order the results, and pick the one that optimizes the criterion.

Some of partitional clustering algorithms include the first ones that appeared in the Data Mining Community [Andr02]:

a. *PAM (Partitioning Around Medoids)*
b. *K-means*

PAM is an extension to *k-means*, intended to handle outliers efficiently. Instead of cluster centers, it chooses to represent each cluster by its *medoid*. A medoid is the most centrally located object inside a cluster, the computational complexity of *PAM* is very large for large data.

K-means is an iterative, non-hierarchical algorithm for clustering very large data sets. It was developed in 1967 by J.B. MacQueen [Hohl01].

The algorithm starts by partitioning the input points into ***k*** initial sets, either at random or using some heuristic data. It then calculates the mean point, or centroid, of each set. It constructs a new partition by associating each point with the closest centroid. Then the centroids are recalculated for the new clusters, and these steps are repeated by alternate application of these two steps until convergence, which is obtained when the points no longer switch clusters (or alternatively centroids are no longer changed). The algorithm has remained extremely popular because it converges extremely quickly in practice. In fact, many have observed that the number of iterations is typically much less than the number of points. The quality of the final solution depends largely on the initial set of clusters, and may, in practice, be much poorer than the global optimum. Since the algorithm is extremely fast, a common method is to run the algorithm several times and return the best clustering found.

Another main drawback of the algorithm is that it has to be told the number of clusters ***k*** to find. If the data is not naturally clustered, you get some strange results. Also, the algorithm works well only when spherical clusters are naturally available in data [Kmea06].

The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments. It maximizes inter-cluster variance and minimizes intra-cluster variance, but does not ensure that the result has a global minimum of variance [Andr02].

The steps of the classic K-means clustering algorithm are [Peng04]:

1. Choose ***k*** cluster centers randomly generated in a domain containing all the points,
2. Assign each point to the closest cluster center,
3. Recompute the cluster centers using the current cluster memberships,
4. If the convergence criterion is met then stop; otherwise go to step 2.

2.10 AVI File [Msdn98]

The Microsoft audio-video interleaved (AVI) file format is a resource interchange file format (RIFF) file specification used with applications that capture, edit, and play back audio-video sequences. In general, AVI files contain multiple streams of different types of data. Most AVI sequences use both audio and video streams. A simple variation for an AVI sequence uses video data and does not hold an audio stream.

AVI files use the AVI RIFF format. The AVI RIFF formation is identified by the **FOURCC** (four-character code) 'AVI '. All AVI files include two mandatory LIST chunks. These chunks define the format of the stream and stream data. AVI files might also include an index chunk. This optional chunk specifies the location of data chunks within the file. Figure (2.3) shows the typical structure of an AVI file with these components.

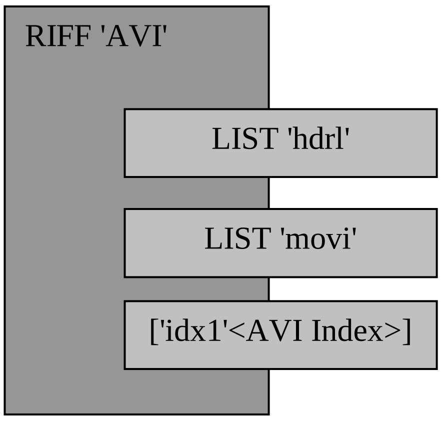

Figure (2.3) The AVI file chunks [Msdn98]

The LIST chunks and the index chunk are subchunks of the RIFF 'AVI ' chunk. The 'AVI ' chunk identifies the file as an AVI RIFF file. The LIST 'hdrl' chunk defines the format of the data and usually is the first required LIST sub-chunk. The LIST 'movi' chunk contains the data for the AVI sequence and is the second required LIST sub-chunk. The 'idx1' sub-chunk is the index chunk. AVI files must keep these three components in the proper sequence.

Figure (2.4) show an example of the AVI RIFF form expanded with the chunks needed to complete the LIST 'hdrl' and LIST 'movi' chunks. For more details about the AVI file format see the appendix (A).

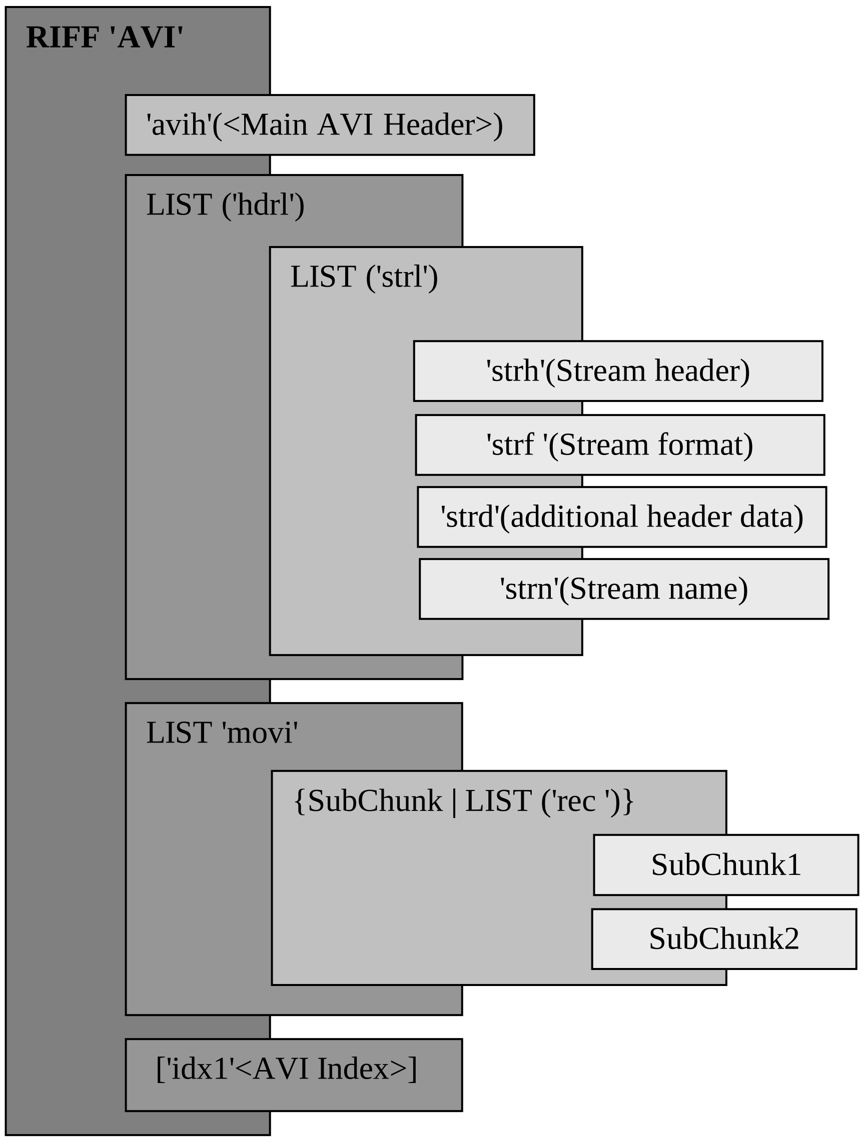

Figure (2.4) The LIST 'hdrl' and LIST 'movi' chunks [Msdn98]

2.11 Image File Format

In computer graphics, types of image data are divided into two primary categories: Bitmap and vector. Bitmap images are represented by the RGB color image model, where the pixel data $\{I_R(r,c), I_G(r,c), I_B(r,c)\}$ stored successively in the file [Umba98].

The bitmap structure defines the type, width, height, color format, and bit values of a bitmap [Msdn98].

Every BMP file consists of three parts; First part is the header which begins with a *bitmapfileheader* which contains information about file type and size also it specifies the location of the pixel data in the file. *Image header* followed the *bitmapheaderfile*, the *image header* contains information about the image (like width, height), second part is the color palette (if it is exists) which is followed the file header and it contains information about colors value of the image pixel, and last part is the image data (represent the pixels values).

Most of the types of file format fall into the category of Bitmap images some of the format uses compression, so that the I(r,c) values are not directly available until the file is decompressed. some of more complex file formats, the header may contain information about the type of compression used and any other necessary parameters to create the image, I(r,c). For more information about the image file format see the appendix (B).

Certification of the Examination Committee

We chairman and members of the examination committee, certify that we have study the thesis entitle (***Video Data Mining Using Color and Textural Features Analysis***) presented by the student **Hind Ali Al-Kitt** and examined her its content and in what is related to it, we have found it worthy to be accepted for the degree of Master of Science in Computer Science.g

Signature:
Name**: Dr. Imad H. Al- Hussaini**
Title**: Assistant Professor**
Date**: / / 2006**
(Chairman)

Signature**:**
Name**: Dr. Ali Abid D. Al- Zuky**
Title**: Assistant Professor**
Date**: / / 2006**
(Member)

Signature**:**
Name**: Dr. Taha S. Bashaga**
Title**: Lecturer**
Date**: / / 2006**
(Member)

Signature**:**
Name**: Dr. Loay A. George**
Title**: Senior Researcher**
Date**: / / 2006**
(Supervisor)

Signature**:**
Name**: Dr. Laith A. Al Ani**
Title**: Dean of College of Science**
Date**: / / 2006**

References

[Abdu96]

L. Abdul Aziz, "*Classification of digital Satellites image*", (PhD) Thesis, College of Science, Al-Nahrain University, Iraq, 1996.

[Andr02]

P. Andritsos, "*Data Clustering Techniques*" Paper, Department of Computer Science, University of Toronto, Canada, 2002.

[Band02]

J. Oh and B. Bandi, "*Multimedia Data Mining Framework for Raw Video Sequences*", In Proc. of ACM Third International Workshop on Multimedia Data Mining (MDM/KDD2002), Edmonton, Alberta, Canada, 2002.

[Band03]

J. Oh, J. Lee, S. Kote, and B. Bandi, "*Multimedia Data Mining Framework for Raw Video Sequences*", Department of Computer Science and Engineering, University of Texas at Arlington, 2003.

[Bane00]

S. Banerjee, "*Motion Estimation and Compensation of H.263 Video*", Technical report, Department of Electrical and Computer Engineering, University of Texas, 2000.

[Bolt06]

S. Boltz, E. Wolsztynski, E. Debreuve, E. Thierry, M. Barlaud and L. Pronzato, "*A Minimum-Entropy Procedure for Robust Motion Estimation*", In proceedings, International Conference on Image Processing – ICIP, Atlanta, Georgia, USA, 2006**.**

[Chan99]

Y. Rui, T. Huang, S. Chang, " *Image Retrieval: Current Techniques, Promising Directions, and Open Issues*", Journal of Visual Communication and Image Representation 10, 39–62, 1999.

[Ciuc02]

M. Ciucu, P. Heas, M. Datcu and J. Tilton**,** "*Scale Space Exploration for Mining Image Information Content*", In Proc. of ACM Third International Workshop on Multimedia Data Mining (MDM/KDD2002), Edmonton, Alberta, Canada, 2002.

[Colo03]

"*Color Images*", 2003

http://homepages.inf.ed.ac.uk/rbf/HIPR2/Glossary-Color Images.htm

[Colo06]

"*Color*", 2006

http://en.wikipedia.org/wiki/Color

[Comp87]

"*Computer Vision IT412*", School of Computer Science, Software Engineering 1987

http://www.undergraduate.csse.uwa.edu.au/courses/233.412

[Cont06]

"*Contrast (vision)*", 2006

http://en.wikipedia.org/wiki/Contrast_(vision)

[Data05]

"*Data mining*", internet survey 2005

http://www.megaputer.com/dm/dm101.php3#whyuse.

[Datc02]

M. Datcu, K. Seidel, "*An Innovative Concept for Image Information Mining*", In Proc. of ACM Third International Workshop on Multimedia Data Mining (MDM/KDD2002), Edmonton, Alberta, Canada, 2002.

[Fern03]

J. mas and G. Fernandez, "*Video Shot Boundary Detection Based on Color Histogram*", paper, Digital Television Center La Salle School of Engineering, Ramon Llull University, Spain, 2003.

[Fosc01]

P. Foschi, D. Kolippakkam, H. Liu and A. Mandvikar, "*Feature Extraction for Image Mining*", Workshop on Multimedia Information Systems, DOCIS Documents in Computing and Information Science, 2001.

[Gonz92]

R. Gonzalez and R. Woods, "*Digital Image Processing*", Addison Wesley Publishing Company, 1992.

[Gorm99]

G. Gormley, "*Scene Break Detection & Classification in Digital Video Sequences*", Technical Report, School of Electronic Engineering, Dublin City University, Ireland, 1999.

[Gyao03]

A. Gyaourova, C. Kamath, S. Cheung, "*Block matching for object tracking*", research, Department of Energy's (DOE), Office of Scientific and Technical Information (OSTI), 2003**.**

[Han98]

O. Zaiane, J. Han, Z. Li, J. Hou, "*Mining multimedia data*", Proceedings CASCON'98: Meeting of Minds, Toronto, Canada, 83-96 1998.

[Haup02]

A. Hauptmann, R. Yan, Y. Qi, R. Jin, M. Christel, M. Derthick, M. Chen, R. Baron and W. Lin, "*Video Classification and Retrieval with the Informedia Digital Video Library System*", paper, NIST Special Publication: SP500-251, The Eleventh Text Retrieval Conference (TREC), 2002.

[Hohl01]

B. Hohlt, "*Pthread Parallel K-means*", CS267 Applications of Parallel Computing, UC Berkeley, 2001.

[Kmea06]

"*K-means algorithm*", 2006

http://en.wikipedia.org/wiki/K-means_algorithm

[Kuan03]

W. Kuang Li and S. Hong Lai, "*Integrated video shot segmentation algorithm*", research, Department of Computer Science, National Tsing-Hua University, Taiwan, 2003.

[Laro04]

D. Larose, "*Discovering Knowledge in Data: An Introduction to Data Mining*", Published by John Wiley & Sons, Inc., 2004.

[Mali05]

H. Malik, "*iARM: Image Association Rule Mining Language*", COMS W4115: Programming Languages and Translators, Department of Computer Science, Columbia University, 2005.

[Mari04]

N. Marios, P. Constantinos, P. Marios, T.Vasilios, K. Efthyvoulos and K. Dimitris, "*Multiscale Texture Feature Variability Analysis in Images during Laparoscopy under Different Viewing Positions*", Medical Informatics Laboratory, Department of Computer Science, University of Cyprus, 2004.

[Mien02]

A. Miene, T. Hermes, G. Ioannidis, R. Fathi, and O. Herzog, "*Automatic Shot Boundary Detection and Classification of Indoor and Outdoor Scenes*", paper, NIST Special Publication:SP 500-251,The Eleventh Text Retrieval Conference (TREC), 2002.

[Msdn98]

MSDN Library, Visual Studio 6.0 release, 1998

[Moti00]

"*Welcome to the Motion Estimation Tutorial*", 2000, http://stargate.ecn.purdue.edu/~ips/tutorials/me/ .

[Peng04]

J. Peng and Y. Xia**,** "*A new theoretical framework for K-means-type Clustering*", Advanced Optimization Lab, Department of Computing and Software, McMaster University, 2004.

[Rgb06]

"*RGB color model*", 2006 http://en.wikipedia.org/wiki/RGB_color_model

[Smea99]

A. Smeaton, J. Gilvarry, G. Gormley, B. Tobin S. Marlow and N. Murphy, "*An Evaluation of Alternative Techniques for Automatic Detection of Shot Boundaries in Digital Video*", Paper, Centre for Digital Video Processing, Dublin City University, Ireland, 1999.

[Smea00]

A. Smeaton, P. Browne, N. Murphy, N. O'Connor, S. Marlow and C. Berrut, "*Evaluating and Combining Digital Video Shot Boundary Detection Algorithms*" paper, Centre for Digital Video Processing, Dublin City University, Ireland, 2000.

[Umb98]

S. Umbaugh, "*Computer Vision and Image Processing- A Practical Approach Using CVIP Tools*"**,** Prentice-Hall, Inc., 1998.

[Watk94]

J. Watkinson**,** "*The Engineer's Guide to Motion Compensation*"**,** Published by Snell and Wilcox Ltd. Durford Mill, Petersfield Hampshire GU13 5AZ, 1994.

[Wazi99]

V. Wazir, "*An Investigation into the use of neural network in texture classification*", (PhD) Thesis, college of Science, Al-Nahrain University, Iraq, 1999.

[What01]

"*What is data mining*", An internet survey 2001 http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm

[Zaia98]

O. Zaiane, J. han and Z. Nian, "*multimediaminer: A System prototype for multimedia data mining*", Proceedings of ACM-SIGMOD International Conference on Management of Data, pp. 581 – 583, 1998.

[Zaia99]

O. Zaiane , E. and J. Han, "*Word Taxonomy for On-line Visual Asset Management and Mining*", Fourth International Workshop on Application of Natural Language to Information Systems, pp 271-276, Klagenfurt, Austria,, 1999.

[Zaia00]

O. Zaiane, J. Han and H. Zhu, "*Mining Recurrent Items in Multimedia with Progressive Resolution refinement*", Int. Conf. on Data Engineering (ICDE'2000), pp. 461-470, San Diego, CA, 2000.

[Zhan01]

J. Zhang, W. Hsu and M. Lee, "*Image Mining: Issues, Frameworks and Techniques*", Proceedings of the Second International Workshop on Multimedia Data Mining (MDM/KDD'2001), in conjunction with ACM SIGKDD conference. San Francisco, USA, 2001.

[Zhan02]

R. Zhang, Z. Mark and T. Watson, "*A Clustering Based Approach to Efficient Image Retrieval*"**,** 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02) p. 339, 2002.

[Zhon00]

D. Zhong and S. Chang, "*Video Shot Detection Combining Multiple Visual Features*", research, Department of Electrical Engineering, Columbia University, 2000.

[Zhu04]

X. Zhu and X. Wu, "*Sequential Association Mining for Video Summarization*", research, Department of Computer Science, University of Vermont, 2004.

Republic of Iraq
Ministry of Higher Education and scientific research
Al- Nahrain University

Video Data Mining Using Color and Textural Features Analysis

A Thesis

Submitted to the

College of Science, Al-Nahrain University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Computer Science

By

Hind Ali Hussain Al-Kitt

(B.Sc.2003)

Supervisor

Dr. Loay A. George

2006 1247

Table of contents

الاهداء

الى عائلتي الغالية ابي وامي
واخوتي
شكراً لكل ما قدمتموه لي من
حب وتضحيات

الى روح الشهيدة الغالية
الست ***منى جليل*** تغمدها الله
برحمته الواسعة

هند

بـــــــــــم الله
الــرحـــمـــن الــرحـــــــيـــم

وَ يسألونكَ عَنِ الروحِ قُلِ الروحُ مِن أمرِ
رَبي وَ ما أوتَيتُم مِنَ العِلم ◌ِ إلا قَلِيلاً

صدق الله الـعظــيم

سورة الإسراء
آية (٨٥)

وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كلية العلوم

التنقيب عن بيانات الفديو بأستعمال التحليل للخصائص اللونية والنسجية

رسالة مقدمة إلى كلية العلوم، جامعة النهرين كجزء من متطلبات نيل شهادة
الماجستير في علوم الحاسبات

من قبل

هند علي حسين الكط

بكالوريوس

2003

المشرف

د. لؤي ادوار جورج

```
[.ShellClassInfo]
LocalizedResourceName=@%SystemRoot%\system32\shell32.dll,-21815
```