ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

DEPARTMENT OF COMPUTER SCIENCE

Content Dynamics Differentiation for MPEG Video

Master Thesis Report

Kourtis Michail-Alexandros

Under the supervision of Dr.Koumaras Harilaos,

Dr. Xylomenos Georgios and Dr. Polyzos Georgios

Abstract: This report presents a method for content dynamics differentiation of MPEG video. The method relies upon the extraction of motion information from the video signal in order to define the content's spatiotemporal dynamics levels. For this purpose, this reports introduces a motion compensation and content residual grid, on where each video signal is represented as a point, whose coordinates are defined by its motion dynamics and content residual, respectively. The proposed method can be effectively integrated in video adaptation systems, which perform spatial or temporal (or both) adaptation actions, in order to optimize the video service delivery. Such a method for measuring and quantifying the dynamics of a video signal is crucial for the decision making process of systems like MPEG-DASH.

Keywords: Video Differentiation, Motion Vector Analysis, Video Dynamics, H.264.

Table of Contents

1. INTRODUCTION

In recent years, due to the ever-growing demand for multimedia services, a vast and increasing request for video content services has been observed. This has created a substantial need for high-quality, trust-worthy and efficient video services. However, this proved to be a complex and difficult task, due to the fact that the video stream is usually transmitted over error-prone IP network environments, whose network impairments (e.g. packet loss, delay and delay variation for IP networks) may cause quality degradation. In order to meet the high-end user demands, and simultaneously tackle effectively the obstacles created by the heterogeneous network environment, several solutions have been proposed.

One of the proposed solutions is the video adaptation framework using the MPEG standard Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [1]. Depending on network conditions or user feedback, the video delivery stream is dynamically switched seamlessly among different pre-encoded streams, which have been adapted either in the spatial (i.e. resolution) or the temporal (i.e. frame rate) domain.

For the decision making process of MPEG-DASH and for the successful preparation of the video signal at various spatial or temporal adapted versions, it is crucial to differentiate the spatiotemporal video content dynamics of the test signal. For example, video content with high temporal and low spatial dynamics will be significantly degraded by a temporal adaptation decision action (introducing jerky motion and pauses), thus spatial adaptation should be preferred instead. Conversely, video signals with high spatial and low temporal dynamics, should be adapted by applying a temporal adaptation process in order to minimize the perceptual impact of the adaptation action, and vice versa.

Therefore, it is necessary for the optimization of the decision making process of adaptive systems, like MPEG-DASH, to consider in the adapted stream preparation/encoding process, appropriate tools that differentiate the video content dynamics of the signal both at the spatial and temporal domain.

This report proposes a method to quantify and depict the spatiotemporal video content dynamics of each video signal under test. In order to do that a mapping of the spatial and temporal domains on a motion compensation and content residual grid is necessary. This is achieved by utilizing the signal's motion vectors. The content residual metric refers to the video's average motion vector length, thus the activity level of the video's elements, meaning the shift between successive frames. The motion compensation metric measures the average motion vector number of the video, thus depicting the overall motion activity of a video sequence, how many macroblocks changed position during the video sequence. By positioning the measured indexes on the motion compensation and content residual grid, it is possible for each video signal to be differentiated to four categories depending on its content dynamics, namely:

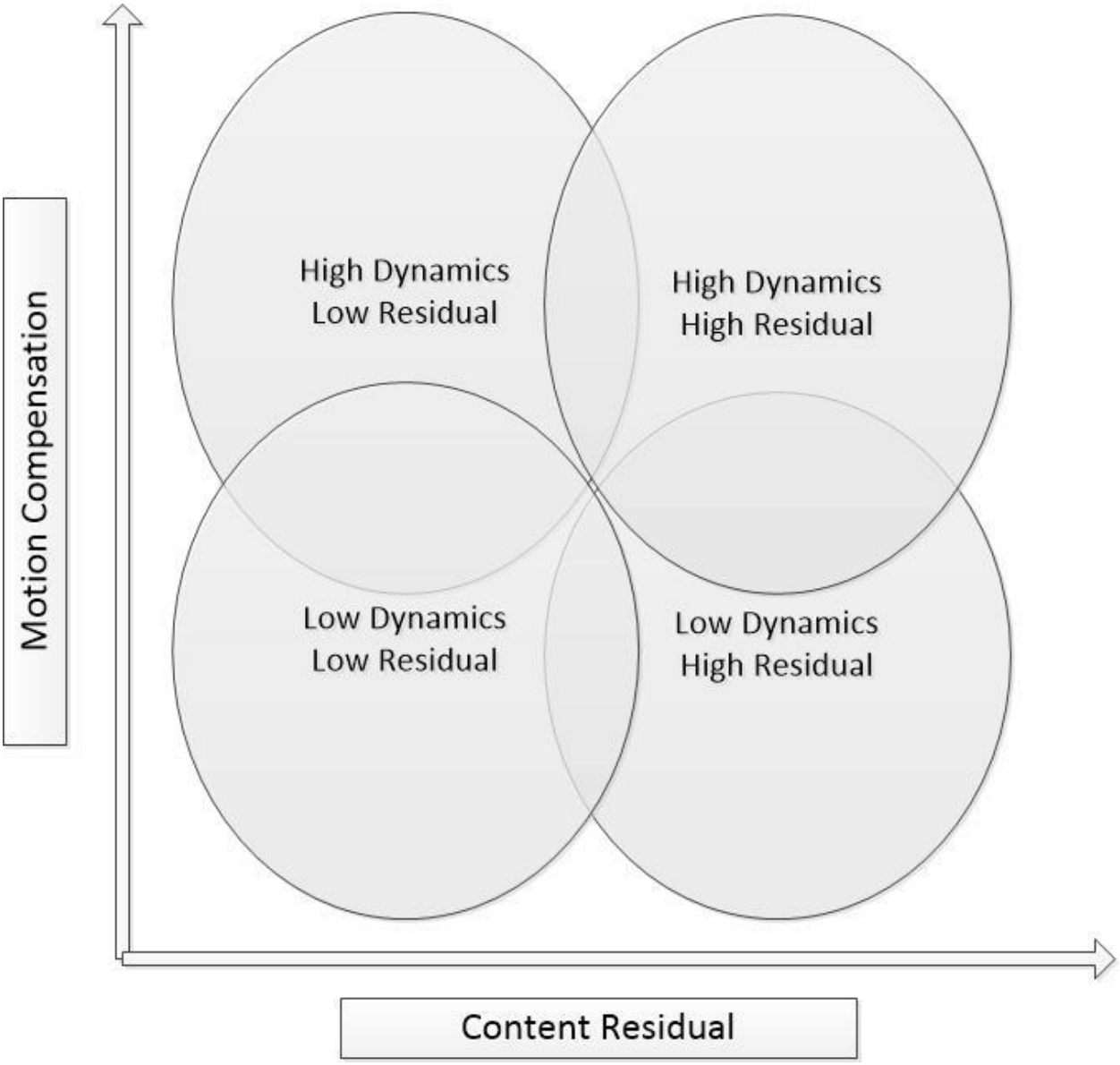

Figure 1 Video Differentiation Plot

- Low Dynamics – Low Residual, which is defined as the lower left quarter in the grid.
- High Dynamics – Low Residual, which is defined as the upper left quarter in the grid.
- Low Dynamics – High Residual, which is defined as the lower right quarter in the grid.
- High Dynamics – High Residual, which is defined as the higher right quarter in the grid.

For the classification of each test signal, we use two discrete metrics for quantifying the content residual and motion compensation components of its content.

The rest of the report is organized as follows: Section 2 presents some background information and the methodology used, analyzing its individual steps in detail. The experimental results of the proposed technique are discussed thoroughly in Section 3. Finally, Section 4 concludes the report and presents the overall contribution in its field, and Section 5 further discusses the future work and perspectives of the current research outcomes.

2. OVERVIEW OF VIDEO CODING

2.1 Principles of Video Coding

All forms of video-coding that have compression as a primary goal, try to minimize redundancy in the media. A video consists of a number of frames, meaning separate pictures, which when projected one after the other at a particular rate, they give the human eye the feeling of continuous movement. This leads us to the fact that we can have 2 kinds of redundancy, spatial and temporal. Spatial redundancy refers to intraframe coding techniques, which means that we use neighboring similar pixels of the same frame to encode it. Temporal redundancy has to do with interframe coding, meaning the usage of past and future frames to encode our current frame.

Therefore video compression techniques are divided into 2 categories based on the redundancy type. The temporal phase exploits the similarities between successive frames, aiming to reduce the temporal redundancy in a video sequence. The spatial stage exploits spatial similarities located on the same frame, reducing in this way the spatial redundancy. Then the output parameters of the temporal and spatial stages are further quantized and compressed by an entropy encoder, which removes the statistical redundancy in the data, producing an even more compressed video stream. Thus, all the video coding standards are based on the same basic coding scheme, which briefly consists of the following phases: the temporal, the spatial, the transform, the quantization and the entropy coding phase.

Finally, it must be also noted that in more simplistic systems every frame is coded separately, using only intraframe redundancy, so that the loss of one frame will not affect the coding of the other frames. Due to the simplicity of these systems, this methodology is not analyzed in the chapter, because it is a part of the more complicated coding systems that use both spatial and temporal techniques in order to achieve greater effectiveness and efficiency in video compression ratio.

2.2 Compression at the Temporal Plane

As input to the temporal stage of the encoding process, the uncompressed video sequence is used, which contains a lot of redundancy between successive frames. The scope of this stage is to remove this redundancy by constructing a prediction of each frame based on previous or future frames, enhanced by compensating for fine differences between the selected reference frames. Depending on the prediction technique by which it is constructed, each frame is classified to three discrete types, namely, Intra-frame (I), Predictive (P) and Bidirectional predictive (B), widely referred as I, P and B. The I frames are also called Intra frames, while B and P are known as Inter frames.

- I frames do not contain any prediction from any other coded frame.
- P frames are coded based on prediction from previously encoded I or P frames.
- B frames are coded based on prediction from previously or future encoded I or P frames.

The pattern of successive types of frames like IBBPBBPBBP… forms a Group of Pictures (GOP), whose length is mainly described by the distance of two successive I frames (see Figure 2).

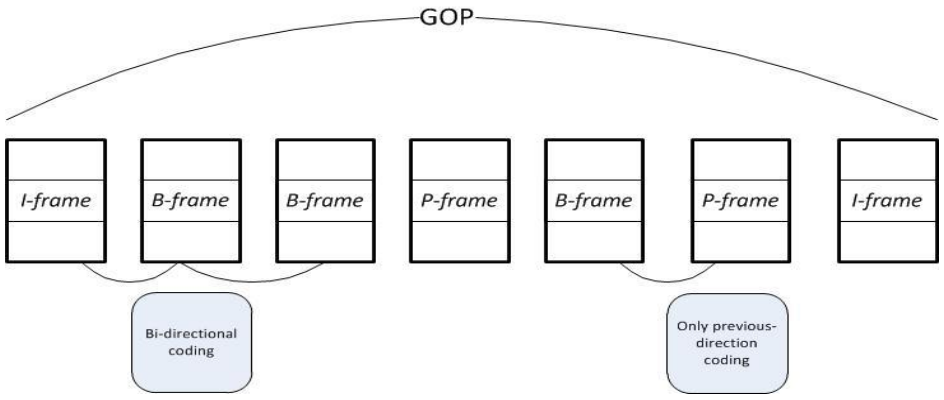

Figure 2 GOP Structure

In order to perform temporal compression, two discrete processes are performed: motion estimation and motion compensation. Both these processes are usually applied on specific rectangular regions of a frame, called blocks if their size is 8x8 pixels or MBs if they are 16x16 rectangular pixel regions. In the latest standards (i.e. H.264) variable block sizes are used for motion compensation depending on the content, as Figure 3 depicts, achieving better coding efficiency.

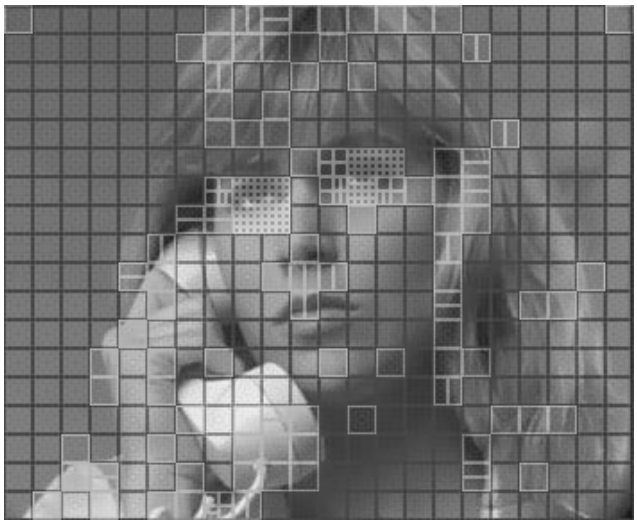

Figure 3 ***Example of variable block size coding***

During motion estimation, the encoding algorithm searches for an area in the reference frame (past or future frame) in order to find a corresponding matching region. The process of locating the best match between a current frame and a reference one, which will be used as a predictor of the current frame, is called motion estimation. This is performed by comparing specific rectangular areas (i.e. blocks/MBs) in the reference and current frame, until the best match is detected. For this purpose, their spatial differences are calculated, using the Sum of Absolute Differences (SAD) or Sum of Absolute Errors (SAE), which is defined as:

$$SAD(d_x, d_y) = \sum_{i=0}^{15} \sum_{j=0}^{15} | f(i, j) - g(i - d_x, j - d_y) |$$

where *f(i,j)* and *g(i,j)* denote the luminance pixels of the current rectangular area (in this case a MB) and the reference one, respectively. The reference area is relatively defined by the current one using the motion vectors (d_x,d_y), denoting the position of the best matching region (see figure 4).

When the best match has been performed, then motion compensation follows. During this process the selected optimal matching region in the reference frame (i.e. the region that sets the SAD minimum) is subtracted from the corresponding region in the current frame so as to produce a luminance and chrominance residual block/MB that is transmitted and encoded

along with the reference motion vectors. The deduced frame by the motion compensation process is called residual frame, which contains the result of the subtraction of the reference regions from the corresponding ones of the current frame. In the residual frame the static areas correspond to a difference equal to zero, while darker areas denote negative differences and lighter areas positive differences respectively. A typical example of a residual frame is represented in figure 5.

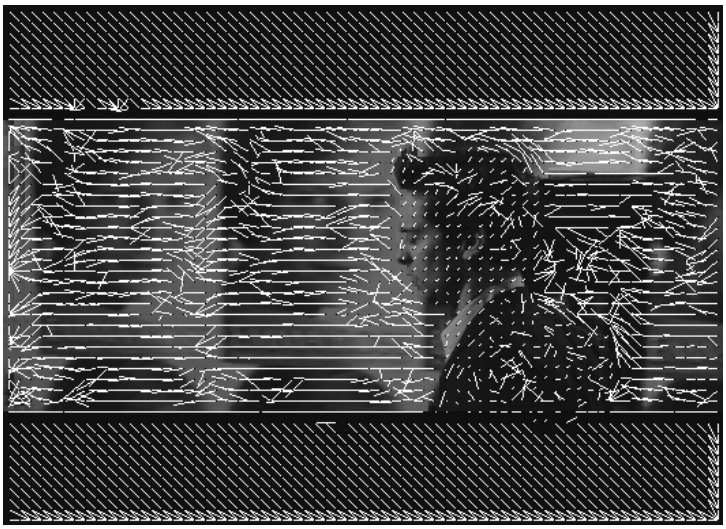

Figure 4 ***A frame where motion vectors appear denoting the position of the best matching region***

Figure 5 ***A residual frame (on the left) denoting the differences between two successive frames for the fireman reference sequence (shown on the right).***

Thus motion compensation enhances the efficiency of the motion estimation by adding at the predicted frame the fine differences that may contain the motion estimated predicted regions

in comparison to the actual frame. Adding this motion compensated residual information on the motion estimated frame, an accurate and efficient prediction of the current frame can be performed, using regions of past or future frames.

2.3 Compression at the Spatial Plane

Similarly to the temporal stage, where predictive coding is performed between successive frames, a prediction of an image region may be also performed based on samples located within the same image or frame, which is usually referred as Intra coding. At the spatial stage, the encoder performs a prediction for a pixel based pattern on combination of previously-coded pixels located on the same frame. Especially for frames that contain homogeneous areas, the spatial prediction can be quite efficient. The process in the terminology of video coding is call intracoding prediction. In the case of a good prediction then the residual energy is small and the corresponding compression ratio high.

More specifically, the spatial stage predicts the pixels of the current block using reconstructed blocks of neighboring blocks interpolated along different orientations, which results in a closely characteristics related image correlation. The computational complexity of this process significantly increases because of the number of different block partitioning modes and prediction directions. In order to reduce the computational resources needed, various complexity reduction strategies, along with novel hardware accelerators are used.

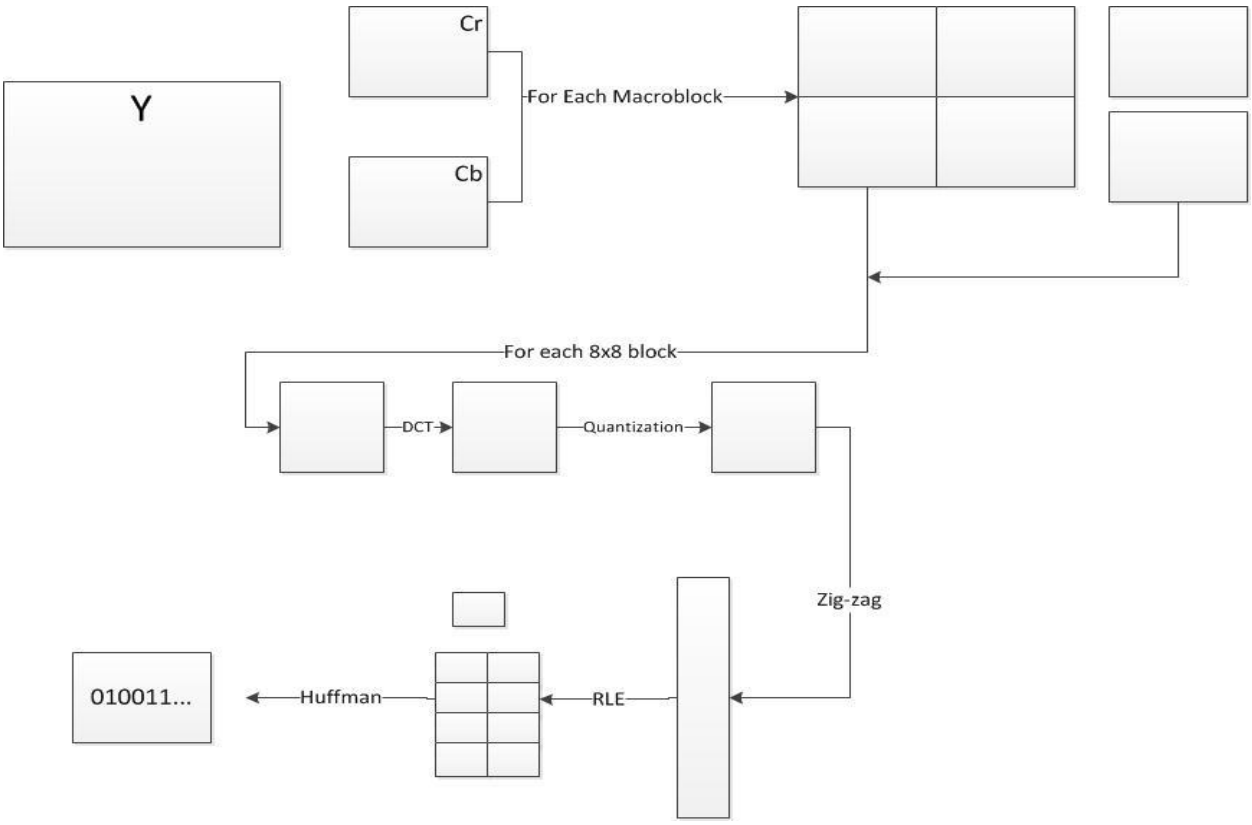

Figure 6 ***Steps of the Intra-Coding Prediction***

Based on the input signal, each frame is divided into Macroblocks (MBs), which are 16x16 pixel areas on Y plane of original frame. Each MB unit consists of 4 Y blocks, 1 Cr block and 1 Cb block. So the steps for Spatial Plane Compression (as it is called Intra Coding Prediction) are depicted on Figure 6 and are the following:

- Sub-divide picture into 16x16 pixel blocks: Macroblocks
- Apply DPCM, i.e. intra-prediction of the (16) 4x4 pixel blocks inside one Macroblock
- Residual transform (e.g. DCT algorithm), quantization and redundancy reduction

In natural sequences, the optimal intraprediction mode usually represents the texture direction in a block. Therefore, directional prediction can reduce the texture redundancy significantly. H.264/AVC introduces directional intraprediction in the spatial domain. Intra_4x4 and Intra_8x8 both have up to nine prediction modes, namely mode 0 - mode 8. Except for the DC mode (mode 2), the other eight modes correspond to different prediction directions as illustrated in Figure 7.

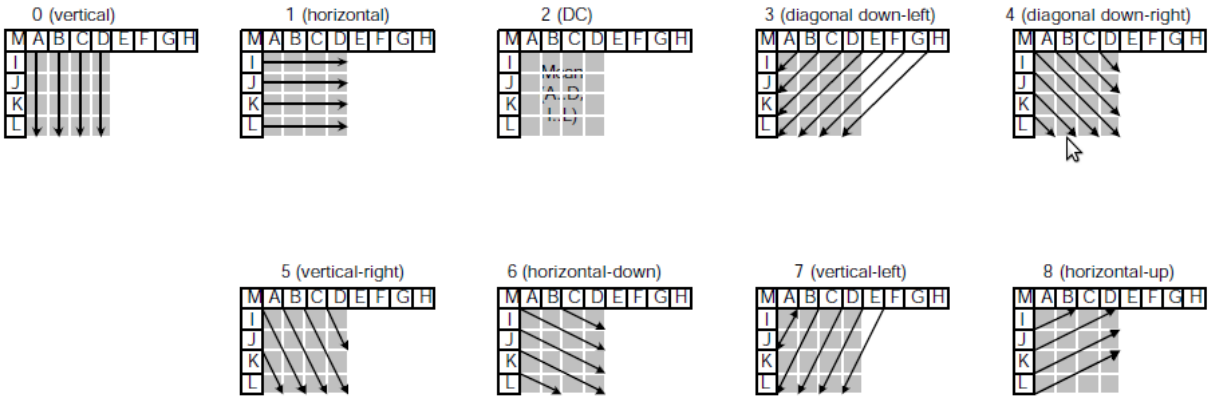

Figure 7 *H.264/AVC Intracoding Prediction Modes*

An example of intracoding prediction is depicted on Figure 8 (Richardson, 2003)[10], where a specific macroblock identified with white border has been selected to be intracoded predicted.

Figure 8 *H.264/AVC prediction example*

Based on the possible predictions, the intracoding algorithm will select the best mode by choosing the one with the smallest SAE. In addition, if multiple modes have the same, smallest SAE, then multiple solutions exist.

2.4 Transform Coding Phase

At this stage the spatially/temporally encoded frames or the motion-compensated residual data are converted into another domain, usually called the transformed domain, where the optically correlated data become decorrelated. The use of transformation facilitates the exploitation in the compression technique of the various psycho-visual redundancies by transforming the picture to a domain where different frequency ranges with dissimilar sensitivities in the Human Visual System (HVS) can be accessed independently (Winkler, 2005)[11].

The most commonly used transformation is the Discrete Cosine Transformation (DCT). The DCT operates on an X block of N × N image samples or residual values after prediction and creates Y, which is an N × N block of coefficients. The action of the DCT can be described in terms of a transform matrix A. The forward DCT is given by:

$$\mathbf{Y=AXA^T}$$

Where X is a matrix of samples, Y is a matrix of coefficients and A is an N X N transform matrix. The elements of A are:

$$A_{ij} = C_i \cos\frac{(2j+1)i\pi}{2N} \qquad \text{Where} \quad C_i = \begin{cases} \sqrt{1/N}, & i=0 \\ \sqrt{2/N}, & i>0 \end{cases}$$

Therefore the DCT can be written as:

$$Y_{xy} = C_x C_y \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} X_{ij} \cos\frac{(2j+1)y\pi}{2N} \cos\frac{(2i+1)x\pi}{2N}$$

The advantage of the DCT transformation is that it is possible to reconstruct quite satisfactorily the original image, applying the reverse DCT on a subset of the most important DCT coefficients, without taking under consideration the coefficients with insignificant magnitudes (see figure 9).

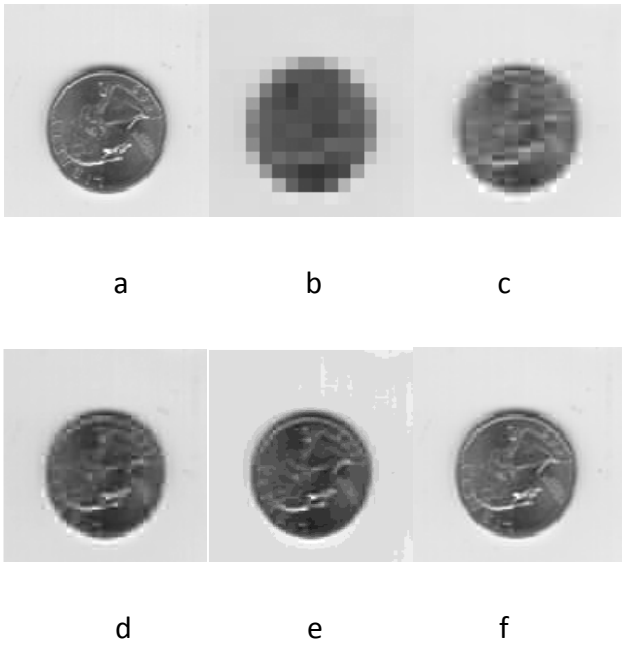

Figure 9 ***Example of DCT efficiency. Figure a is the source image, while b is reconstructed using only 1 DCT coefficient, b exploits 4 coefficients, c uses 8 coefficients, d uses 12, e uses 18 and f 32 out of the 64 total DCT Coefficients for each block (i.e. 8x8).***

Thus, at the cost of some quality degradation, the original image can be satisfactorily reconstructed with a reduced number of coefficient values. This DCT property is exploited by the following stage where quantization of the DCT coefficients is performed.

2.5 Quantization Phase

Quantization is the process of approximating the continuous range of DCT coefficients by a relatively-small set of discrete integer values. The best-known form of quantization is the scalar quantizer, which maps one sample of the input to one quantized output value. A scalar quantization operator *Q()* can be mathematically represented as:

$$Q(x) = g(\lfloor f(x) \rfloor)$$

Where

- x is a real number
- $\lfloor x \rfloor$ is the floor function
- *f(x)* and *g(i)* are arbitrary real-valued functions.

The integer value $i = \lfloor f(x) \rfloor$ is the representation that is usually stored or transmitted, but the final interpretation may be further modified by *g*(*i*). Thus, typically during a scalar quantization process, a rounding of a fractional number to its nearest integer is performed:

$$Y = QP\ round\left(\frac{X}{QP}\right)$$

Where QP is the quantization parameter (i.e. quantization step size), X the initial integer value and Y the deduced quantized number. Table 1 depicts some representative examples of the scalar quantization process for various Quantization Parameters.

X	*Y*		
	QP=1	*QP=2*	*QP=3*
0	0	0	0
1	1	0	0
2	2	2	3
3	3	2	3
4	4	4	3
5	5	4	6
6	6	6	6

7	7	6	6
8	8	8	9
9	9	8	9

Table 1: Quantization Examples

Applying quantization on the aforementioned DCT coefficients is the main reason for the quality degradation and the appearance of artifacts, like blockiness, in digitally encoded videos. The blockiness effect refers to a block pattern of size 8x8 pixels in the compressed sequence, which is the result of the independent quantization of individual blocks of block-based DCT. Due to the quantized DCT coefficients, within a block (8x8 pixels), the luminance differences and discontinuities between any pair of adjacent pixels are reduced. On the contrary, for all the pairs of adjacent pixels, which are located across and on both edge sides of the border of adjacent DCT blocks, the luminance discontinuities are increased by the coding process. This happens because the quantization process is lossy (i.e. not totally reversible) since it is not possible to determine the accurate fractional number from the deduced rounded integer. So it is somewhat equivalent with the case of not exploiting the entire DCT coefficient set for the reconstruction of the original image, as in Figure 4, because some low DCT values may have been quantized to zero.

It must be noted that the quantization stage is the only lossy stage at the described coding chain and is mainly responsible for any visual artifact and quality degradation, which may appear on the coded video signal.

2.6 Entropy Coding Phase

At this final stage a transformation of the video sequence symbols into a compressed stream is performed. The term video sequence symbol stands for all the aforementioned encoding parameters, such as quantization coefficients, motion vectors etc. Two widely known variable

length coding techniques are exploited at this stage: Huffman Coding (Huffman, 1952)[13] and Arithmetic Coding (Witten et al., 1987)[14]. Both methods are briefly analysed hereafter.

Huffman coding creates a binary tree of nodes. Each node contains the symbol itself, and the weight (appearance frequency) of the symbol, and optionally a link to a parent node. The internal nodes contain the symbol weight, links to their child nodes (two), and a link to the parent node to make the reverse tree reading easier. Usually bit ’0’ represents the left child and bit ’1’ the right child. The basic algorithm to create the Huffman binary tree is:

1. Create a node for every symbol, and place all of them in a set.
2. Remove from the set the nodes with the smallest weight, make them children of a new node that will have the weight sum of the two nodes, and place the new node to the set.
3. Repeat recursively step (2) until the set contains one node with the weight sum of all nodes.

An example is given in Figure 10. The encoding then begins by the root of the binary tree. For each symbol we want to encode we memorize the ‘0’,’1’ bits we encounter on the route to the symbol, and the string of ‘0’ and ‘1’ we have memorized is the encoded symbol.

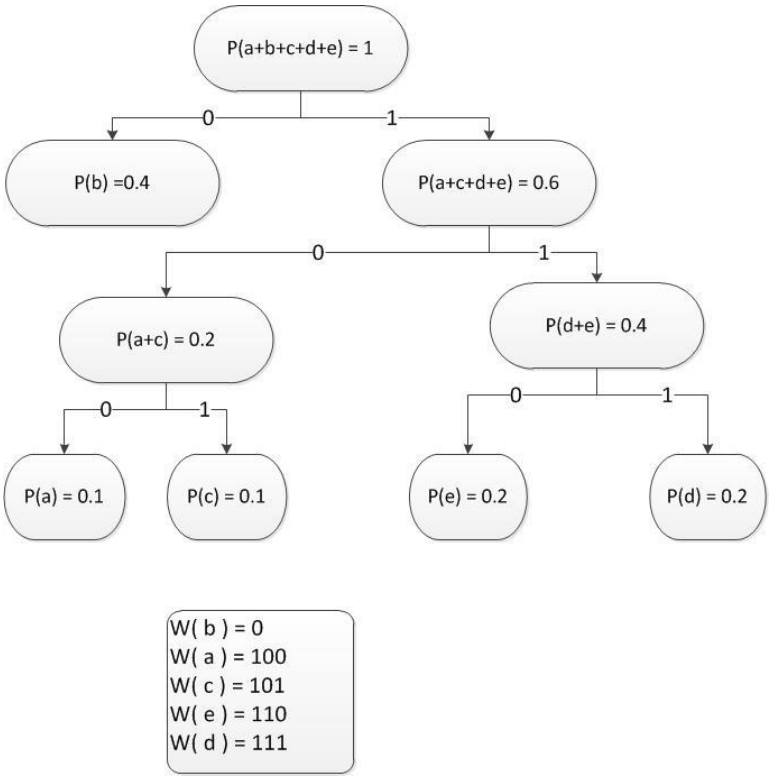

Figure 10 ***Huffman Coding Algorithm***

Arithmetic coding differs from Huffman coding, because it does not separate the input into component symbols and replace each with a code, it encodes the entire message into a single number, a fraction f where (0.0 =< f <1.0). Apart from the weight of its symbol arithmetic coding demands that the terminating symbol does not appear in any other position of the input string. The coding process is the following:

1. All n symbols of input s are sorted, usually in alphabetical order.
2. Every symbol x_i with weight $p(x_i)$ is assigned an interval $[a_i , b_i)$ where $b_i - a_i = p(x_i)$.
3. First symbol is assigned the $[0.0, p(x_1))$ interval.
4. Every next symbol is assigned the $[p(x_1), p(x_1) + p(x_2))$ interval.
5. The last symbol has the interval $[1.0 - p(x_n), 1.0)$

The algorithm for the arithmetic coding technique is:

LowValue = 0.0;

```
HighValue = 1.0;
Do {
        inputString s;
        range = HighValue – LowValue;
        HighValue = LowValue + range * highRangeArray[s];
        LowValue = LowValue + range * lowRangeArray[s];
}
While(symbol != '$')
Print any number in [LowValue, HighValue);
```

Variable length coding methods assign to each video sequence symbol a variable length code, based on the probability of its appearance. Symbols appearing frequently are represented with short variable length codes while less common symbols are represented with long variable length codes. Over a large number of encoded symbols, this replacement of video sequence symbols by variable length codes lead to efficient compression of the data (Held, 1991)[12].

2.7 Video Coding Standards Evolution

Until today, all the commonly known video compression methods/standards have been developed and approved either by ITU-T or ISO. The majority of the standards have followed a discrete development phase by either the ITU-T or the MPEG, while some joint standards have been proposed by both bodies. Figure 11 provides a timeline of the video coding evolution by each standardization body.

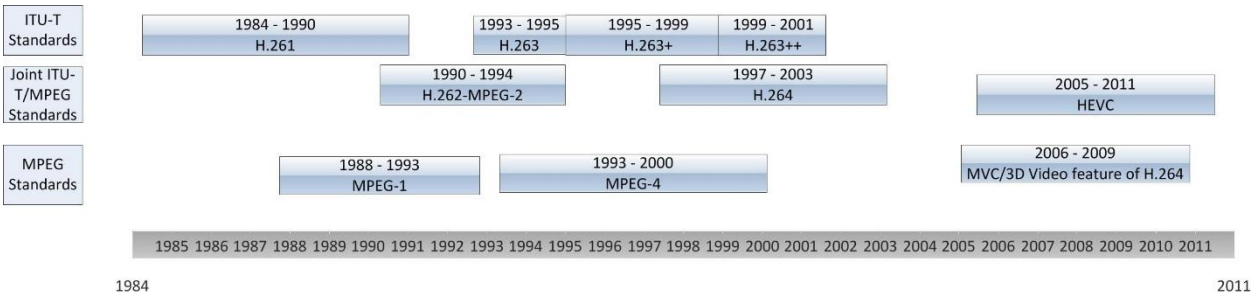

Figure 11 ***Timeline of video coding standards evolution***

On the next paragraph, we provide a brief presentation of the standard H.264 developed by ITU-T and MPEG respectively, focusing on the new features of the standard developed and introduced in the coding community, in 1993. The reason we focus solely on this standard is because it the video coding standard it was used during all the video differentiation process experiments.

- **H.264**

The encoder we used for the experiments was H.264. The intent of the H.264/AVC(Advanced Video Coding) standard was to create a standard capable of providing good video quality at lower bit rates than previous standards (i.e H.263), with an economic design in terms of both complexity and implementation. Another goal of the standard was to be able to easily apply it to a wide variety of applications on a wide variety of networks and systems, including low and high bit rates, low and high resolution video, broadcast, DVD storage, RTP-IP packet networks, and ITU-T multimedia telephony systems.

Due to its wide application on networks and systems H.264 had to be able to provide equal and stable solutions for many of these cases. So it formed a solution based on different profiles, which technically are different configuration files for the encoder. Each profile sets different parameters for the encoder, aiming to provide a suitable solution. H.264 defines a set of three profiles, each supporting a particular set of coding functions and each specifying what is required of an encoder or decoder that complies with the Profile.

3. DIFFERENTION OF SPATIOTEMPORAL DYNAMICS

The impact of either spatial or temporal adaptation on a video signal may have a different impact on the perceptual quality of the video sequence depending on the spatiotemporal dynamics of the content. It is well established from the fundamental principles of video coding theory that action clips with high dynamic content are perceived as degraded in comparison to sequences with slow-moving clips, considering identical encodings.

In order to differentiate a video clip according to the spatial and temporal complexity of its content, a spatiotemporal plot [15] is considered as depicted on Figure 1. According to this approach, each video clip can be classified to four categories depending on its spatiotemporal content dynamics.

For the classification of each test signal, two discrete metrics (one for the spatial and one for the temporal domain) are proposed in this report, which are based on the data analysis of the motion vectors of each video signal frame. The proposed metrics can be applied on every video signal which has been encoded utilizing motion vectors in the motion estimation process. However, for the experimental needs and the validation of the proposed method, this report presents experimental results of video signals that have been encoded by H.264/MPEG-4 AVC [6]. Thus, for the purposes of these experiments, standardized H.264/AVC bit streams of each video signal were used in order to extract the motion vector information.

3.1 Proposed Metrics

In the H.264/AVC video coding standard each frame is encoded using one out of the three different prediction levels: Intra-frame (I), Inter-frame Predictive (P) and Inter-frame Bidirectional predictive (B):

I frames do not contain any prediction information from any other coded frame.

P frames are coded based on prediction from previously encoded I or P frames.

B frames are coded based on prediction from previously, or future encoded I or P frames.

Each frame consists of 16x16 sized regions called macroblocks, which are further subdivided into transform blocks and may be further subdivided into prediction blocks of various sizes.

During the prediction process, the encoding algorithm uses motion estimation to search for an area (macroblock) in the reference frame(s) (past, future or both) in order to find a corresponding matching region. The result of this process is translated into a motion vector for the region under-encoding, which denotes the position where the region will move. Thus, during motion estimation the best matches between reference and current frames are detected and this match is further improved by motion compensation, which calculates the residuals of the motion estimated frame and the actual frame. So, adding this motion compensated residual information on the motion estimated frame, an accurate and efficient prediction of the current frame can be performed, using regions of past or future frames. This data is formatted and stored in the encoded H.264/AVC bit stream file on the corresponding macroblocks.

In order to extract the motion vector information for each frame from the encoded video bit stream file, a motion vector extraction module was implemented and added to the standard H.264/AVC decoder. This additional function stores the motion vector information in the following XML-formatted manner to a file, for later use:

```
<Frame id="integer">
      <Macroblock id="integer">
            <MotionVector x="int"y="y"/>
      </Macroblock>
      .
      .
      .
```

```
    <Macroblock id="integer">
        <MotionVector x="int"y="y"/>
    </Macroblock>
</Frame>
```

As can be observed, the information stored in this file is organized by frame on the first level and by macroblock on the second level. In each macroblock section resides its motion information in the form of the motion vector's x and y coordinates.

Upon parsing the aforementioned file, corresponding to a video signal under-test, an array of frame objects is generated. This array contains all the motion information needed to measure and quantify the motion dynamics of the video signal according to the proposed content residual and motion compensation metrics presented in the next sub-sections.

3.1.1 Content Residual Metric

For every motion vector v_i of a video frame, its norm is calculated by the equation (1) using its *x* and *y* coordinates:

$$|\vec{v_i}| = \sqrt{x^2 + y^2} \qquad (1)$$

Afterwards for each frame, the total sum of all its motion vector norms S_f is calculated:

$$S_f = \sum_{i=1}^{n} |\vec{v_i}| \qquad (2)$$

where *n* is the total number of motion vectors contained in the corresponding frame.

Subsequently we calculate the average of all S_f for the whole video signal duration in equation (3):

$$AvgLengths_v = \frac{\sum_{i=1}^{k} S_{f_k}}{k} \qquad (3)$$

here *k* is the number of the total frames of the video signal. Thus, the result of this equation will be the average motion vector norm of the video signal, which is proposed as the content residual metric for the quantification of the video's total pixel and artifact shift from frame to frame.

Below on figure 12 we present a visualized example of a video with a high valued content residual index. Notice the high valued motion vectors of the fast moving ball on the right picture.

Figure 12 ***Example of a video frame with a high valued content residual index.***

3.1.2 Motion Compensation Metric

Since a high content diversity among the depicted video signal's objects will have an impact on the number of motion vectors per frame, then the proposed metric for the quantification of the spatial aspect of the video content dynamics is proposed the average number of motion vectors per frame, which is calculated in equation (4):

$$AvgTotals_v = \frac{\sum_{i=1}^{k} t_i}{k} \qquad (4)$$

where t_i is the total number of motion vectors per frame, and *k* the total number of frames of the video signal under test. The average motion vector total per frame creates the motion compensation metric. This is derived by the fact that an index of the video's averaged total motion activity can give us the video's motion compensation.

Below on figure 13 we present a visualized example of a video with a high valued motion compensation index. Notice the differences from the previous frame on the image on the right.

Figure 13 ***Example of a highly motion compensated video frame.***

3.2 Video Differentiation Process

The proposed differentiation method is based on the values of the two proposed metrics for the measurement of the motion compensation and content residual aspect of the video's content dynamics.

Considering the values of the two metrics as coordinates on the proposed content residual and motion compensation grid of figure 1, each video sequence (subject to short duration and homogeneous content) is depicted as a spot in the content residual and motion compensation plane, where the vertical axis refers to the motion compensation aspect of the content dynamics through the proposed metric (4) (i.e. the mean motion vector number) and the horizontal axis refers to the content residual aspect of the content dynamics through the proposed metric (3) (i.e. the mean motion vector norm).

The advantages of the proposed two dimensional representation of the spatiotemporal video content dynamics are the unique graphical characterization of each test signal and the relative comparison of content residual and motion compensation dynamics, while the main drawback is the difficulty in the characterization of highly heterogeneous test signals.

4. EXPERIMENTAL RESULTS

For the experimental needs of this report towards the validation of the proposed method, 16 reference video clips were used [5], which are representative of various spatiotemporal video content dynamics. The test signals have spatial resolution 416x240, 832x480, 1280x720 and 1920x1080. The videos and their corresponding resolution are listed in Table 2, along with a representative video frame on Figure 14. For the validation needs of this report all videos were encoded from their original YUV format to ISO AVC High Profile with quantization parameter equal to 12. Across the encoding process the reference software was used. During the decoding process the decoder reference software was modified in order to export an xml file with the motion vector information. No other changes were made that could alter the decoder's behavior or results.

Figure 14 Snapshots of the Experimental Validation video signals. Starting from top left towards right BasketballDrill, BasketballDrive, BasketballPass, BlowingBubbles, BQSquare, FourPeople. Keiba, KristenAndSara, Mobisode, Parkrun, PartyScene, RaceHorses, Stockholm, Kimono, Tennis, Parkscene.

Video Name	Resolution
BQSquare	416x240
BlowingBubbles	416x240
Mobisode	416x240
BasketballPass	416x240
Horses	832x480
Keiba	832x480
BasketBallDrill	832x480
PartyScene	832x480
FourPeople	1280x720
KristenAndSara	1280x720
Parkrun	1280x720
Stockholm	1280x720
BasketballDrive	1920x1080
Kimono	1920x1080
Parkscene	1920x1080
Tennis	1920x1080

Table 2: The test signals of the Experimental Validation grouped by video resolution.

The test signals were grouped into 4 experimental sets that contain test signals of the same spatial resolution in order to be possible the relative comparison of their content dynamics based on the proposed grid and metrics. Thus, based on this grouping, the following sets have been created.

The first differentiation experiment is among 4 different 416x240 resolution video signals (Basketball Pass, BQSquare, BlowingBubbles and Mobisode), its results are presented in Figure 15.

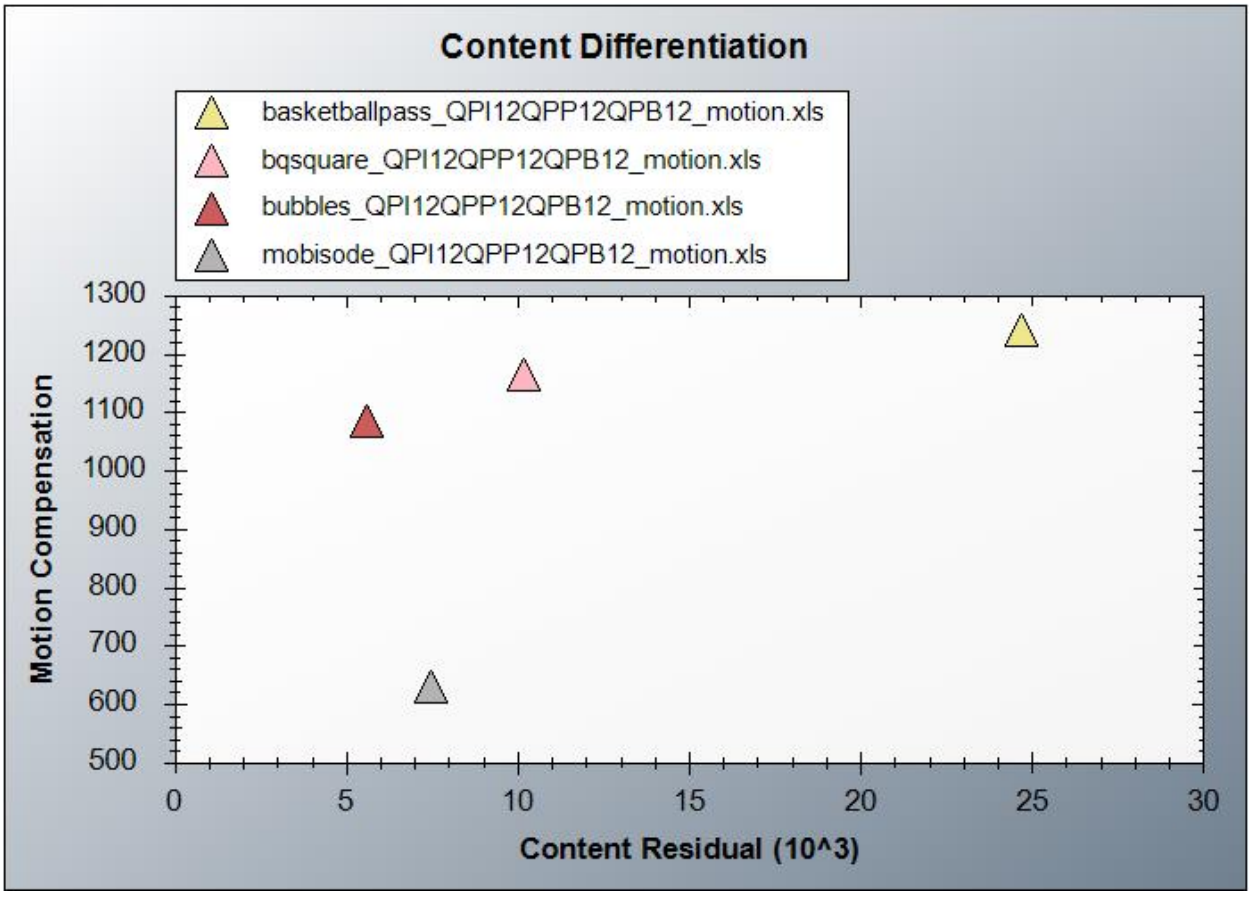

Figure 15 : Video differentiation chart for videos with resolution of 416x240.

Based on the visual results of Figure 15, it is evident that the video sequence basketballPass is the most active of the group in both the motion compensation and the content residual plane. This result is utterly justified by the video's content. The basketballPass sequence contains rich spatial and temporal motion activity as it comprises of several athletes moving constantly during a basketball practice. The overall motion activity is not only continuous during the duration of the video, but also very rich as it contains abrupt and fast movement from frame to frame, depicting the athletes' quick movements. Additionally, the camera alters its focus during the video, thus creating background changes which contribute to the overall mean motion vector number. All this quick and fast motion activity of the video elements are translated into a high valued mean motion vector length (content residual), but also into a large motion vector number (high valued motion compensation index), thus placing the baketballPass video signal

on the top right corner relatively to the other 3 videos, which deciphers in both rich spatial and temporal motion activity.

Heading southward on the results diagram we meet the Mobisode video sequence which is placed as the video signal with the third largest motion vector length per frame, thus the second largest content residual index, but at the same time with the smallest motion compensation index. The Mobisode video sequence contains mainly homogeneous content, with a few scene changes and camera zooms, and without an overall high motion activity. So, its relatively mid valued content residual index is justified by the scene changes, in which a fair number of motion vectors shift or change completely in order to compensate for the successive image differences, but also from the camera zooms where large pixel areas are shifted. Due to the video signal's color, luma homogeneity and low motion dynamics, the encoded signal does not have a large number of motion vectors, as the color shifts in adjacent frames are minimal, thus the smallest valued motion compensation index.

The remaining 2 videos BQSquare and BlowingBubbles appear to be very similar spatio-temporally. Based on the results of the video differentiation method, they both have significantly richer content residual than the Mobisode sequence. Initially, the BQSquare sequence consists of a single shot recording on a square with several people doing various activities. The camera zooms out during the recording of the sequence which contributes to the growing number of motion vectors (motion compensation index) from frame to frame, as new objects enter and exit our perspective and contribute to the overall residual increase of the video signal. In the case of the BlowingBubbles sequence, the video sequence consists of 2 girls in a party blowing bubbles. The bubble artifacts create the most motion activity in both axes, but their continuous shift and movement through frames generates a lot of motion compensation, thus the high valued motion vector number. However, their continuous slow movement throughout the whole video does not add up significantly to the mean motion vector length, thus the BlowingBubbles sequence has the lowest value of the group.

The second evaluation set consists of 4 videos of 832x416 image resolution the BasketballDrill, Horses, Keiba, and PartyScene. Figure 16 presents the visualized results for the video differentiation process applied to the aforementioned group of videos.

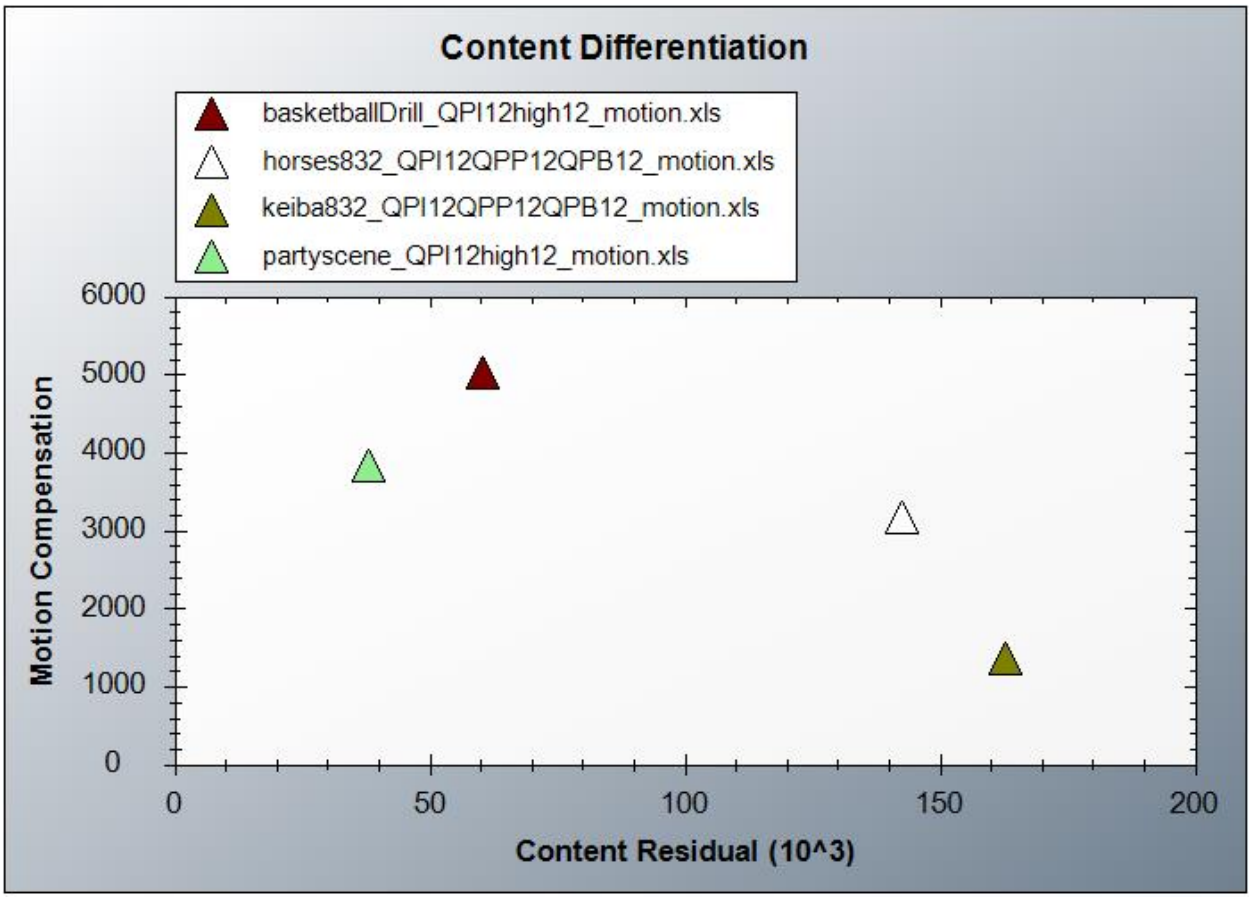

Figure 16 : Video differentiation chart for videos with resolution of 832x480.

The most distinct video differentiation position in this graph is the score of the Keiba video sequence. The Keiba video signal is recorded during a horse race, where the camera is locked upon a single horse and follows it during the whole video. The motion dynamics in the video sequence are very high because of the fast background change, as the camera is locked-in, an abrupt and constant background change is observed. The horse's shift from frame to frame translates into high valued motion vectors, as the horse's pixels shift fast. The high valued mean motion vector length translates into a high valued content residual index. The main motion compensation generation created by the background change as the camera follows the horses,

but the background has a repetitive pattern, which do not add up to the total motion vector number, and thus the overall motion compensation index.

Moving on the second most motion compensated video sequence under test, the RaceHorses video signal is examined. The video RaceHorses appears to be the middle case among all the videos in the group under test, the video signal is a close up to a group of walking horses along with their riders. The rich color moving and the rapid horse movement generate rich spatiotemporal activity, along with the camera close-up. The camera zoom creates multiple divergent areas. The horses' a similarly high valued mean motion vector length, as the Keiba video sequence. However, the video's homogeneous background and colors, create a smaller number of motion vectors than the other 2 videos as they smoothly shift from frame to frame, thus not creating large pixel area movements, and a relatively low valued content residual index.

For the remaining 2 video signals PartyScene and BasketballDrill, it is observed that they have the highest motion compensation levels and the lowest content residual index. The PartyScene video signal consists of a few children playing and moving around in a room near a plant. The children's continuous slow paced movement results in low valued motion vector lengths and keeps the content residual index, at a low level. The camera zooms out during the whole video, thus the background changes (significant pixel area shifts) and this results in a high motion vector number and a high valued motion compensation index. In the last video, BasketbalLDrill, 4 basketball players perform a basketball exercise, by hitting one after the other against the basketball on the board. Similarly to the PartyScene video sequence BasketballDrill contains short macroblock shifts, due to the ball and player movement, which occur during whole video sequence. However, alike the PartyScene sequence the fast paced and continuous movement generates a large number of motion vectors. Which is why BasketballDrill has the highest valued motion compensation index.

The third evaluation set consists of 4 high definition videos of 1080x720 image resolution the Stockholm, Parkrun, KristenAndSara, and FourPeople. Figure 17 presents the visualized results for the video differentiation process applied to the aforementioned group of videos.

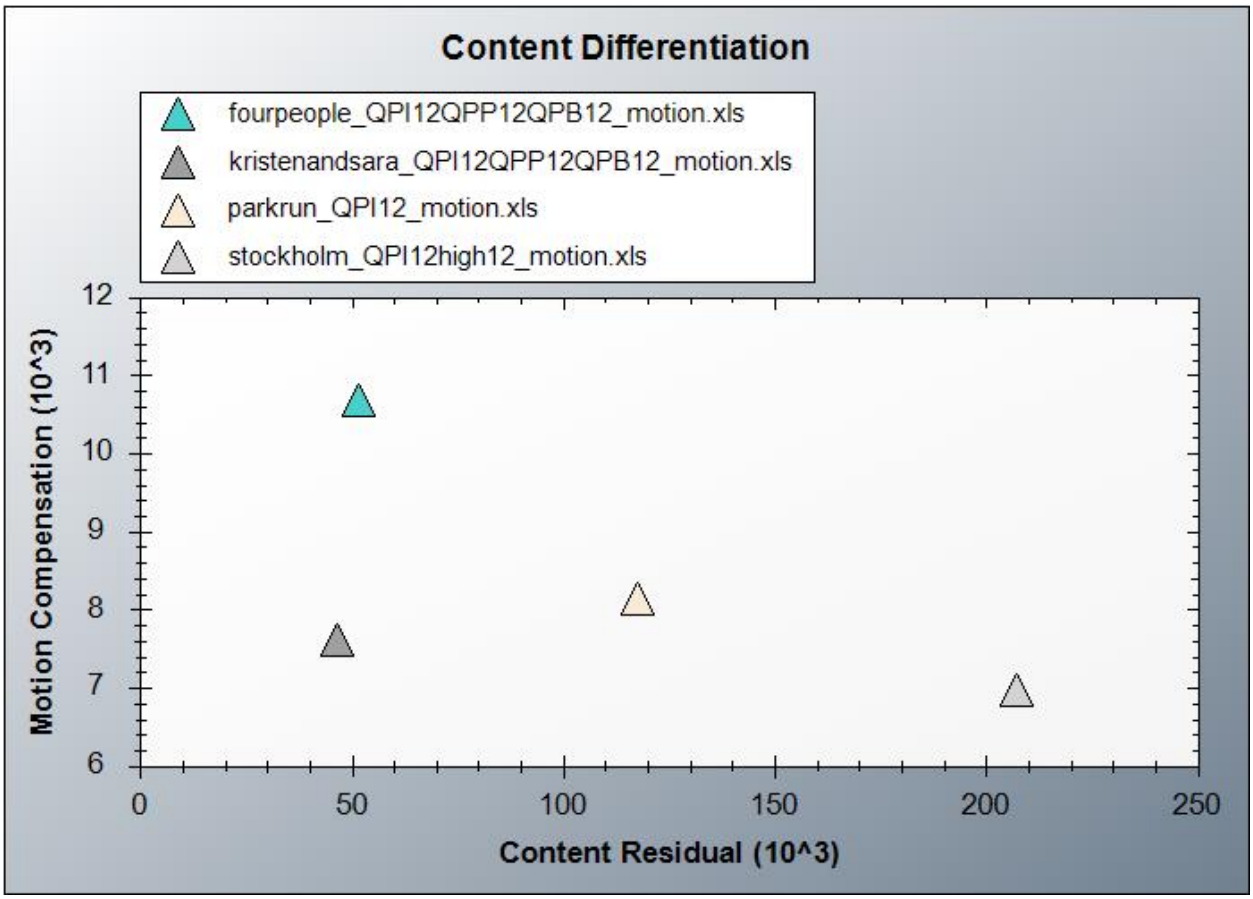

Figure 17 : Video differentiation chart for videos with resolution between 1280x720.

The visualized motion differentiation results for the third video set do not display significant motion compensation and content residual index differences among the current video sequences under-test, apart from the case of the Stockholm video signal which scores considerably higher on the content residual domain. The Stockholm video presents a panoramic overview of the city of Stockholm. The camera shifts in a fast pace on the horizontal axis and captures the ongoing activity of the city, mostly car traffic. The mean motion vector length (content residual domain) is relatively the highest in the group, due to the active motion activity. The shifting camera does not introduce new large moving objects and does not cause significant background change on the capturing frame, this generates a low motion vector number, thus holding the average motion vector number and the motion compensation index to its lowest value among the test set.

Heading westwards on the diagram, the next video, Parkrun, features a man running alongside a river in a park. The mean motion vector length is mainly generated by the movement of the man, but also from the fact that the river and the scene background record wide macroblock movements, due to the rich and highly detailed scenery. However, the detailed nature background, mainly because of the video's high definition provoke a considerable number of motion vectors, which adds up from frame to frame and finalizes in a relatively high average motion vector number per frame. Additionally, the continuously moving camera contributes a part to the increased number of motion vectors.

Finally, the remaining 2 videos FourPeople and KristenAndSara, both feature people talking, 4 and 2 respectively. The 2 aforementioned videos, based on the differentiation process results, have the lowest content residual index. The KristenAndSara video content does not record overall a lot of active motion activity, as it only consists of 2 women have a chat, with mild body movement. Similarly, on the video FourPeople there are 4 people sitting on a conference table exchanging documents and having a discussion. In a similar manner, there is no fast, or abrupt motion recorded, which leads to the low motion vector length. Nonetheless, the average mean motion vector number is relatively high compared to the rest of the videos of the group under test. The reason why this occurs is the camera is recording at a very close distance, so every movement is intensified, and this generates a high valued mean motion vector number. The persons' body movement are continuous throughout the video. The macroblock movement is not significantly active, as the camera is still (no background changes), thus the final low content residual index result for both videos.

The fourth and final evaluation set consists of 4 high definition videos of 1920x1080 image resolution the ParkScene, BasketballDrive, Kimono, and Tennis. Figure 18 presents the video differentiation process visualized results applied to the aforementioned group of video signals.

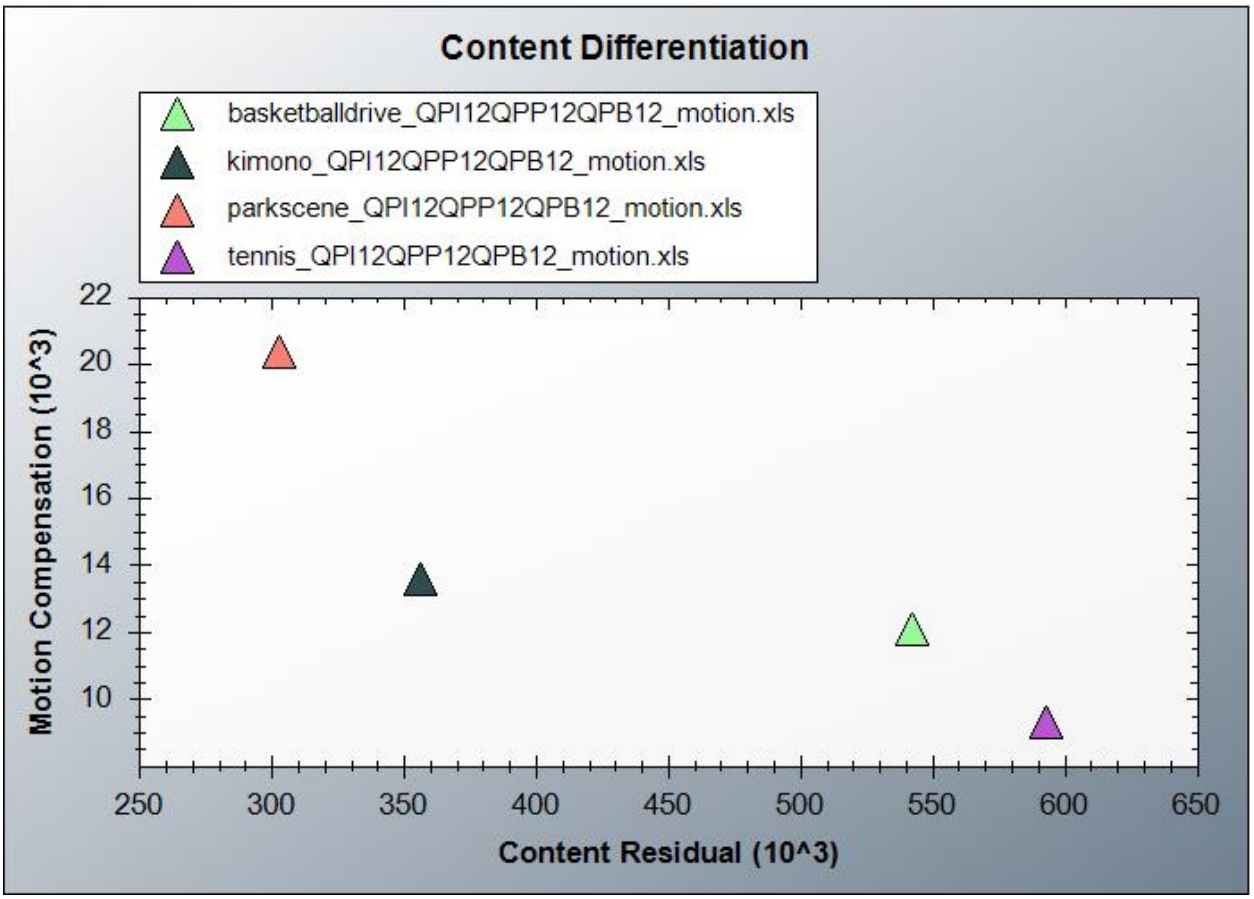

Figure 18 : Video differentiation chart for videos with resolution between 1920x1080.

In the last video set under test, it can be observed that the Tennis video sequence is by far the most motion active (content residual index is high) video sequence of the video group. The Tennis sequence features several tennis players preparing for a tennis match, and then records their activity during a play of the match. During the video the players continuously move across the terrain (generation of motion vectors) in a fast-paced manner, which translates into the high-valued content residual index of the video. However, the camera is recording from a distance, which means that the players' motion activity do not record a large number of motion vectors, thus the relatively lowest motion compensation index. The next most motion active video sequence is the BasketballDrive video signal, in this sequence we watch a basketball practice with similar non-stop movement during the duration of the video a static view prospect of the practice. The continuous fast movement creates high valued motion vector lengths, thus the high content residual index.

In the following sequence Kimono, we watch a lady pass through rich forest background and on the next scene she is approaching a pagoda building. The background in the first scene is very rich in detail and colors and changes significantly as the woman passes through. The background richness and the camera's zoom generate large number of motion vectors, as the tree-leaf pixels generate a lot of MB and pixel shifts. The relatively low content residual index is a result of the low motion dynamics of the video, as no significantly active movement is recorded. In the last video signal, the ParkScene namely, the video sequence features various people as they roam and meet each other in the park. The camera position is moving during the whole filming process generating a lot of motion vectors, due to background scenery change. The motion vectors are created by the bikers. It is justified that the ParkScene sequence has the highest valued motion compensation index as the background is rich and the general movement is continuous. But it has the lowest valued mean motion vector length as the bikers' and camera movement is slow paced and generates a low valued content residual index. The following table, Table 3, presents in detail the video differentiation process experimental results for all video signals.

Video Name	**Motion Compensation Index**	**Content Residual Index**
BQSquare	1164.795341	10212.07848
BlowingBubbles	1083.976048	5613.23795
Mobisode	629.3488372	46450.92796
BasketballPass	1238.401198	24714.48644
Horses	3177.336667	142580.3155
Keiba	1359.242525	163087.9215
BasketBallDrill	5017.786427	60467.14544
PartyScene	3825.742515	38022.49353
FourPeople	10681.075	51665.28634
KristenAndSara	7614.205	46450.92796
Parkrun	8138.491722	117497.9814
Stockholm	6963.178808	207171.527
BasketballDrive	12087.83832	542212.2435
Kimono	13589.025	356193.0327

Parkscene	20352.41667	303002.7519
Tennis	9274.041667	593283.856

Table 3: This table presents in detail the video differentiation process results Motion Compensation Index and Content Residual Index for each video sequence.

5. EVALUATION AND EXPERIMENTAL PROCESS

5.1 Overview of the Video Differentiation Process

The video differentiation procedure consists of several individual video processing steps. The video differentiation process requires a video signal encoded in H.264 format, in order to be able to extract from it the motion vectors of B and P frames, which are required to calculate the overall motion information indexes, as discussed in previous chapters.

The whole procedure was performed by the under-development MN Lab Video Tools program. The MN Lab Video Tools is technically a program suite, which offers several video processing functions, but also combines and enhances other video processing programs to create an enhanced and uniform video processing platform, where you can perform easily various video analysis and processing tasks, without constantly switching environments. The following diagram visualizes in a high level the whole video differentiation process and its partial components. Each of the components is an individual and independent MNLab Video Tools function, altogether create the enhanced graphical video processing environment, in which the video differentiation process was performed.

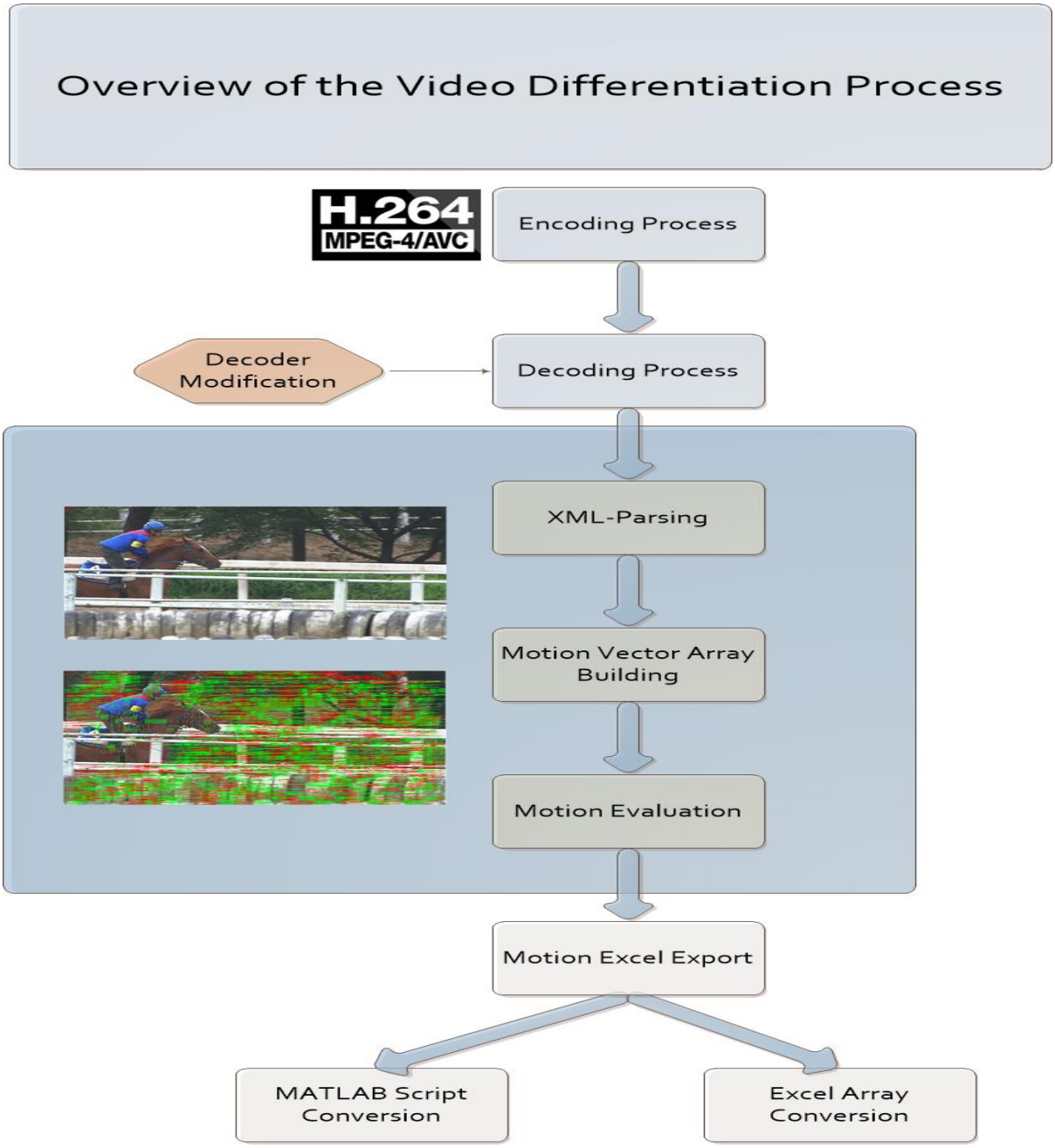

Figure 19 : Overview of the modules which comprise the Video Differentiation process.

For each reference video signal that was chosen for the experimental process the first stage was the encoding of the video sequence. Each video sequence was in YUV format, uncompressed and at various screen resolutions, as already discussed. The encoder chosen was the widely used in multimedia services applications, the AVC/H.264. The parameters used for each video sequence encoding were the same and remained

unaltered during the entire process. In order to achieve the best quality assurance of the encoding's outcome the quantization parameter (QP) value was selected to be 12. In more detail, the QP value regulated how much spatial detail is retained. When the QP is very small, nearly zero information is lost.

Regarding the fact that the main scope of this process was to differentiate a group of video signals based upon their spatial and temporal information, meaning a set of lower video layer features. It is important to keep intact the video's original and uncompressed visual information as much as possible, but at the same time generate the motion information we need, by applying the H.264 encoding process upon the video. The H.264's efficient motion vector prediction is designed to minimize the visual information fully encoded from frame to frame, meaning that during the video transmission we encode and send only the motion vectors of the precedent and adjacent frames, produced by their difference to I frame, which is fully encoded.

Another important encoding parameter is the H.264 profile. The video profile used in the entire experimental process was the High profile. But H.264 has a wide variety of profiles, each one targeting to a specific class of applications. The most widely used are the Baseline profile, the Main profile and the High profile. The Baseline profile is used for low-cost applications, where additional complexity and need for computational resources needs to be minimized and the quality of the final outcome is not considered a priority. The Main profile is used for standard definition video broadcasts, and used to be used for high definition video signals before the arrival of the High profile. Finally the High profile is used for broadcast and disc storage applications, specifically for high definition video services and offers the best compression/quality trade-off from all the aforementioned profiles. The High profile was the ideal choice for the experiment's standards, as it combines excellent compression efficiency and produced image quality, for a wide variety of video resolutions, and for the video differentiation process's case a range from 416x240 to 1920x1080.

The encoding's process outcome for each video signal was a H.264 standard compliant .264 format file.

The next step of the video differentiation process is the decoding process, the functional core of the proposed method. The whole architecture depends on the decoding procedure of the encoded video signal. In this step, all the necessary information extraction is performed. After we have encoded the original uncompressed video signal and created a .h264 bitstream file, where all the video data is stored, we have to decode it in order to recreate a distorted original. The bitstream file contains the motion information required to recreate all the frames that were not fully encoded.

Our method requires obtaining the motion vector information from the bitstream file and storing it in an independent file, for further processing, studying and analysing. The way to achieve it, was to modify the standard H.264 decoder functionality and enhance it with a XML trace file creation. A brief code snippet of the discussed functionality is presented in the following section:

Xml Motion Vector Code Snippet:

```
xml_write_start_element("Absolute");

xml_write_start_element("X");

xml_write_int(curr_mv[0]);

xml_write_end_element();

xml_write_start_element("Y");

xml_write_int(curr_mv[1]);

xml_write_end_element();
```

The presented code snippet was inserted on the macroblock source code file, where the motion vectors are extracted from the encoded H.264 bitstream file. The motion vector values are then inserted in the decoding video's xml file for the particular macroblock, which is decoded at that point. After the decoding process is completed the XML trace

file is finalized, and is available for further analysing. The XML generated has the motion information ordered by each frame and by each macroblock, as the motion vectors are related to the macroblock structure. The XML file's format is previously described in section 2.

Following the XML motion vector file creation, a XML parser program was implemented in order to parse and store the XML file's motion vector values grouped by macroblock and then by frame.

To improve code structure, motion vector information storage was organized by object modelling. For the needs of this task, the objects Frame, Macroblock, SubMacroblock, and MotionVector were implemented, accordingly. The Frame object has a dynamic array of Macroblocks, populated as we parse the XML file and discover the Macroblocks of each frame. One level down to the hierarchy is the Macroblock which may consist of SubMacroblocks, or include the motion vector arrays. A SubMacroblock object may contain the motion information we need, in that case we match the Submacroblock motion vectors on its parent Macroblock. Each Macroblock and SubMacroblock object has an array of motion vectors. After we store on runtime the XML-parsed motion vector values in the according object arrays, we sum up the results by frame, which summates to a video sequence motion information object collection.

This motion information object collection is then exported into an excel format file. Storing it into a file we can then access it anytime and perform various calculations and estimations with the data. The calculations and the platforms to perform them have no restrictions, and can be further extended in future research. For our experiments we took advantage of this information by implementing 2 solutions for video differentiation, one for MatLab and one for Excel; the latter was picked for the representation of our video differentiation visualized results.

The MNLab Video Tools were developed using the C# programming language under the ASP .NET 4 framework. The MNLab Video Tools also made use of various programs and libraries to achieve its purpose. These are the reference H.264/AVC encoder developed

by the Fraunhofer Institut [9], the corresponding decoder although a modified version of it, and the MatLab tools platform.

5.2 Snapshots of the MNLab Video Tools

This chapter presents snapshots from the functionality of the MNLab Video Tools program suite, which display various video processing scenarios performed during the research experiments.

In our first snapshot we present the first screen/tab of the MNLab Video Tools program suite.

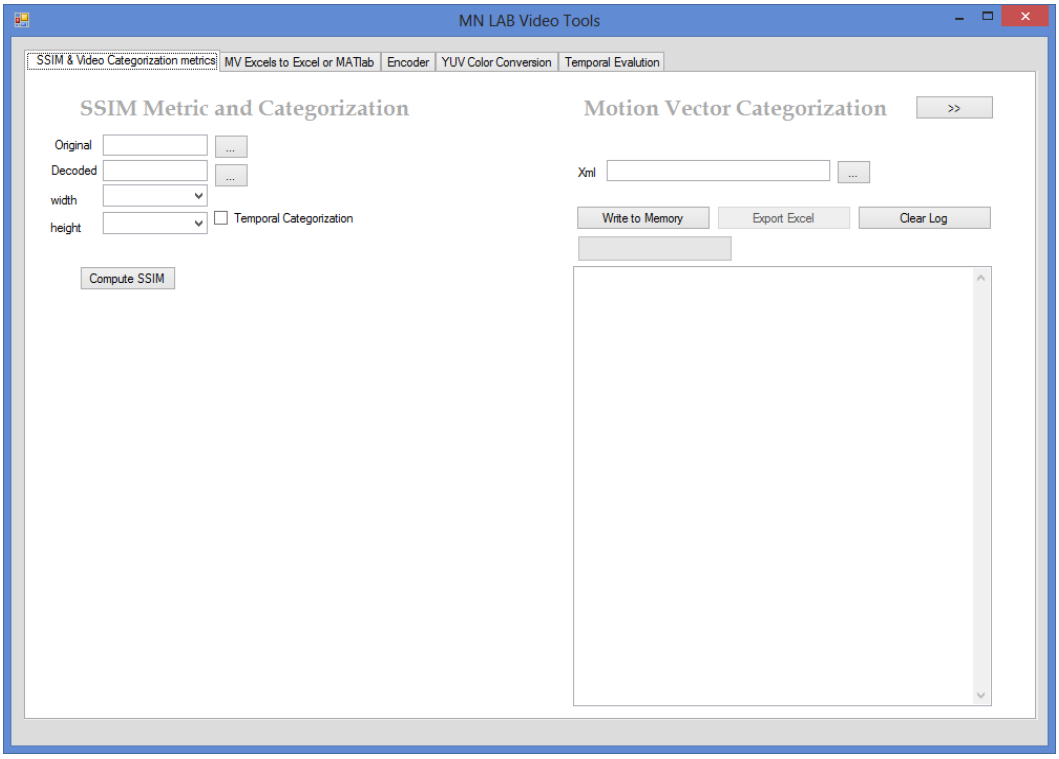

Figure 20 Snapshot of the first screen of the MN Lab Video Tools, with the SSIM quality assessment module and the motion vector categorization module.

As we can see from figure 20 the first tab of the tool contains 2 functionalities, the first one is a video quality assessment tool using the SSIM, and the second is the motion vector export function. In the Xml field we insert the motion vector XML file for a video signal, which was previously generated by our modified decoder.

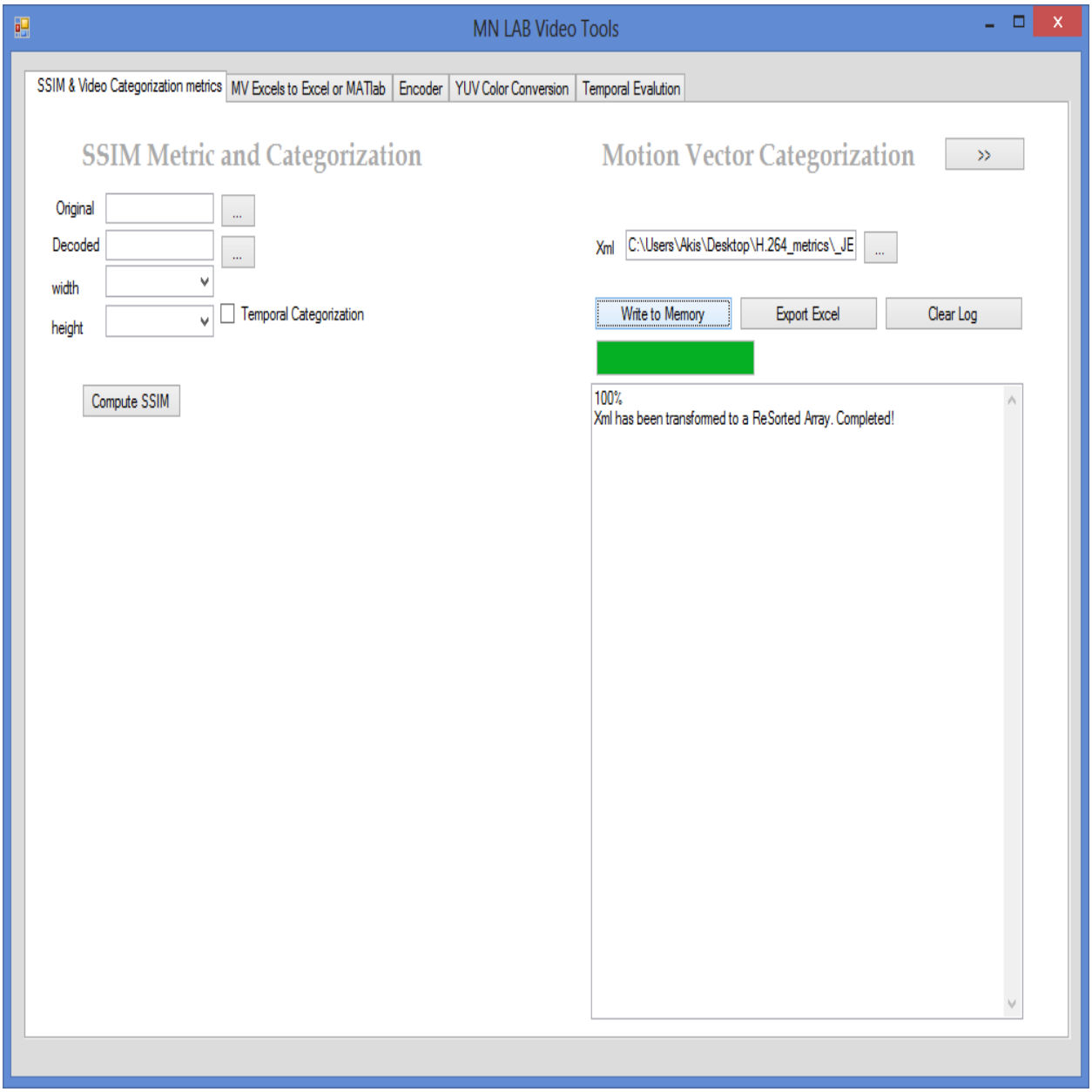

Figure 21 The first screen after a motion vector XML is parsed and stored to memory for later use.

In figure 21 we can see the program after we have executed the Write to Memory command, in which the program reads the XML motion vector file and creates the Frame, Macroblock and Motion Vector objects and inserts them in a dynamic array. This array technically contains all the motion information of the video sequence under evaluation.

In the next figure we execute the command to export it into an Excel file, in order to be able to perform our motion estimation process.

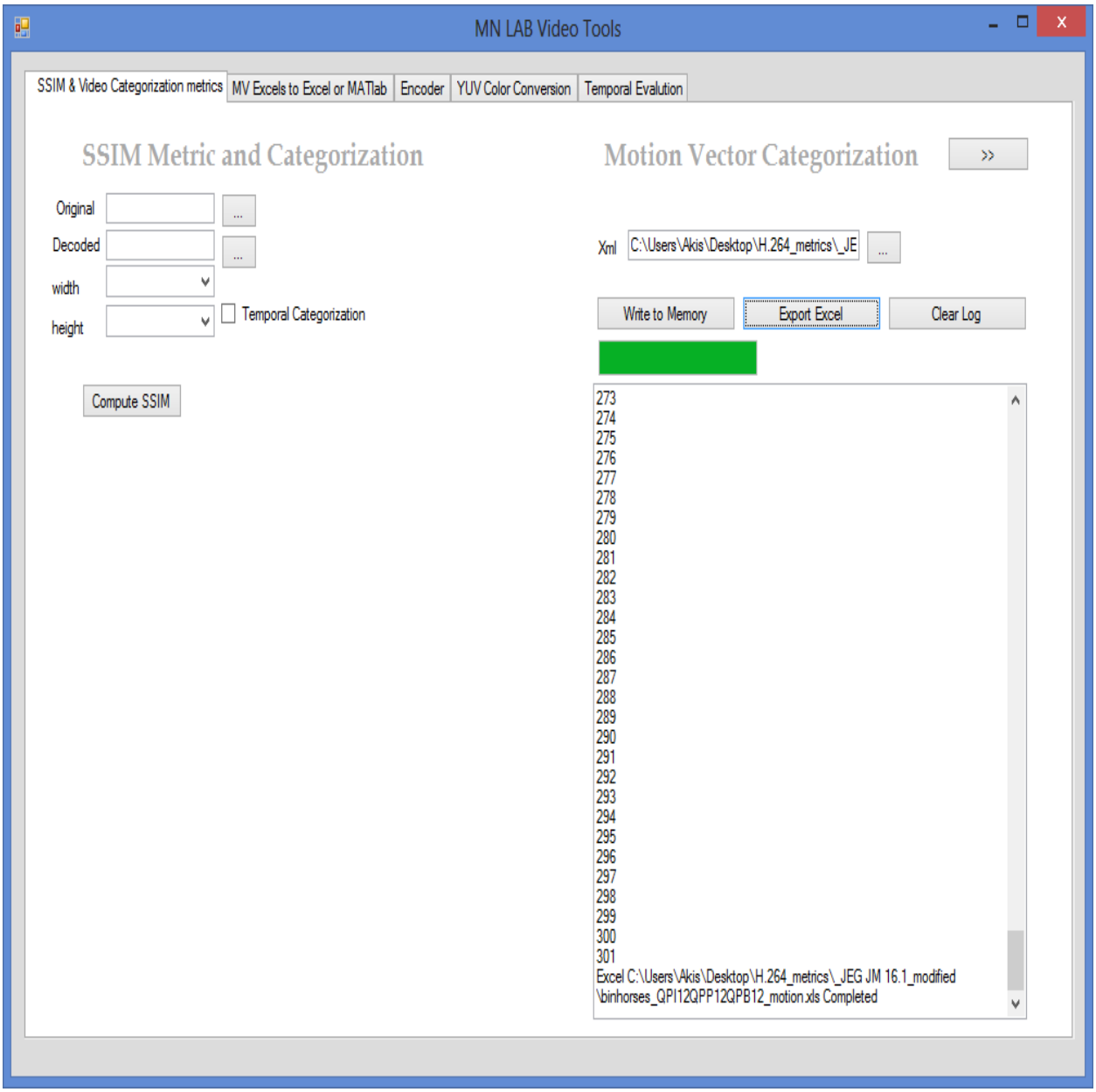

Figure 22 The first screen of the MNLAB Video Tools after the stored motion vector arrays are exported into an Excel file, in order to proceed to the next step of the video differentiation process.

While the command is executed, the program transfers the data from the arrays created in the previous snapshot to an excel file, which has 2 columns and every line is for every frame of the video signal. The first column represents the mean motion vector length of each frame and the second column is the motion vector sum of each frame.

In the next snapshot a typical XML motion vector file for a video sequence is presented.

```
<SubMacroBlock num="1">
    <Type>3</Type>
    <TypeString>P_L0_4x4</TypeString>
    <MotionVector list="0">
        <RefIdx>0</RefIdx>
        <Difference>
            <X>0</X>
            <Y>0</Y>
        </Difference>
        <Absolute>
            <X>-1</X>
            <Y>-4</Y>
        </Absolute>
    </MotionVector>
```

Figure 23 Typical XML motion vector file.

In figure 23 we see a submacroblock structure, along with one of its motion vectors, and the values of the motion vector. The coordinates/vector the program exported was the absolute values.

In the next figure we move onto the next tab of the MNLab Video Tools program suite.

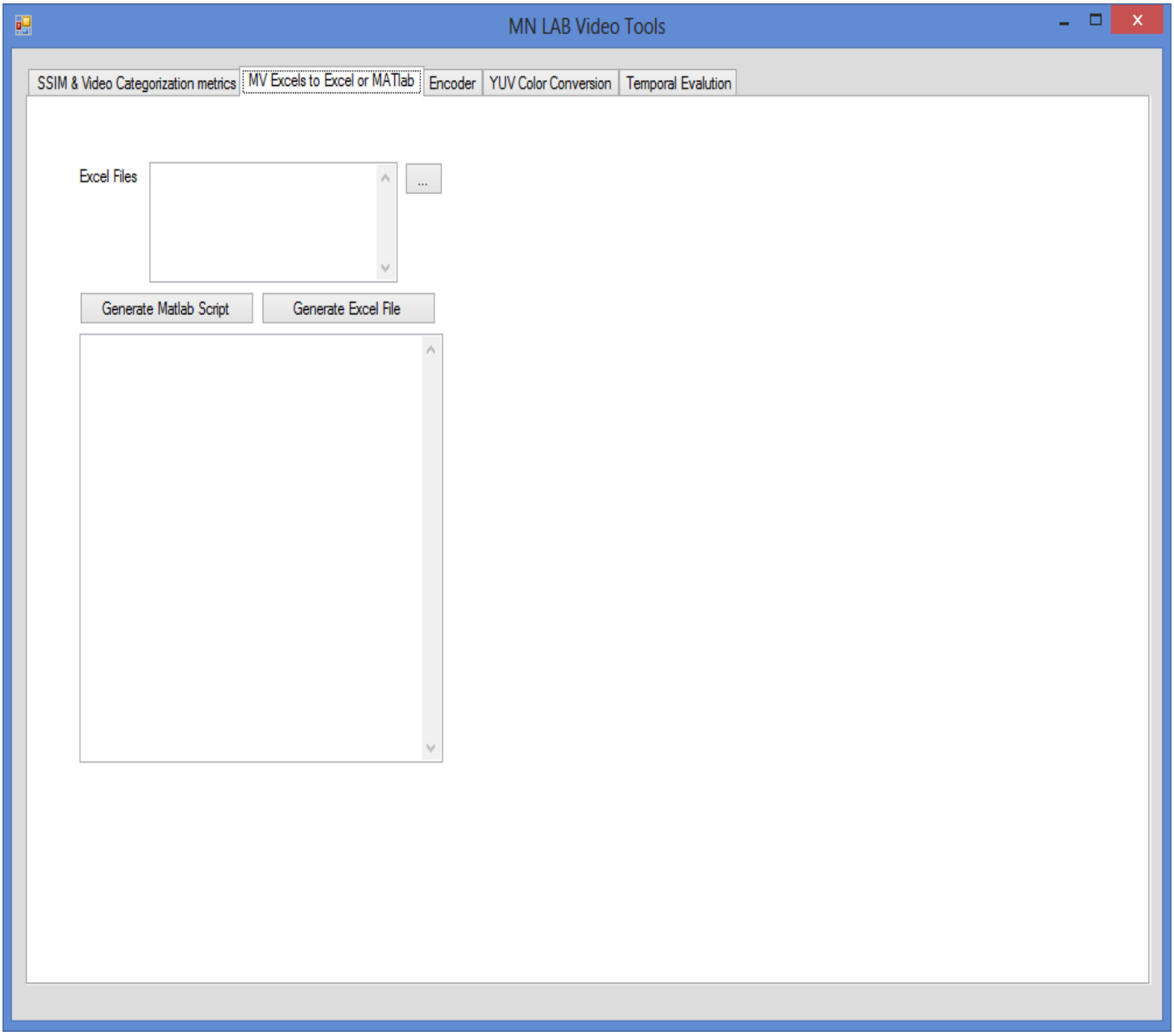

Figure 24 Second Tab of the MNLab Video Tools.

The functionality of the second tab is to parse the previous generated Excel file, which contained all the motion information of a video signal, calculate the video sequence's overall mean motion vector length and mean motion vector number, and finally print them in a script command for the MatLab program, or in Excel formatted cells. This enhanced functionality was implemented in order to facilitate the visualization of a large number of video differentiation results, without losing precious time copying and transferring data from platform to platform.

In the next figures we see examples of this functionality.

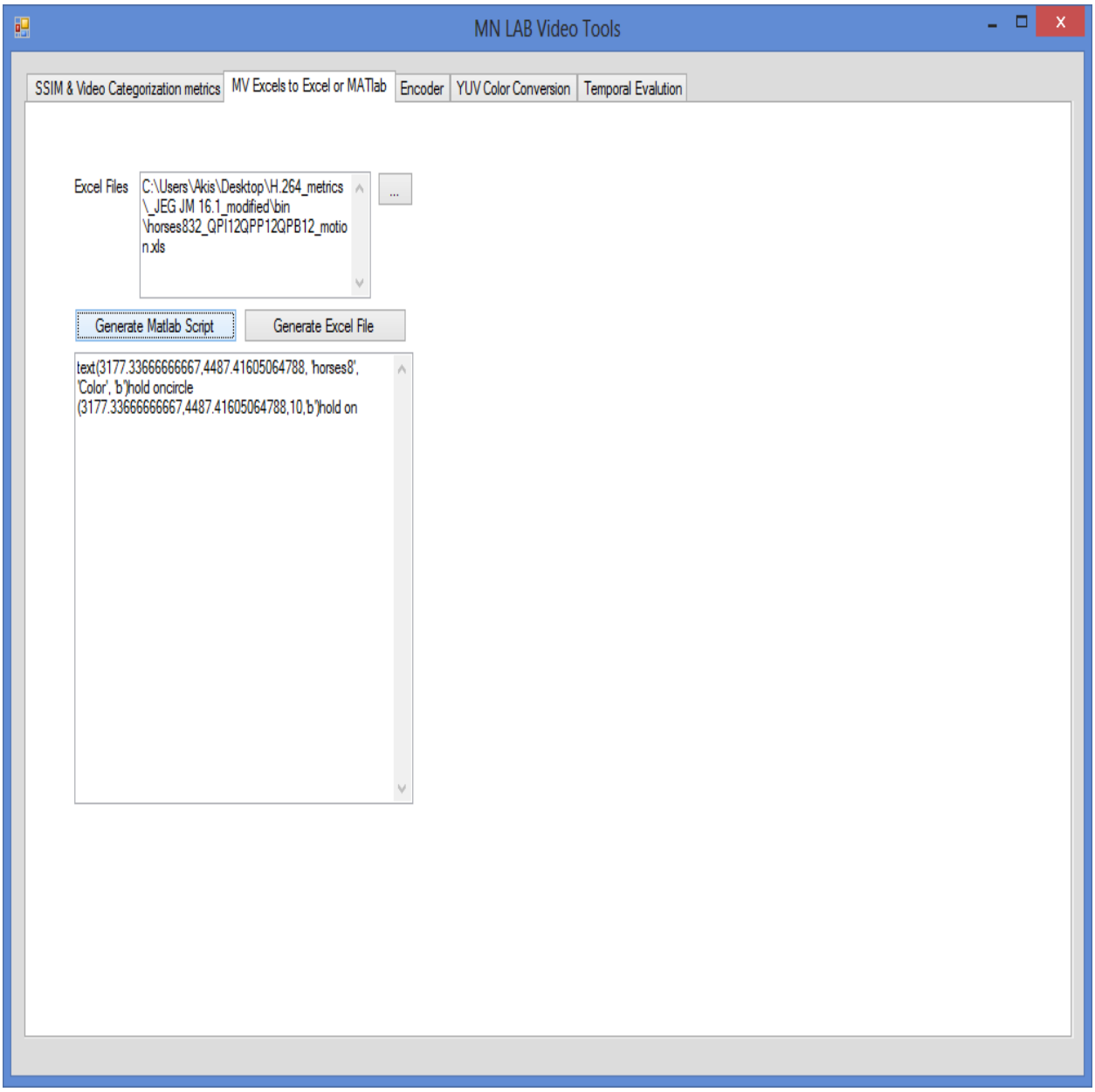

Figure 25 Generation of MatLabScript

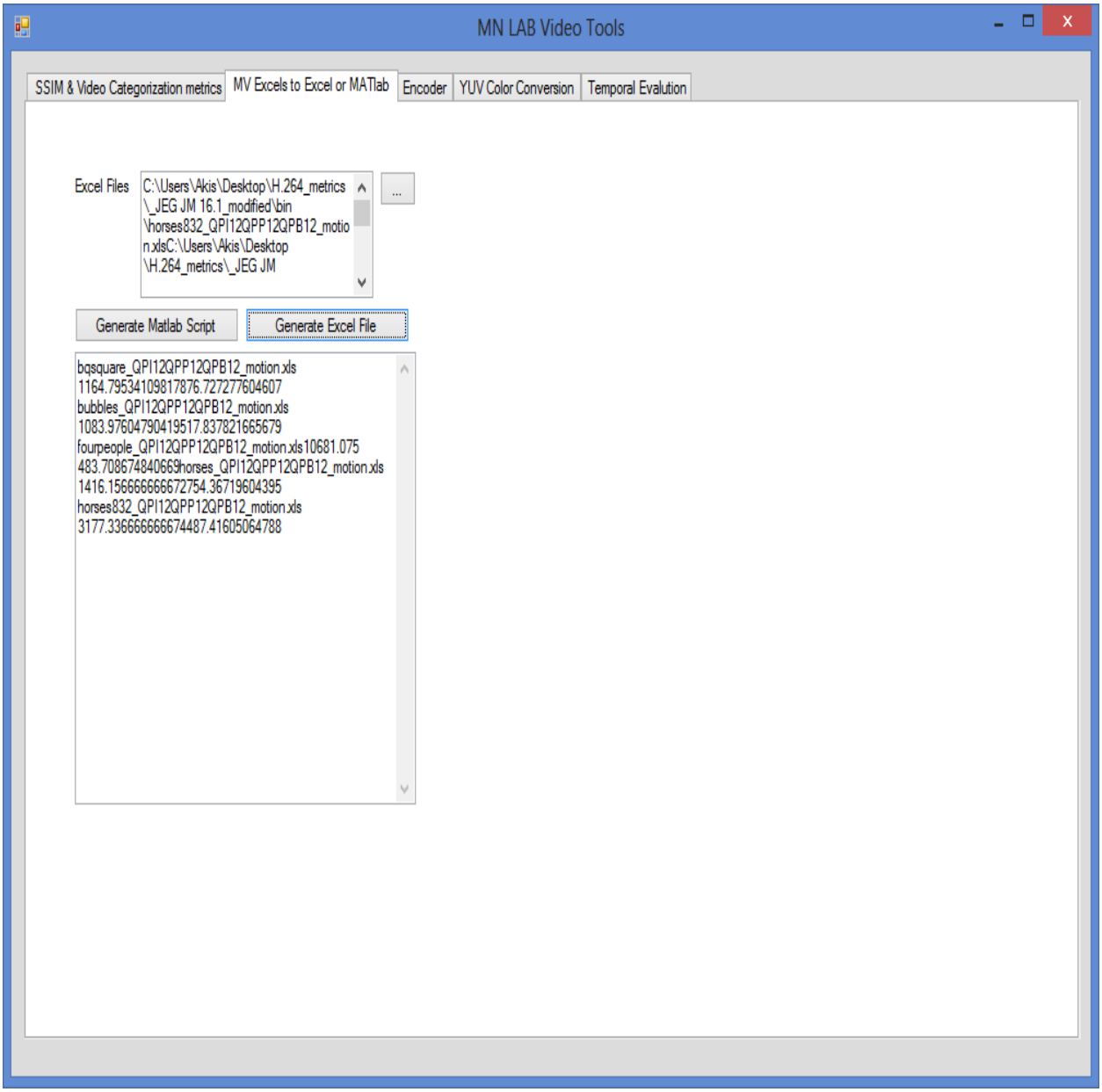

Figure 26 Generation of Excel Cells Formatted Records

After we have seen some examples of the functionalities of the second tab, we move on to the third tab which contains a very vital and basic functionality required for our experiments. The functionality contained is the H.264 video encoding.

In the following figure a general overview of the tab's interface is presented.

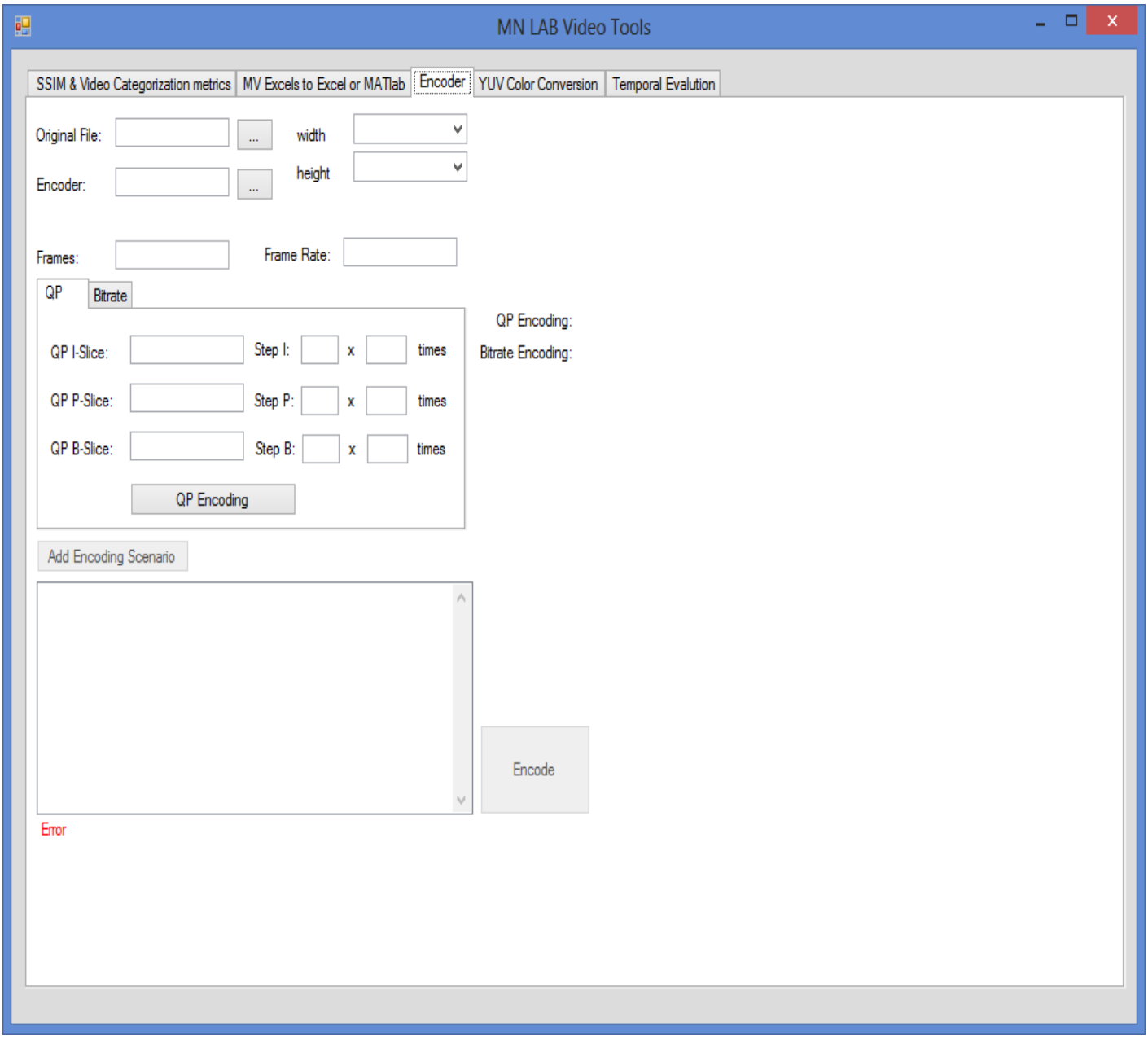

Figure 27 Overview of the H.264 video encoding functionality tab.

In figure 27 we can see an interface which simplifies and facilitates the h.264 video encoding process. For the needs of the entire process, but also for any provisioned future work, this page contains fields for video encoder parameters.

An example functionality is presented in the next figure.

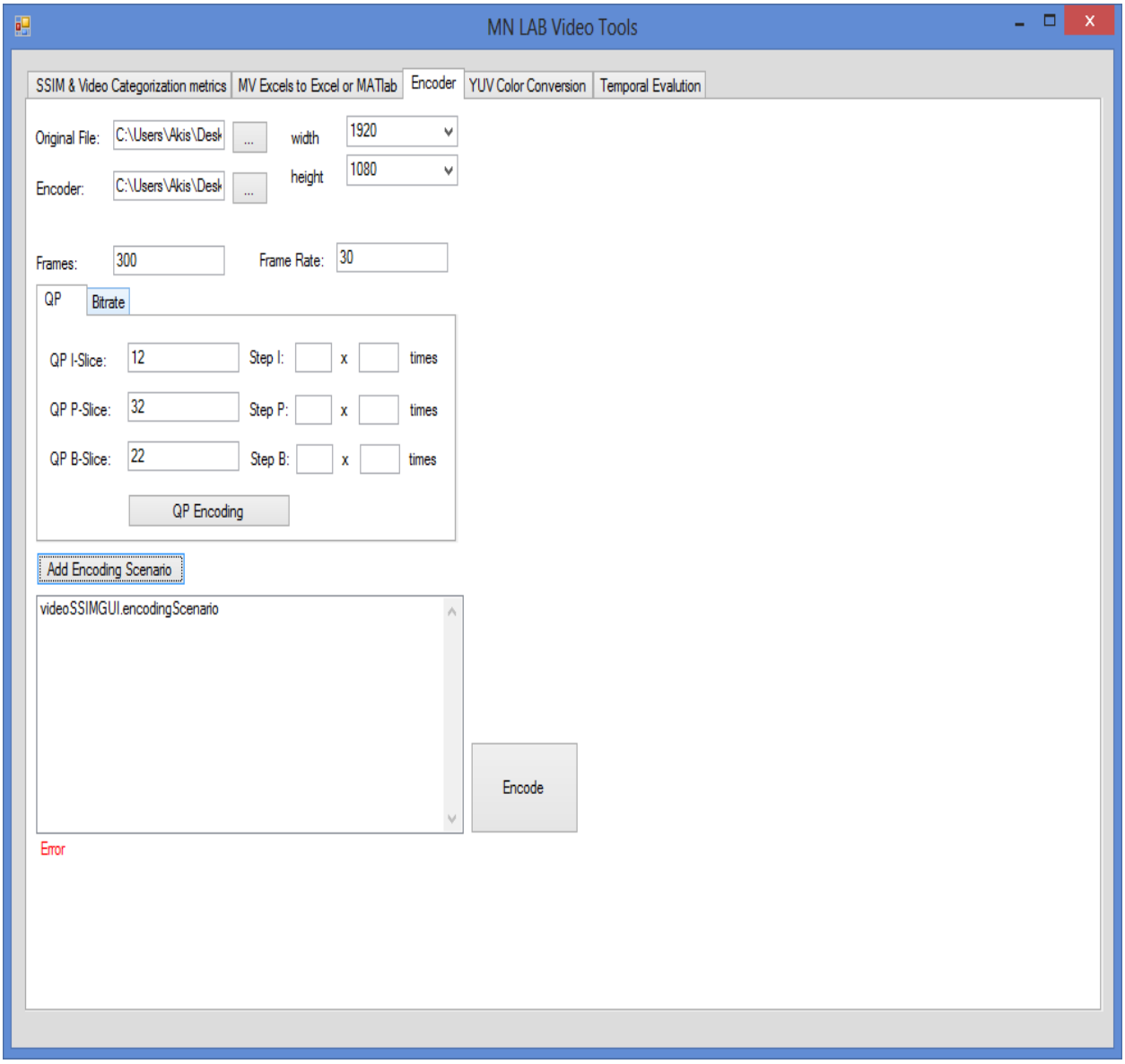

Figure 28 Example video encoding case.

In figure 28, a video signal and its encoding parameters are selected. This video encoding case refers to a QP related encoding, meaning we choose the QP value for encoding the I, P and B slices. After we have selected the desired parameters we add the encoding scenario to the list. Note that we can create multiple encoding scenarios and execute one after the other. In this manner, we do not have to waste a lot of time supervise the process, in order to put the next video encoding process.

In the next figure we see example output of the selected encoding scenario.

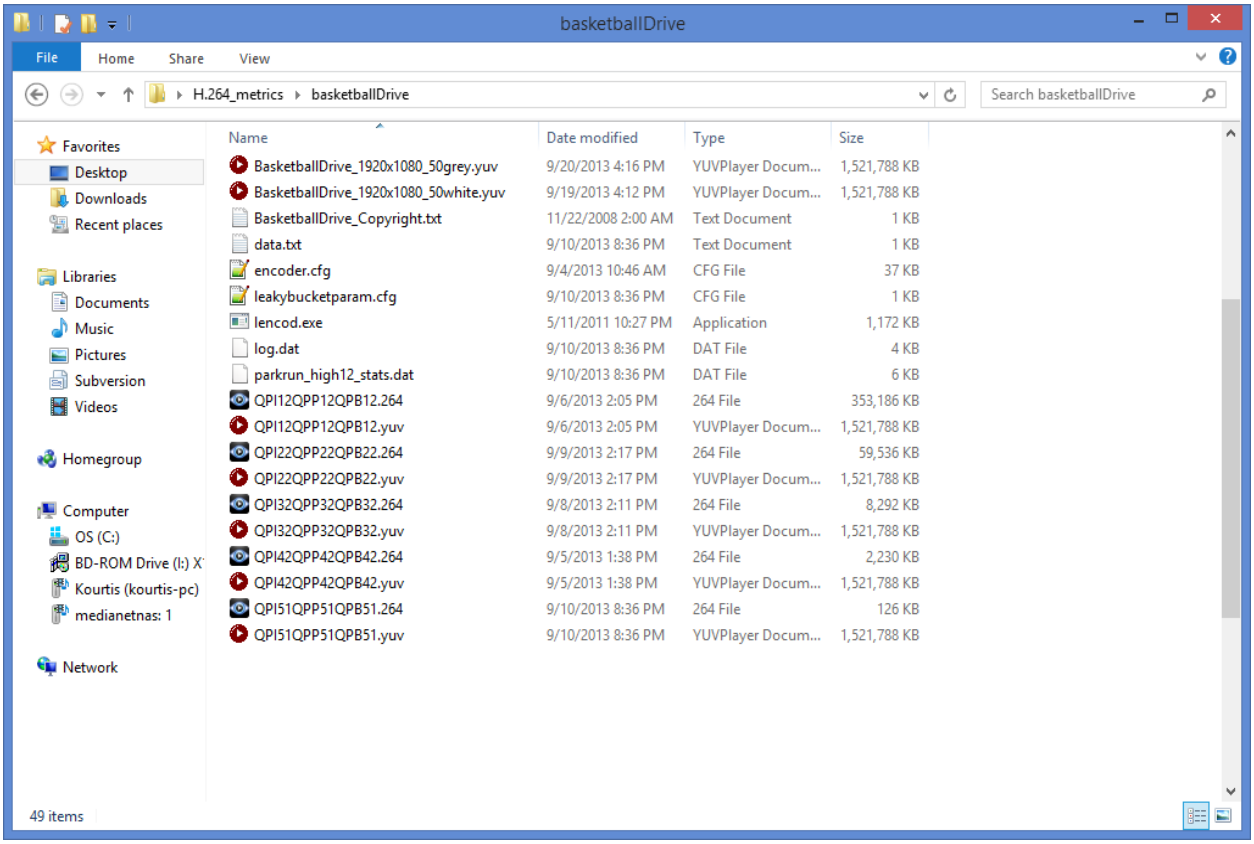

Figure 29 QP related encoding scenario output results.

In Figure 29 we observe the other functionality of the video encoding module of the MNLab Video Tools program.

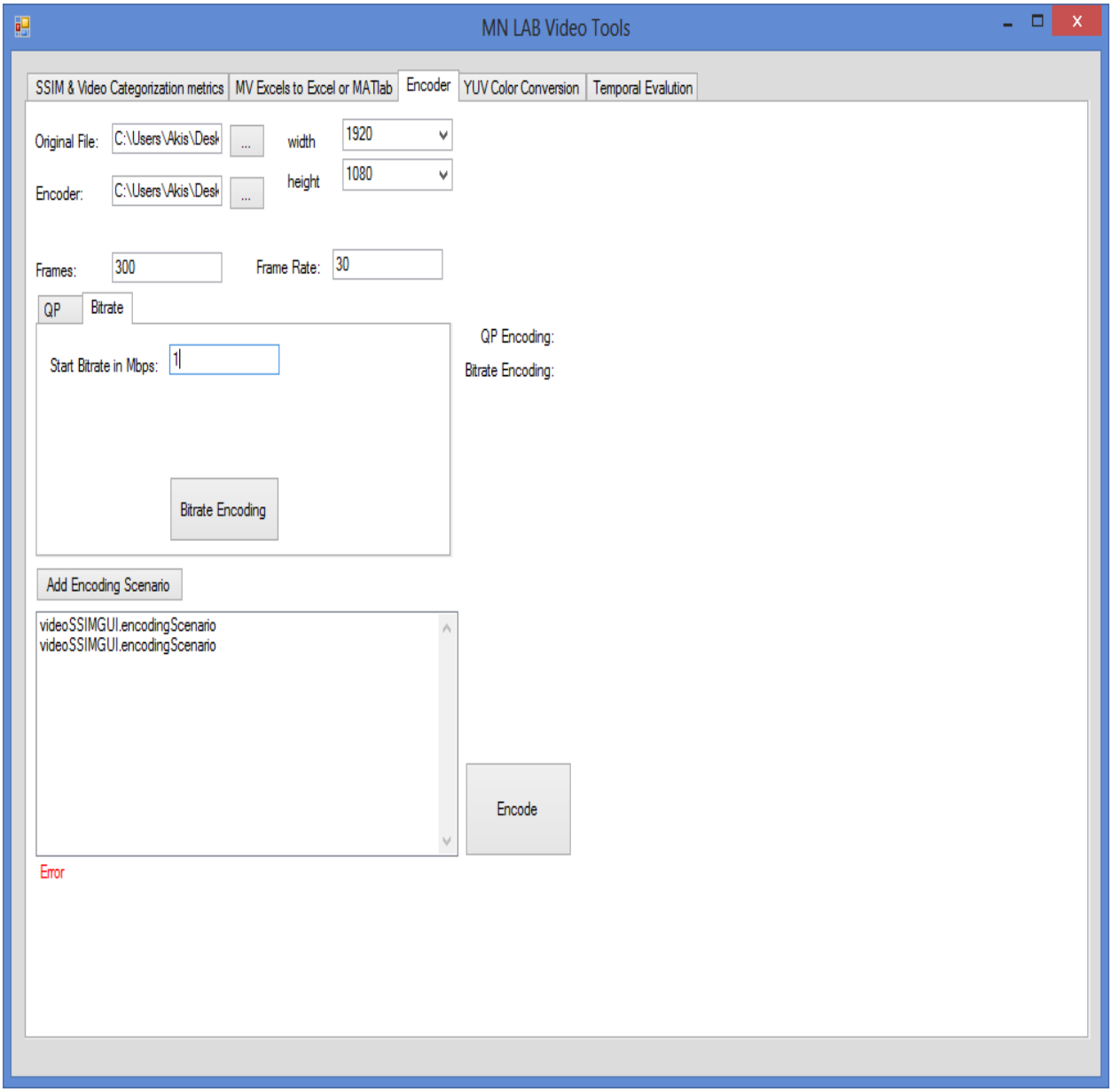

Figure 30 Bitrate related video encoding

As we can see on the previous figure we can also choose the bitrate of the encoded video signal by the Bitrate parameter functionality of the program.

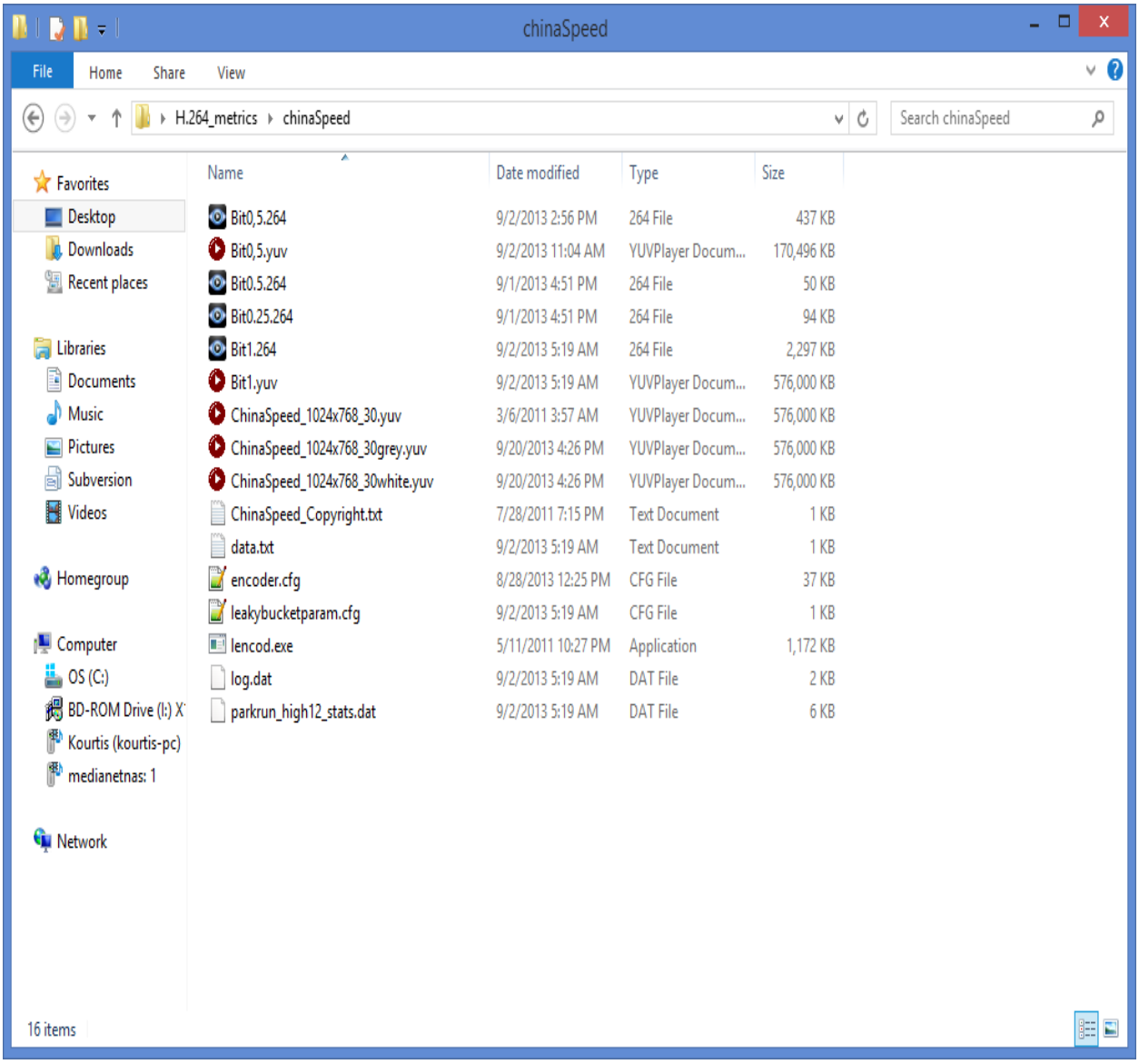

Figure 31 Bitrate Encoding output examples.

Figure 31 presents an example of a Bitrate related video encoding scenario. As we can see we encoded signals of the original video in various bitrates.

5.3 Validation and Evaluation Process

The video differentiation process requires a large amount of low-level information from the video signal under test. This creates the challenging task to validate the information gathered and verify its correctness compared to the original signal.

For this reason, during the course of the experimental process, at the stage of the motion vector value extraction, the program CodecVisa [8] was used in order to verify that our motion vector value results correspond to the actually decoded signal's motion vector values. CodecVisa is a widely used video analysis program used by many large and successful multimedia service enterprises to analyze their video encoder output.

In the next figures, we present a few snapshots of the program's functionality and how this assisted to our experimental validation process.

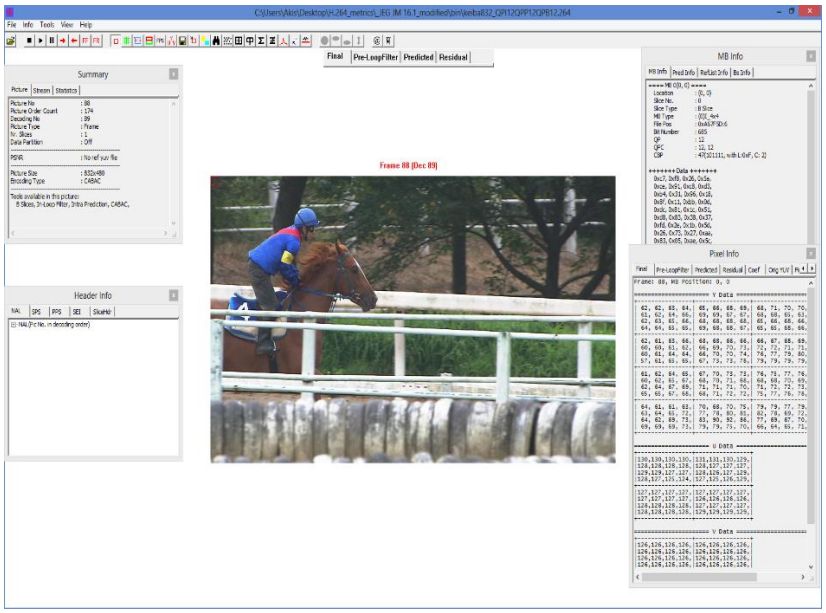

Figure 32 Overview of the CodecVisa program.

In figure 32 we are presented with the main screen of the CodecVisa program and its several video feature analysis screens. The video sequence on screen is the Keiba.

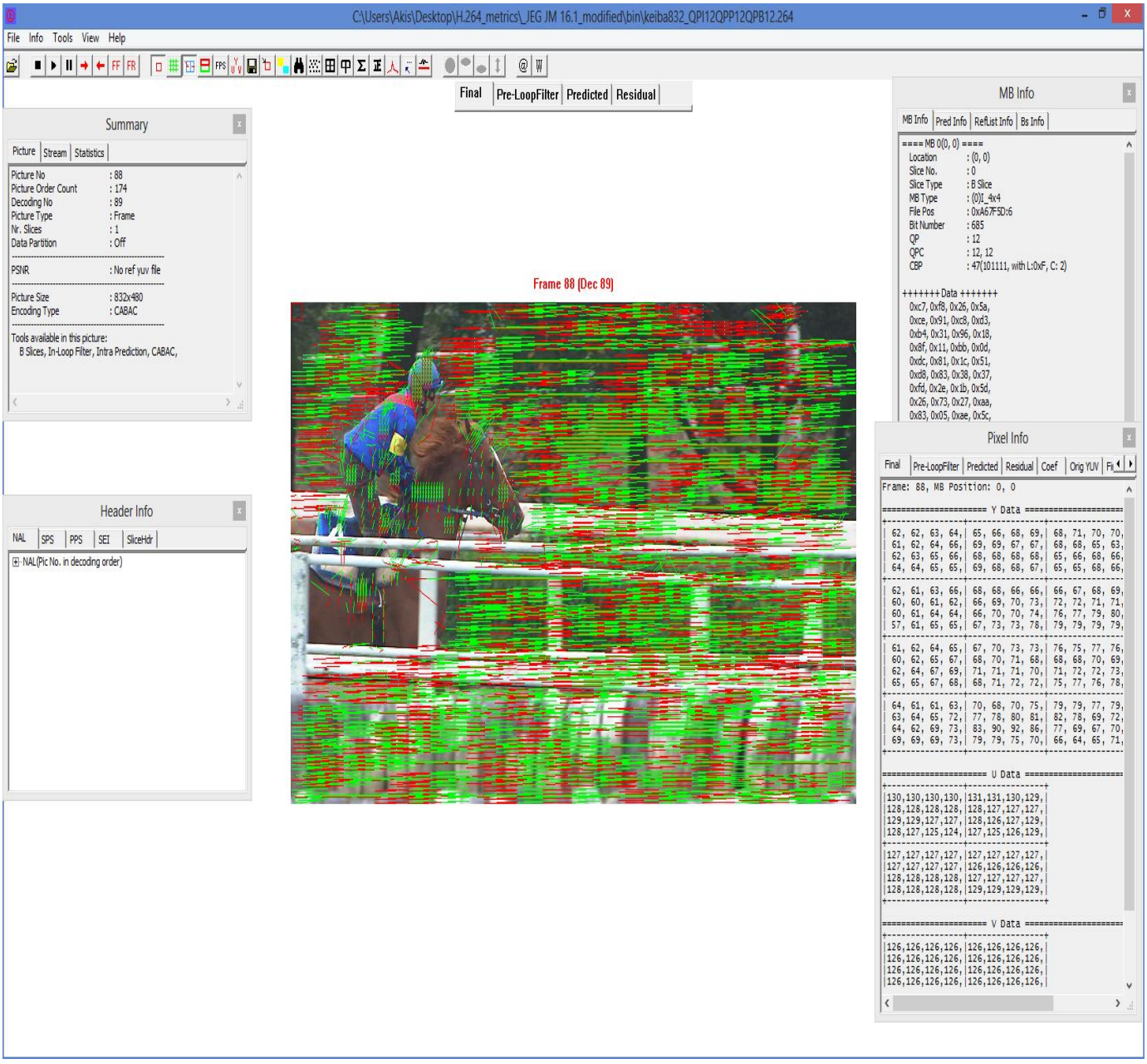

Figure 33 Motion Vector Visualization on the Codec Visa video tool.

In figure 33 we observe the CodecVisa program after it has executed the motion vector visualization command on the Keiba video sequence. Notice the MB (Macroblock) tab where we can see all the decoded information of a selected macroblock for the current frame. The CodecVisa tool was mainly used for verification of the MN Lab Video Tools program output results.

5. CONCLUSIONS

This report proposes a novel video differentiation method based on individual video motion vector information. It is shown that the proposed technique achieves to capture and represent successfully the unique content of each video signal under test. Additionally, the proposed algorithm does not require highly complex calculations. Furthermore, it can be used as an efficient content aware module on video transmitting systems, to enhance the providing service as a video content characterization tool that can improve the encoder's efficiency and increase the perceived video quality. More specifically, it can be used for a MPEG-DASH video scaling selector, where depending on the temporal, or the spatial activity of the transmitted video signal, we can decide whether to scale down the video resolution highly motion compensated videos, or drop the frame rate on videos with a high valued content residual, based on the proposed video differentiation results.

6. FUTURE WORK - EVALUATION

Future work includes continuation of this work implementing an encoding parameter selector module for the MPEG-DASH protocol. Early research and work point towards an implementation based upon the open source multimedia framework GPAC [16]. In more detail GPAC includes the multimedia packager MP4BOX [17]. MP4BOX can perform various manipulation tasks on a wide variety of multimedia files. Our focus will be on the video file format manipulation, and more specifically modifying the preparation function of HTTP Adaptive Streaming content. Next steps include the cooperation and communication between our motion differentiation function and the MP4BOX function, in order to create the optimal spatiotemporal (Video resolution or Frame rate modification) adaptations of various video signals, based on their content dynamics. After completing the implementation of a functioning video differentiation system, further experiments can be performed, in order to evaluate, measure, and assess the perceived quality of the system's outcome. Nonetheless, more advanced motion information representation techniques, along with more efficient and advanced motion vector information fusion

techniques. It would be interesting to investigate new methods for calculating even more accurately and more descriptively the video motion information mainly on the temporal plane of the signal under test. Additionally, a database functionality could be added in order to compare video motion information in a large scale. Another aspect is to add into the evaluation process more videos, especially not reference ones and examine the commercial performance of the proposed method. The previous defined steps are a guideline towards creating a video content gnostic framework, which can provide the best available video service for the user, depending on the network conditions, but also be used and as a video recommendation service as videos with similar content dynamics can appear close to the user's preferences.

REFERENCES

[1] ISO/IEC 23009-1:2012

Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats

[2] http://tools.ietf.org/html/rfc3550

[3] http://iphome.hhi.de/suehring/tml/

[4] Sodagar, I., 2011. The MPEG-DASH Standard for Multimedia Streaming Over the Internet.MultiMedia, IEEE (Volume:18 , Issue: 4).

[5] ftp://ftp.tnt.uni-hannover.de/testsequences

[6] "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVTG050, 2003.[7] "A Survey on Video Coding Principles and Standards", M. A. Kourtis, H. Koumaras, to be published in Multimedia Networking and Coding: From Capture to Display Edited by Reuben A. Farrugia and Carl J. Debono, IGI Global Publishing.

[8] http://www.codecian.com/

[9]http://iphome.hhi.de/suehring/tml/JM%20Reference%20Software%20Manual%20(JVT-AE010).pdf

[10] Richardson, I.E.G. (2003). H.264 and MPEG-4 Video Compression, Wiley, 2003, ISBN 0-470-84837-5

[11] Winkler, S. (2005). Digital Video Quality – Vision Models and Metrics,Wiley, 2005, ISBN 0 470 02404 6

[12] Held, G. & Marshall, T. R. (1991). Data Compression, Wiley, 1991, ISBN 0 471 92941 7

[13] Huffman, D. (1952). A method for the construction of minimum redundancy codes, Proceedings of the IRE, 40, pp. 1098–1101.

[14] Witten, H. & Neal, M. & Cleary G. (1987). Arithmetic coding for data compression, Communications of the ACM, 30 (6), pp.520-540.

[15] N. Cranley and L. Murphy, “Incorporating User Perception in Adaptive Video Streaming Systems”, in Digital Multimedia Perception and Design (Eds. G. Ghinea and S. Chen), published by Idea Group, Inc., May 2006. ISBN: 1-59140-860-1/1-59140-861-X

[16] http://gpac.wp.mines-telecom.fr/home/

[17] http://gpac.wp.mines-telecom.fr/mp4box/