TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Yoshihisa Furushita 194244IVCM

Sources of artifacts in video

Master's Thesis

Supervisors: Dr Matthew Sorell, PhD

Center for Digital Forensics and
Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Pavel Tsikul, M.Sci

Center for Digital Forensics and
Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia Teaduskond

Yoshihisa Furushita 194244IVCM

Artefaktide allikad videos

Magistritöö

Juhendaja: Dr Matthew Sorell, PhD

Center for Digital Forensics and
Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Pavel Tsikul, M.Sci

Center for Digital Forensics and
Cyber Security
Tallinn University of Technology
Tallinn, Estonia

Tallinn 2021

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Yoshihisa Furushita

14.05.2021

Abstract

Video forensics experts need to explain video evidence presented in court. However, there is no established method to explain whether a certain feature of a video is real or a result of compression. This paper proposes a tool to visualize how parts of a video were encoded and show whether video artifacts are due to coding or actual features. This research also extends to Deepfake video analysis.

List of Abbreviations

PCAST	President's Council of Advisors on Science and Technology
GOP	Group Of Pictures
MSE	Mean Square Error
MPE	Max Pixel Error

Table of contents

List of Figures

List of Tables

1 Introduction

Video data would be massive if saved without any processing, so it is necessary to compress it to be easily stored, sent, and received. There are two types of compression techniques, which are lossless compression and lossy compression.

Lossless compression organizes redundancy in data. Since no information is lost after compression, it is used for numerical data, documents, programs, and other data whose contents must not change.

Lossy compression uses a technique that reduces the file size by discarding less critical information and is often used for video data, discarding information which the eye and brain does not normally care about in casual viewing. This can cause significant challenges when such compression is used in video as evidence.

If the compressed data fails to generate enough data to reproduce the original video, artifacts such as distortions that were not present in the original video may appear. It is also recognized that compression distorts spatial artifacts of interest, such as faces and license plates Although there is a lot of literature on video compression artifacts, Zeng K et al. review the perceptual artifacts caused by video compression in modern video coding techniques and organize the spatial and temporal perceptual distortions commonly observed in compressed video.[1]

1.1 Problem statement

Video forensics experts need to explain the artifacts in the video evidence presented in court to the jury. Non-experts cannot determine whether the vertical lines on a car door or the outline of a suspect's nose are real or an illusion created by artifacts. These features in the video footage may affect the verdict. However, there are no established means of efficiently explaining the artifact to non-experts in a limited amount of time.

1.2 Purpose statement

The primary goal of this research is to find out how the features in the video were generated. To solve this problem, I developed a specialist visualization tool that loads a video, visualizes video coding parameters such as motion vectors and macroblocks, and analyzes features in the frame, parsing the movements of video encoding parameters. Developing a scaled-up or automated product is not the goal of this research. The purpose, rather, is for an expert examiner to dissect video data structure.

The videos used in the study were limited to those encoded in MPEG-4, one of the most commonly and widely deployed video and audio compression standards, with a resolution of 1920×1080 pixels. [1]All the videos were available for free on the Internet.

Compression artifacts to be analyzed were limited to repetition artifacts due to motion vectors, shape distortion artifacts due to approximation of residual block coding, and optical artifacts in low light due to long exposure. Regarding the development of tools, we do not use advanced techniques for visualization because we expect to use tens to hundreds of frames (up to one minute) of video in this study. Also, for security and privacy reasons, the tool was designed to work only in the local environment of the forensic expert's PC or laptop.

1.3 Overview of the thesis

This paper mainly presents the prototype tool. After introducing the literature review (Section 2), the mechanism of the video compression process is explained (Section 3). Then, I describe an overview of the prototype tools employed and the functions (Section 4) and show analysis using the prototype tool (Sections 5 and 6). Finally, I conclude by presenting the current limitation of the prototype tool and the future of the prototype tool.

[1] https://www.pexels.com/videos/

2 Literature review

The following papers outlined below are some of the more recent works or information in (1) Video forensics, (2) Video analysis software, (3) FFmpeg toolbox, and (4) Forensic science in criminal court.

2.1 Video forensics

Video forensics has many unanswered research questions because the compression process on video data erases traces of tampering and makes it impossible to recover the processing history.

Milani S et al. presented a method for detecting macroblocks, estimating motion vectors, and analyzing the number of compressions as coding information of a video.[2] Many studies have also been reported on detecting double compression in a video, as it can reveal part of the processing history of the video.[3,4,5] Furthermore, Wang and Farid showed that recompression after deleting or adding frames results in detectable artifacts in MPEG video.[6] Stamm et al. proposed an anti-forensic technique based on this work.[7]

Research on digital video forgery detection can be categorized into spatial and temporal domains. For forgery detection in the spatial domain, Kobayashi et al. proposed a method to identify the tampered area by determining the characteristics of compression artifacts for each pixel value in a stationary frame.[8] Furthermore, Zhou et al. proposed a method to detect tampered regions by pre-extracting features in an I-frame using the discrete cosine transform.[9] For forgery detection in the temporal domain, Dong et al. and Su et al. proposed a method to detect video tampering by using artifacts generated by blocking artifacts propagating from the I-frame to subsequent frames.[10,11]

In video forensics, there is much research on video tampering detection to verify the authenticity of video images. However, virtually no papers analyze whether artifacts in the video are caused by video coding in the context of digital forensics analysis.

2.2 Video analysis software

Jerian M et al. developed an image processing software for analyzing crime scene videos and images. The tool can read the writing on a car license plate through frame averaging, display images clearly, and remove compression artifacts.[12] Al Ameen Z et al. describe several software applications that remove these artifacts to address the problem that important information present in an image can be hidden by artifacts (image noise, lighting defects, blurring). It also introduces tools for visibility enhancement and tamper detection.[13] The paper mentions Amped five, a forensic image and video enhancement software developed by Amped software, and Amped Authenticate, photo analysis and tamper detection software.[14] However, there were no papers on tools to analyze how artifacts in a video are created.

2.3 FFmpeg toolbox

I examined papers on video analysis systems that use FFmpeg, free software for recording, converting, and playing video and audio. Lei X et al, introduces the design and implementation of video stream analysis system by using FFMPEG.[15] Ryu C et al. proposed an extensible video processing framework that modifies the FFmpeg library to optimize it for the Hadoop environment.[16] Zeng H et al. used FFmpeg to encode/decode H264 and HEVC video, and to achieve format-to-format conversion between H264 and HEVC formats.[17]

2.4 Forensic science in criminal court

In 2016, the President's Council of Advisors on Science and Technology (PCAST) released a report on the use of forensic analysis and expertise in criminal trials.[18] The report explains the scientific validity of feature-comparison methods used in DNA analysis, fingerprint analysis, and firearms analysis, and so on.

PCAST argued that feature-comparison methods belong to the scientific field of metrology (measurement science). The methods can be considered valid if they adhere to the standards set by PCAST and that error rates can only be established using properly designed studies.

However, in 2021, the U.S. Department of Justice refuted these claims.[19] First, the feature comparison method as currently practiced relies on the subjective judgment of the analyst rather than scientific analysis and mathematical calculation and does not belong to metrology. In addition, PCAST's claim that feature-comparison methods can be validated using the standards set by PCAST is inconsistent with international standards. There are no single accepted means of validating scientific methods. Finally, error rates are related to the assumptions, choices, and conditions of the experiment, and it is not scientifically valid to generalize to the scenario set forth by PCAST.

This tool aims to assist a video analyst in dissecting the construction of a video of interest. It is not satisfactory for a subjective interpretation concerning the content of a video if the limitations of its encoding, and the resultant distortions and artifacts, cannot be established. Therefore, considering the current background on evidence analysis in criminal trials, the development of the video forensics tool conducted in this study is significant.

3 How video compression works

Lossy compression introduces some unique ways to remove unnecessary information and redundancy from data efficiently.

First, color information is replaced with less informative pixel value because the human eye is sensitive to brightness (luminance) but less sensitive to color (chrominance). Second, a video appears to be moving by displaying a series of still images called "frames." There is temporal repetition with only slight changes in the image between some successive frames. There is also spatial repetition within each frame with the same or similar colors used in many areas. All these can be removed as redundancy.

These methods are called inter-frame prediction and intra-frame prediction, respectively, and are the underlying techniques of current video compression technologies.

3.1 Inter-frame prediction

Inter-frame prediction is a technique to improve coding efficiency by removing redundancy that exists between successive frames. In this technique, the pixel value at the same position in the previous frame is used as the predicted value of the pixel value in the target frame.

3.1.1 Motion compensation

The simple frame-to-frame prediction described above alone cannot be compressed efficiently because the amount of information becomes too large when camera work or object motion is included. Motion compensation is the process of calculating the motion vectors (horizontal and vertical displacements) of the areas where objects have moved in the frames before and after the target frame and creating a predictive frame from the target frame and the motion vectors. This method compresses only the difference between the predicted frame and the actual image, thereby achieving an efficient compression process. Figure 1 shows an example of motion compensation.

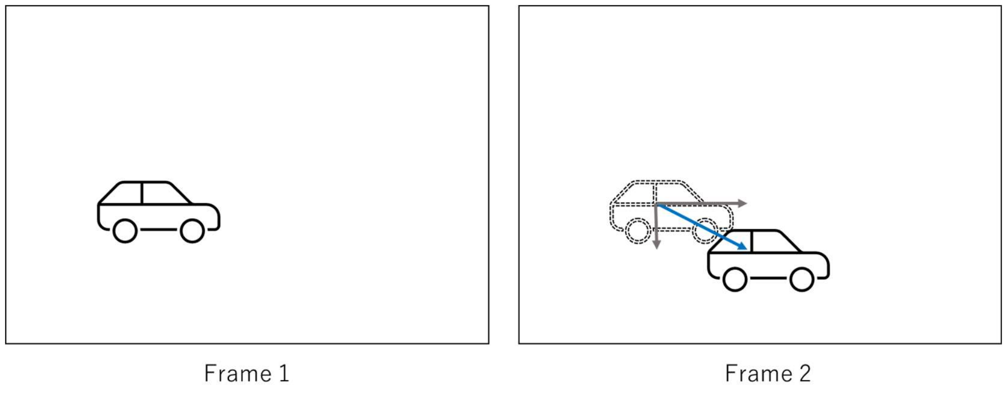


Figure 1 Example of motion compensation

3.1.2 Motion vector estimation

This section explains how to detect motion vectors in a frame. First, pixel values (16 x 16 = 256 pixels) in a 16 x 16 area (macroblock) at the same coordinates in the target frame and reference frame are compared, and the sum of the absolute differences is calculated (or some other value such as mean square error). Second, while shifting the macroblock position in the reference frame one by one, the macroblock in the target frame is compared with the pixel value of the macroblock in the entire reference frame. Then, the macroblock position for which the sum of the differences in pixel values with the macroblock in the reference frame is the smallest is used as the motion vector in a certain target frame. The same calculation is performed for all macroblocks in the target frame. Figure 2 shows an example of motion vector estimation.

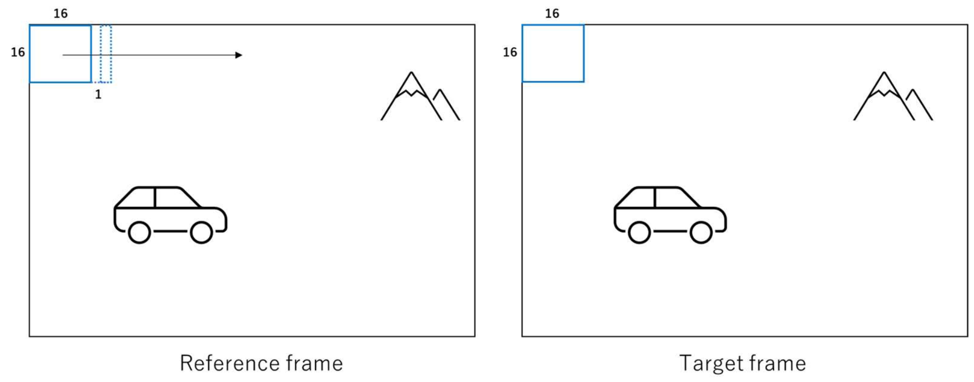


Figure 2 Example of motion vector estimation

As shown in Figure 3, a region of a macroblock can be coded by dividing it into two 16×8, two 8×16, or four 8×8 regions in addition to one 16×16. Also, an 8×8 block can be encoded as one 8×8, two 4×8, two 8×4, or four 4×4 sub-blocks.[20] This method increases the amount of computation but allows for more precise motion compensation.

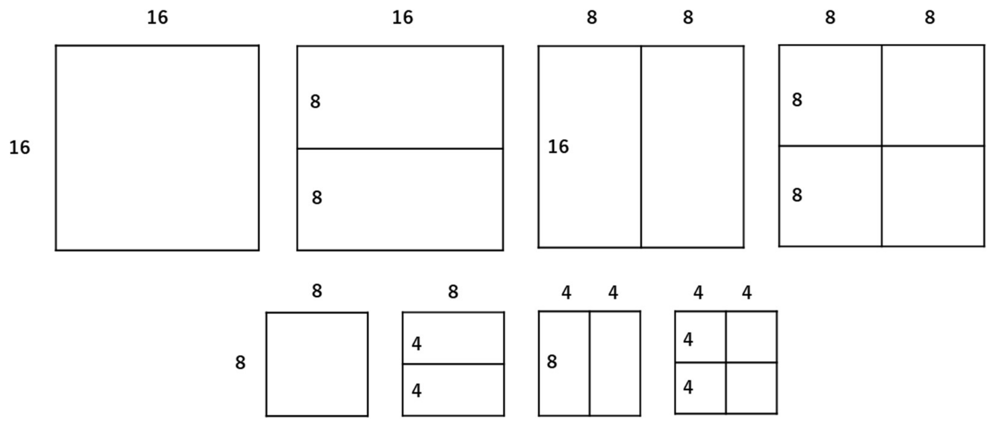


Figure 3 Example of macroblock partitions for motion compensation

Since motion vector estimation is a large part of the video coding process, various algorithms have been researched and developed to reduce its computational complexity. Sorell M proposed a new method to identify motion vector estimation algorithms and use them as fingerprints in the source firmware or software in terms of video forensics.[21]

3.1.3 Frame types

In MPEG4 video, target frames are classified into three types. As shown in Figure 4, I-frames, P-frames, and B-frames, and a set of consecutive frames containing one I-frame is called a GOP (Group of Picture).

P-frame is a frame predicted by motion compensation referring to a past frame. B-frame is a frame predicted by motion compensation from two frames, one in the past and one in the future, and the prediction accuracy is improved by referencing from both directions. An I-frame does not perform inter-frame prediction and has complete information without depending on other frames. The frame is referenced when accurate decoding is not possible due to transmission errors.

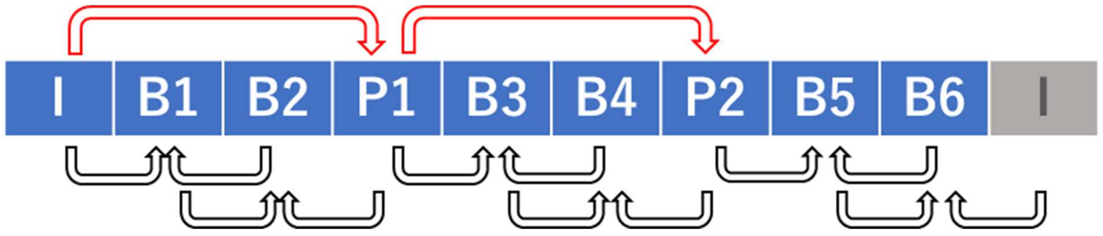


Figure 4 Group of Picture (GOP)

3.2 Intra-frame prediction

The adjacent pixels in a frame are often almost identical, and such redundancy is compressed by encoding the difference in pixel value between the specific area. This method is called intra-frame prediction. MPEG-4 AVC/H.264 divides the pixel values in the target frame into 16×16 macroblocks and 4×4 subblocks and predicts target pixel values from the encoded pixel values in the adjacent blocks.

As shown in Figure5, in intra-prediction with 4×4 sub-blocks, pixel values in the target sub-block are predicted from the values of four pixels in the left block (I, J, K, L), four pixels in the upper block (A, B, C, D), four pixels in the upper right block (E, F, G, H), and one pixel in the upper left block (M). There are nine prediction modes and smaller numbers are assigned in order of the prediction method with the highest frequency of occurrence. DC applies the average value of 8 pixels (A, B, C, D, I, J, K, L) to the target block.[22] Similarly, intra prediction of 16×16 macroblock has four prediction modes as shown in Figure 6, which are not as accurate as 4×4-pixel intra prediction but are effective for predicting flat images and require fewer bits.

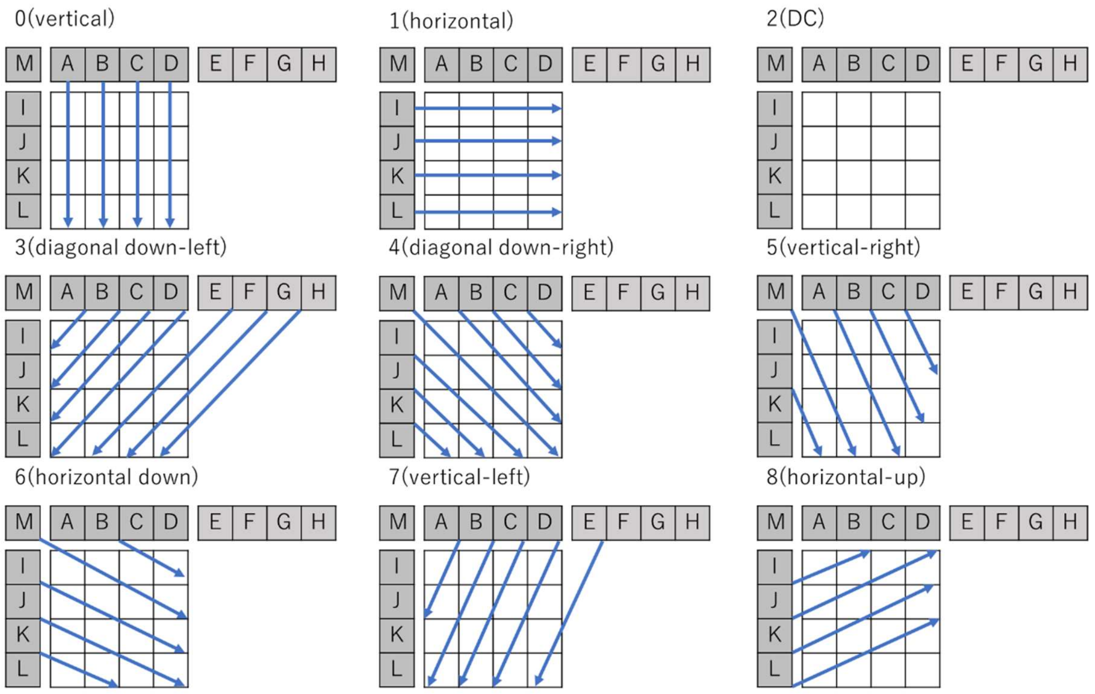


Figure 5 Intra 4×4 prediction modes

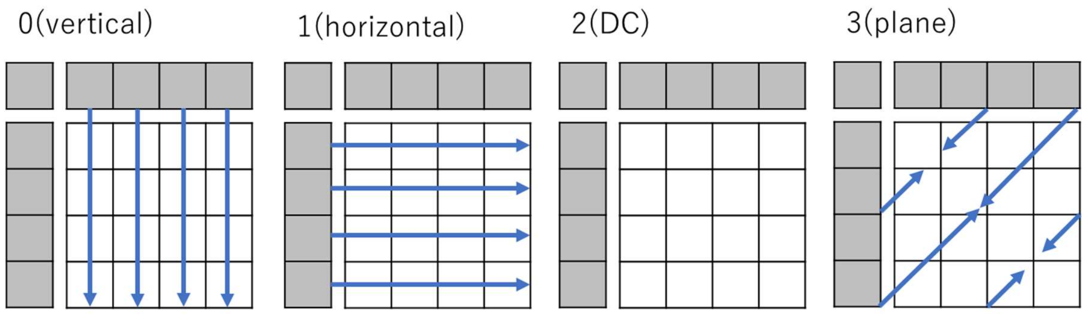


Figure 6 Intra 16×16 prediction modes

4 Prototype tool

The forensic video expert needs to understand the details of the components of the video. However, it is difficult to explain everything mentioned in the previous chapter in a court of law and make it understandable to a non-expert in a limited amount of time. It would help forensic video experts if they could visualize how an image is coded with one tool.

To solve this problem, I chose PyQt as the development environment. [23,24] First of all, it is environment-independent and can run on Mac, Windows, and Linux without any individual support. Secondly, since it is programmed in Python, even those with low programming skills can understand and manage it. All functions are documented, and the code is not difficult to customize with a wide variety of GUI widgets and features. It is possible to extend the functions using various machine learning frameworks such as "numpy" and "pandas." Also, FFmpeg, open-source software that encodes and decodes multimedia such as video and audio, is used for some functions. [25,26,27]

This chapter demonstrates the prototype tool's function with some examples. The tool consists of a "Video Player" for video analysis and an "Image Player" for image analysis. The video player can video playback, macroblock processing, motion vector processing, and frame segmentation. The image player can image display, frame prediction, mean square error (MSE) comparison, and motion vector field visualization. Figure 7 shows the operation screen of the proposed tool as an example. The source code and documentation guide are available in GitHub.[28]

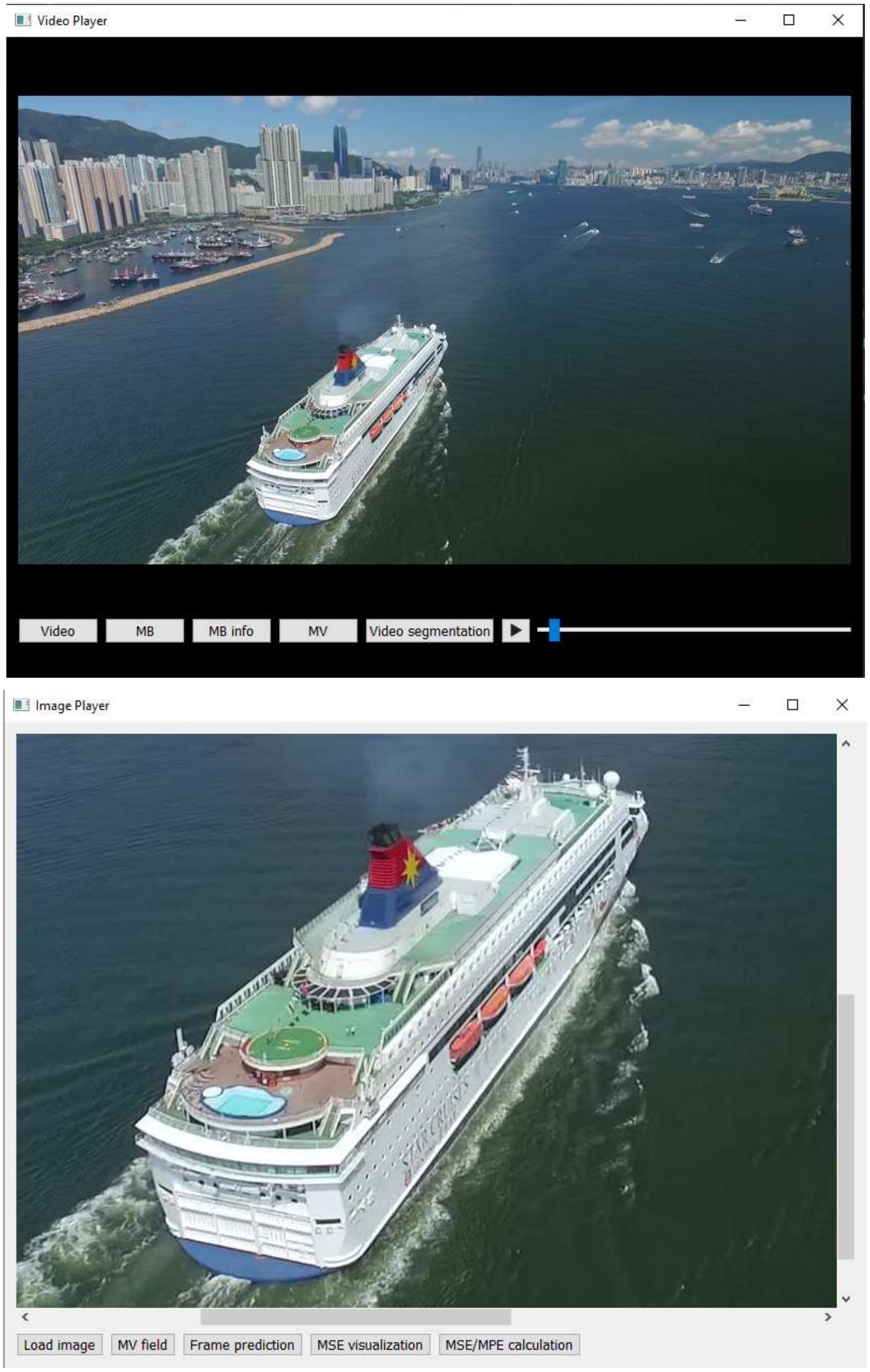


Figure 7 Screenshot of the proposed tool

4.1 Video Player

Video is composed of many frames, and complex compression techniques are used, but it is impossible to understand the mechanism just by looking at it. The "video player" can visualize the coding information and structure of the video, and these functions were implemented using the FFmpeg functions; one function can also be output in text format for analysis.

4.1.1 MB

The MB function is to visualize the macroblocks for all frames of a video. This function's purpose is to show how a certain video and a frame were coded. Figure 8 shows an example of macroblock visualization. The user can identify the coding mechanism of each macroblock by analyzing the displayed chrominance.

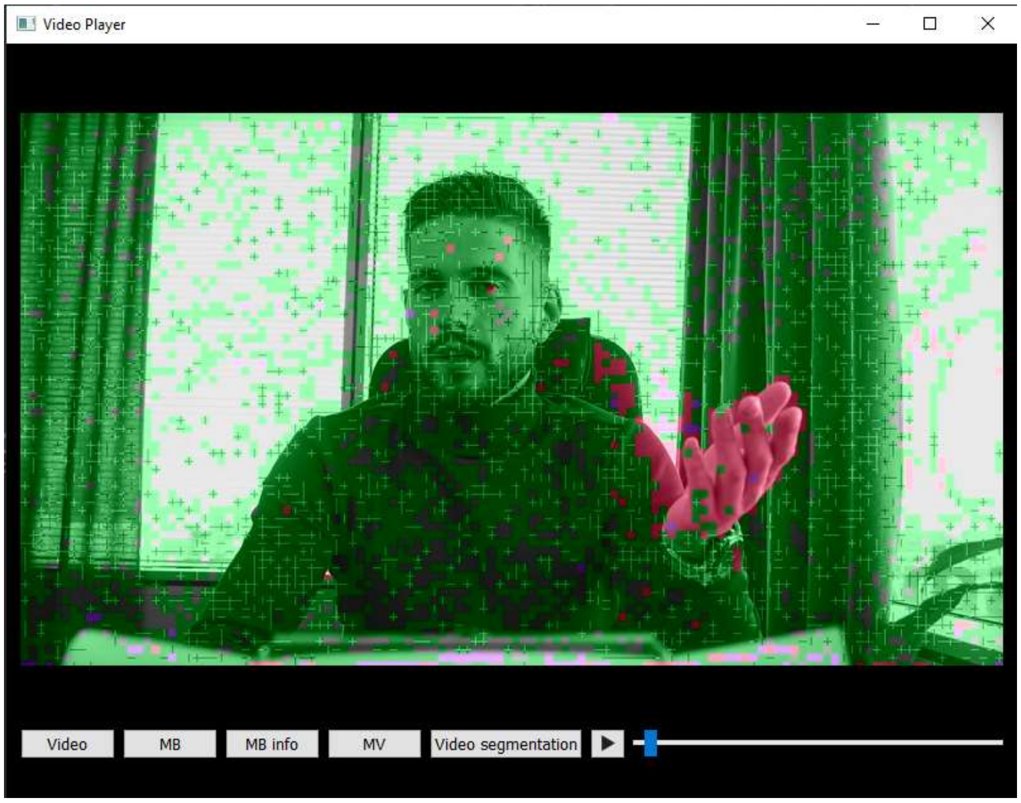


Figure 8 Example of macroblock visualization

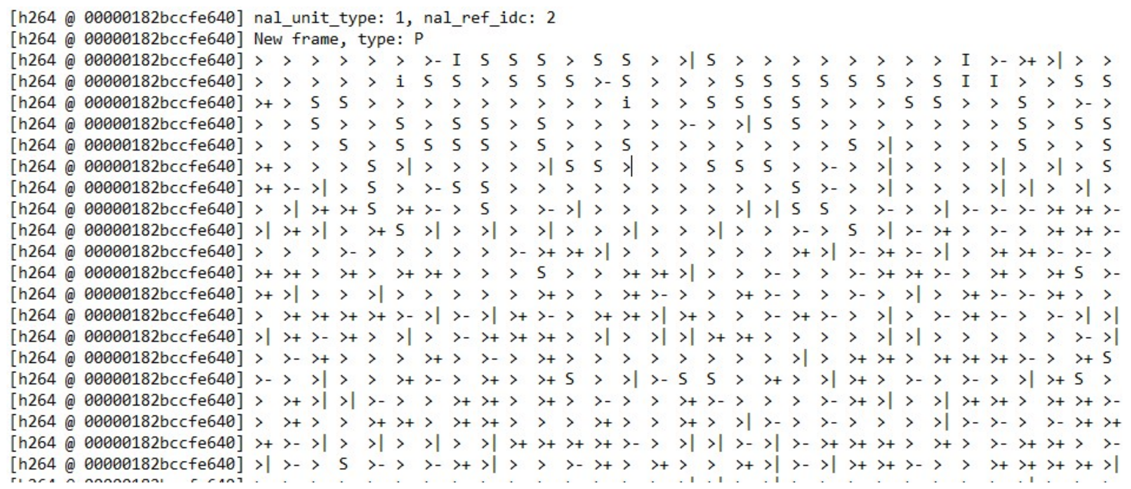
Table 1 shows the types of macroblocks, their coding chrominance, and symbols.[29, 30] According to this figure, we can see that most of the frames in Figure 8 were predicted from the previous frame. Sub-blocks were also used in some of them. Also, 4×4 Intra prediction is mainly used for the hands.

Color	Macroblock Type Condition	Explanation	Symbol
🟥	**IS_PCM (MB_TYPE_INTRA_PCM)**	**Lossless (raw samples without prediction)**	**P**
🟪	**(IS_INTRA && IS_ACPRED) || IS_INTRA16x16**	**16×16 Intra prediction**	**A / i**
🟥	**IS_INTRA4x4**	**4×4 Intra prediction**	**I**
🟫	**IS_DIRECT**	**No motion vectors are sent (B slices)**	**D / d**
🟫	**IS_GMC && IS_SKIP**	**16×16 Skip macroblock (P or B slices)**	**g**
🟩	**IS_GMC**	**Global motion compensation (not relevant for H.264)**	**G**
🟩	**!USES_LIST(1)**	**Reference to past (List 0, P or B slices)**	**>**
🟦	**!USES_LIST(0)**	**Reference to future (List 1, B slices)**	**<**
🟦	**USES_LIST(0) && USES_LIST(1)**	**Reference to past and future (List 1 & 2, B slices)**	**X**
N/A	**IS_SKIP**	**Skip macroblock**	**S**
N/A	**IS_8X8**	**Subblock 8×8**	**+**
N/A	**IS_16X8**	**Subblock 16×8**	**-**
N/A	**IS_8X16**	**Subblock 8×16**	**|**

Table 1 Possible colors and the associated macroblock types

4.1.2 MB info

As shown in Figure 9, the MB info function outputs all macroblock information in a P-frame to a text file with symbols representing how each block is coded. Each symbol corresponds to a 16×16 macroblock. In other words, a frame size of 1920×1080 would display 8160 symbols. The part that was predicted from the past frame or future frame may show two symbols (>+, >-, >|) on it. These symbols indicate that inter-frame prediction was performed using sub-blocks, as shown in the previous chapter.



```
[h264 @ 00000182bccfe640] nal_unit_type: 1, nal_ref_idc: 2
[h264 @ 00000182bccfe640] New frame, type: P
[h264 @ 00000182bccfe640] >  >  >  >  >  >  >- I  S  S  S  >  S  S  >  >| S  >  >  >  >  >  >  >  >  I  >- >+ >| >  >
[h264 @ 00000182bccfe640] >  >  >  >  >  i  S  S  >  S  S  S  >- S  >  >  >  S  S  S  S  S  S  >  S  I  I  >  >  S  S
[h264 @ 00000182bccfe640] >+ >  S  S  >  >  >  >  >  >  >  >  >  i  >  >  S  S  S  S  >  >  >  S  S  >  >  S  >  >- >
[h264 @ 00000182bccfe640] >  >  S  >  >  S  >  S  S  >  S  >  >  >  >  >- >  >| S  S  >  >  >  >  >  >  >  S  >  S  S
[h264 @ 00000182bccfe640] >  >  >  S  >  S  S  S  S  >  S  >  >  S  >  >  >  >  >  >  >  S  >| >  >  >  >  S  >  >  S
[h264 @ 00000182bccfe640] >+ >  >  >  S  >| >  >  >  >  >| S  S  >| >  >  S  S  S  >  >- >  >| >  >  >  >| >  >| >  S
[h264 @ 00000182bccfe640] >+ >- >| >  S  >  >- S  S  >  >  >  >  >  >  >  >  >  >  S  >- >  >| >  >  >  >| >| >  >| >
[h264 @ 00000182bccfe640] >  >| >+ >+ S  >+ >- >  S  >  >- >| >  >  >  >  >  >| >| S  S  >  >- >  >| >- >- >- >+ >+ >-
[h264 @ 00000182bccfe640] >| >+ >| >  >+ S  >| >  >| >  >| >  >  >| >  >  >| >  >  >- >  S  >| >- >+ >  >- >  >+ >+ >-
[h264 @ 00000182bccfe640] >  >  >  >- >  >  >  >  >  >- >+ >+ >| >  >  >  >  >  >  >+ >| >- >+ >- >| >  >+ >+ >- >- >
[h264 @ 00000182bccfe640] >+ >+ >  >+ >  >+ >+ >  >  >  S  >  >  >+ >+ >| >  >  >- >  >  >- >+ >+ >- >  >+ >  >+ S  >-
[h264 @ 00000182bccfe640] >+ >| >  >  >| >  >  >  >  >  >+ >  >  >+ >- >  >  >+ >- >  >  >- >  >| >  >+ >- >- >+ >  >
[h264 @ 00000182bccfe640] >  >+ >+ >+ >+ >- >| >- >| >+ >- >  >+ >+ >| >+ >  >  >- >+ >- >  >| >  >- >+ >- >  >- >| >|
[h264 @ 00000182bccfe640] >| >+ >- >+ >  >| >  >- >+ >+ >+ >  >| >  >| >| >+ >+ >  >  >  >  >| >| >  >  >  >  >  >- >|
[h264 @ 00000182bccfe640] >  >- >+ >  >  >  >+ >  >- >  >+ >  >  >  >  >  >  >  >  >| >  >+ >+ >  >+ >+ >+ >- >  >+ S
[h264 @ 00000182bccfe640] >- >  >| >  >  >+ >- >  >+ >  >+ S  >  >| >- S  S  >  >+ >  >| >+ >  >- >  >- >  >| >+ S  >
[h264 @ 00000182bccfe640] >  >+ >| >| >- >  >  >+ >+ >  >+ >  >- >  >  >+ >- >  >  >  >- >+ >| >  >| >+ >+ >  >+ >+ >-
[h264 @ 00000182bccfe640] >  >+ >  >  >+ >+ >  >+ >+ >  >  >  >+ >  >  >+ >  >| >- >  >- >  >  >  >| >- >- >  >- >+ >+
[h264 @ 00000182bccfe640] >+ >- >| >  >| >  >| >  >| >+ >+ >+ >+ >- >  >| >| >- >| >- >+ >+ >+ >  >+ >  >- >+ >+ >  >-
[h264 @ 00000182bccfe640] >| >- >  S  >- >  >- >+ >| >  >  >- >+ >  >+ >  >  >+ >| >- >| >+ >+ >- >  >  >+ >+ >+ >+ >|
```

Figure 9 Example of macroblock information

4.1.3 MV

The MV function visualizes the motion vectors for every frame of the video. Figure 10 shows one frame of a video, and many motion vectors can be seen in a man’s hand. The example indicates movement from a previous frame. On the other hand, there are fewer motion vectors in the motionless background.

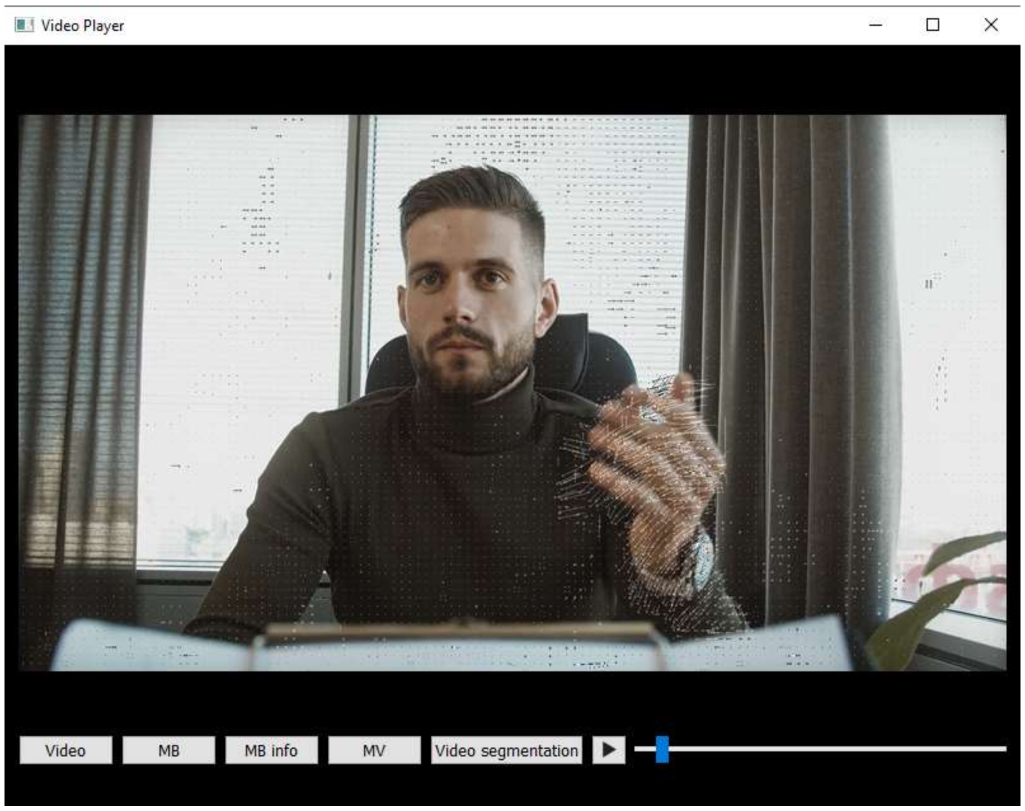


Figure 10 Example of motion vector visualization

4.1.4 Video segmentation

The Video segmentation function divides the video into frames and outputs a text file that aggregates the frame information. When the user enters a frame name, the frame number will be automatically added and saved. (Figure 11) For example, the number 1 is added to the first frame (sample.00001.jpg). This is also useful for dividing a video with visualized macro blocks and motion vectors into frames. Figure 12 shows the extracted frame information from the output text file for frames 1 through 5 that make up a GOP. It contains frame type and frame size, and the "coded_picture_number" indicates in which order each frame was coded. According to the text file, these five frames were encoded in the order 1, 5, 3, 2, 4.

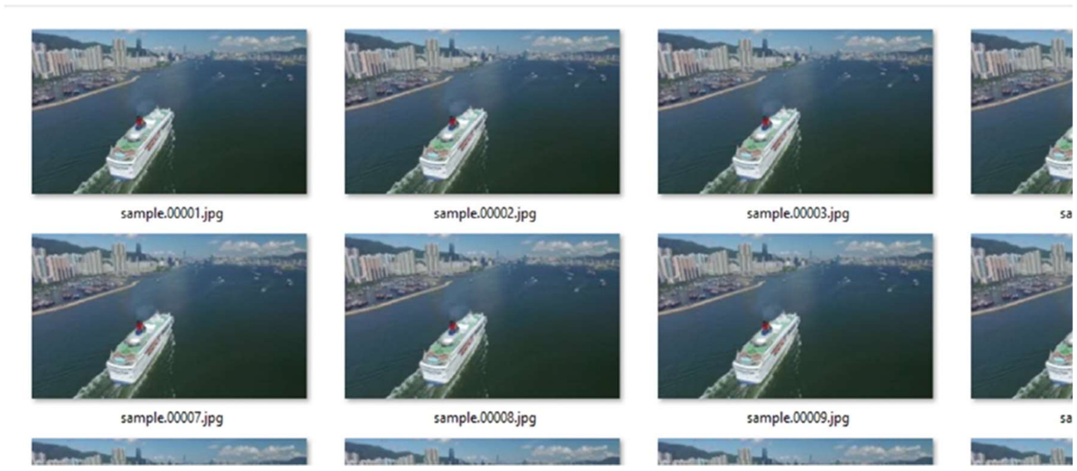
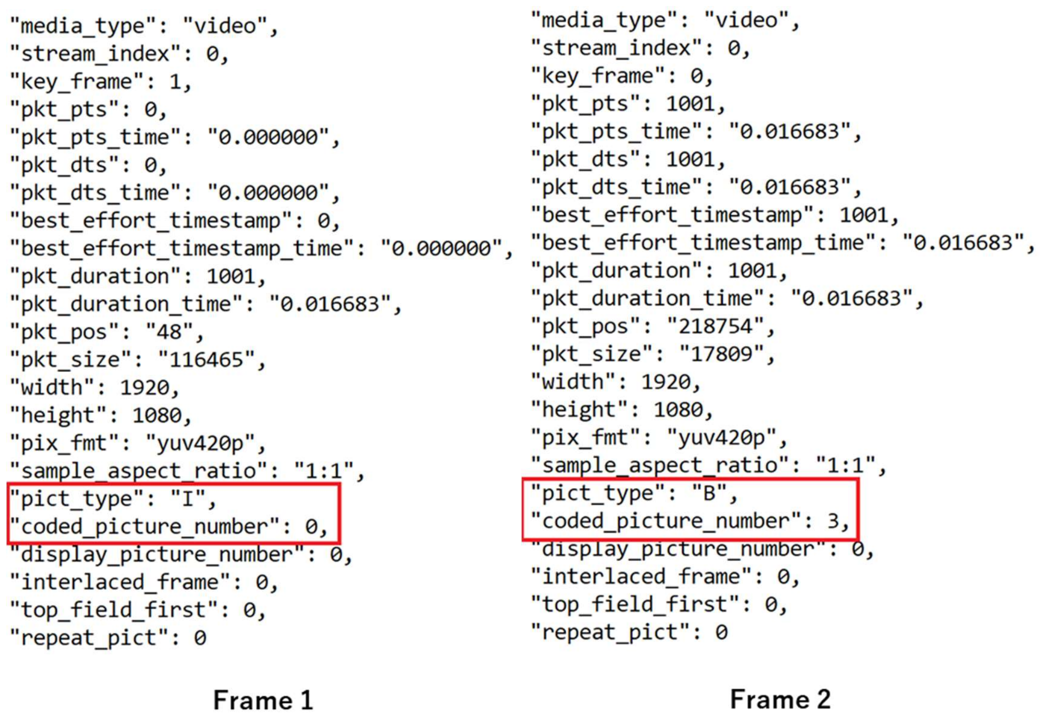


Figure 11 Example of video segmentation function 1

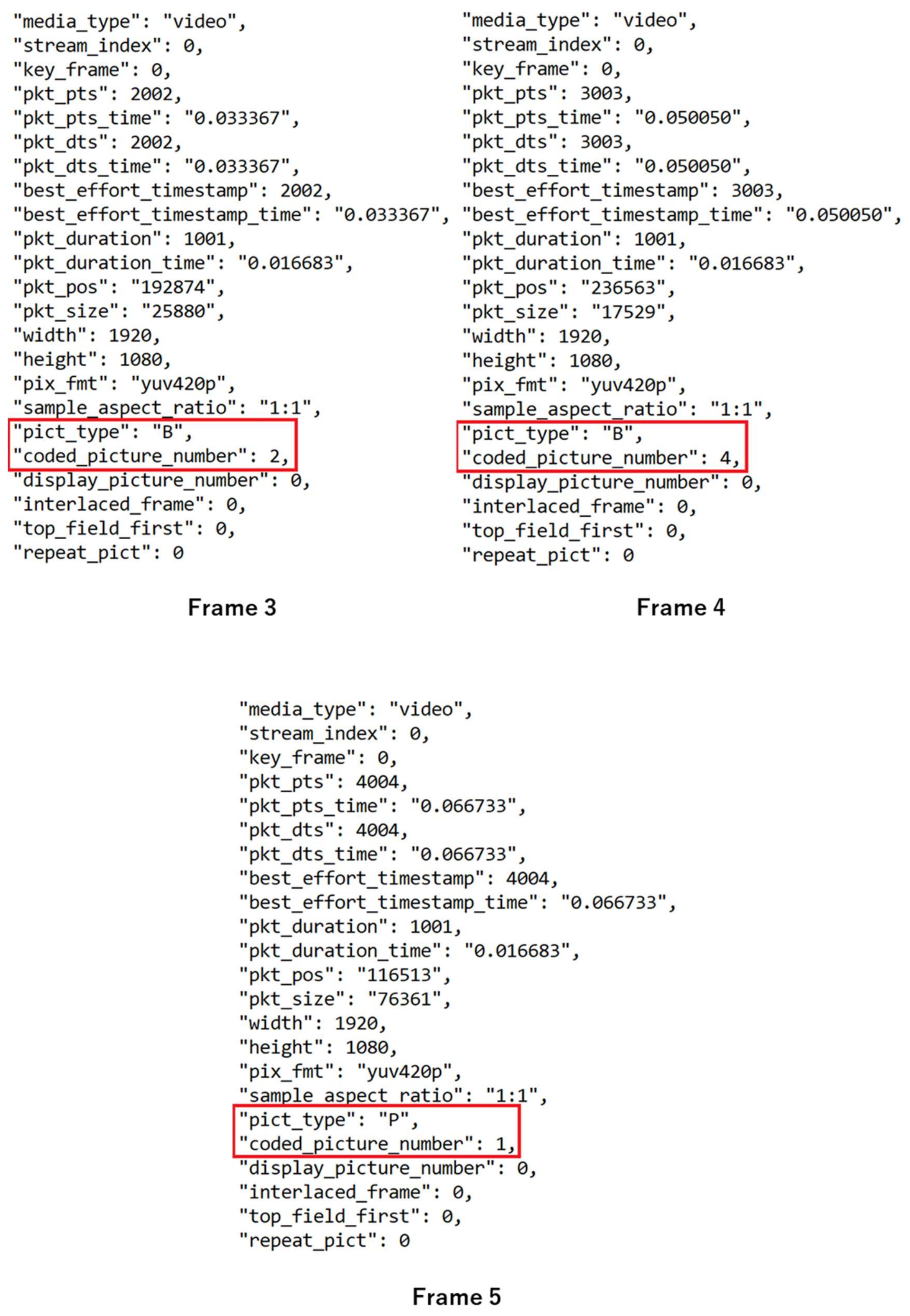


```
"media_type": "video",
"stream_index": 0,
"key_frame": 1,
"pkt_pts": 0,
"pkt_pts_time": "0.000000",
"pkt_dts": 0,
"pkt_dts_time": "0.000000",
"best_effort_timestamp": 0,
"best_effort_timestamp_time": "0.000000",
"pkt_duration": 1001,
"pkt_duration_time": "0.016683",
"pkt_pos": "48",
"pkt_size": "116465",
"width": 1920,
"height": 1080,
"pix_fmt": "yuv420p",
"sample_aspect_ratio": "1:1",
"pict_type": "I",
"coded_picture_number": 0,
"display_picture_number": 0,
"interlaced_frame": 0,
"top_field_first": 0,
"repeat_pict": 0
```

Frame 1

```
"media_type": "video",
"stream_index": 0,
"key_frame": 0,
"pkt_pts": 1001,
"pkt_pts_time": "0.016683",
"pkt_dts": 1001,
"pkt_dts_time": "0.016683",
"best_effort_timestamp": 1001,
"best_effort_timestamp_time": "0.016683",
"pkt_duration": 1001,
"pkt_duration_time": "0.016683",
"pkt_pos": "218754",
"pkt_size": "17809",
"width": 1920,
"height": 1080,
"pix_fmt": "yuv420p",
"sample_aspect_ratio": "1:1",
"pict_type": "B",
"coded_picture_number": 3,
"display_picture_number": 0,
"interlaced_frame": 0,
"top_field_first": 0,
"repeat_pict": 0
```

Frame 2



```
"media_type": "video",
"stream_index": 0,
"key_frame": 0,
"pkt_pts": 2002,
"pkt_pts_time": "0.033367",
"pkt_dts": 2002,
"pkt_dts_time": "0.033367",
"best_effort_timestamp": 2002,
"best_effort_timestamp_time": "0.033367",
"pkt_duration": 1001,
"pkt_duration_time": "0.016683",
"pkt_pos": "192874",
"pkt_size": "25880",
"width": 1920,
"height": 1080,
"pix_fmt": "yuv420p",
"sample_aspect_ratio": "1:1",
"pict_type": "B",
"coded_picture_number": 2,
"display_picture_number": 0,
"interlaced_frame": 0,
"top_field_first": 0,
"repeat_pict": 0
```

Frame 3

```
"media_type": "video",
"stream_index": 0,
"key_frame": 0,
"pkt_pts": 3003,
"pkt_pts_time": "0.050050",
"pkt_dts": 3003,
"pkt_dts_time": "0.050050",
"best_effort_timestamp": 3003,
"best_effort_timestamp_time": "0.050050",
"pkt_duration": 1001,
"pkt_duration_time": "0.016683",
"pkt_pos": "236563",
"pkt_size": "17529",
"width": 1920,
"height": 1080,
"pix_fmt": "yuv420p",
"sample_aspect_ratio": "1:1",
"pict_type": "B",
"coded_picture_number": 4,
"display_picture_number": 0,
"interlaced_frame": 0,
"top_field_first": 0,
"repeat_pict": 0
```

Frame 4

```
"media_type": "video",
"stream_index": 0,
"key_frame": 0,
"pkt_pts": 4004,
"pkt_pts_time": "0.066733",
"pkt_dts": 4004,
"pkt_dts_time": "0.066733",
"best_effort_timestamp": 4004,
"best_effort_timestamp_time": "0.066733",
"pkt_duration": 1001,
"pkt_duration_time": "0.016683",
"pkt_pos": "116513",
"pkt_size": "76361",
"width": 1920,
"height": 1080,
"pix_fmt": "yuv420p",
"sample_aspect_ratio": "1:1",
"pict_type": "P",
"coded_picture_number": 1,
"display_picture_number": 0,
"interlaced_frame": 0,
"top_field_first": 0,
"repeat_pict": 0
```

Frame 5

Figure12 Example of video segmentation function 2

4.2 Image Player

The "Image Player" has some useful functions for analyzing frames.

4.2.1 Load image

The Load image function displays the image and allows the user to zoom in and out on a specific area. Figure 13 shows the frame with the macroblock visualized. Figure 14 shows an enlargement of the hand part of the frame. This frame can be identified as frame B because macroblocks of fingers refer to past and future frames. Figure 15 shows the frame with the motion vectors visualized. This example shows that the hand has moved to the lower right corner from the past frame.

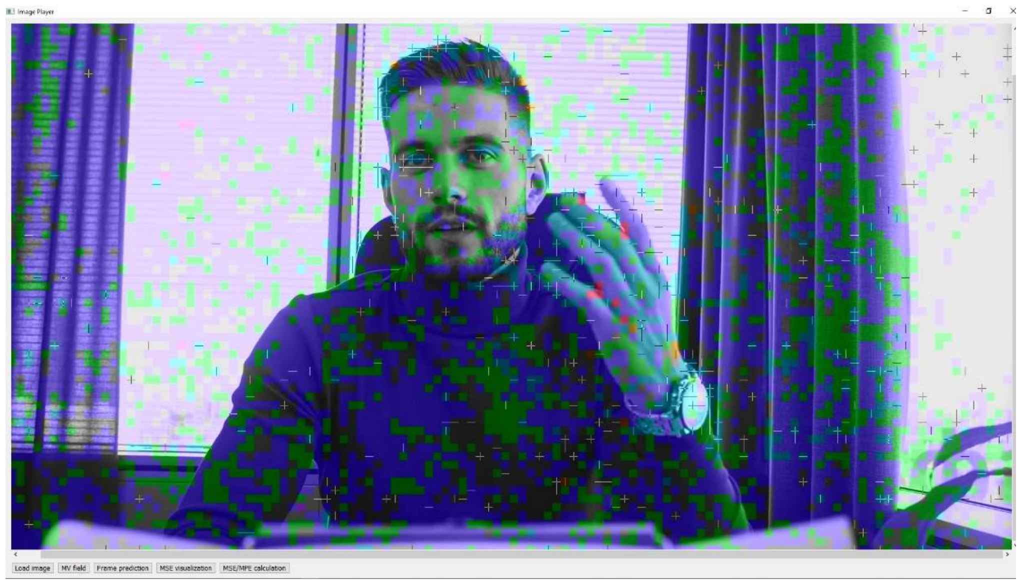


Figure13 Example of load image function 1

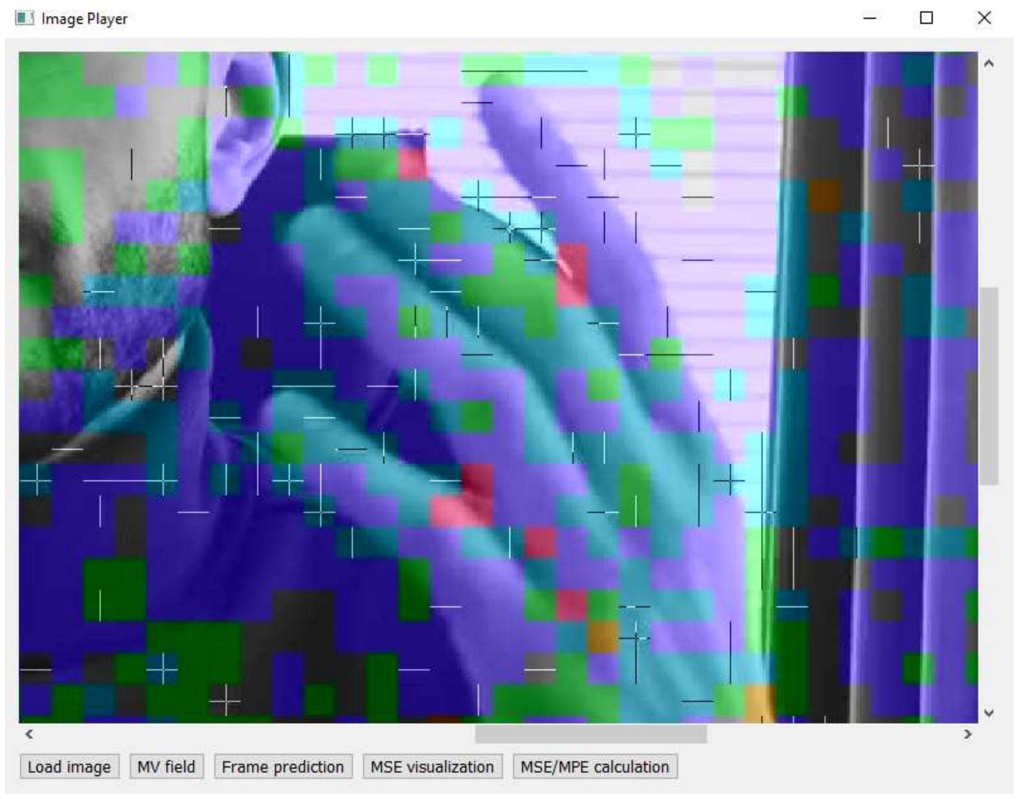


Figure14 Example of load image function 2

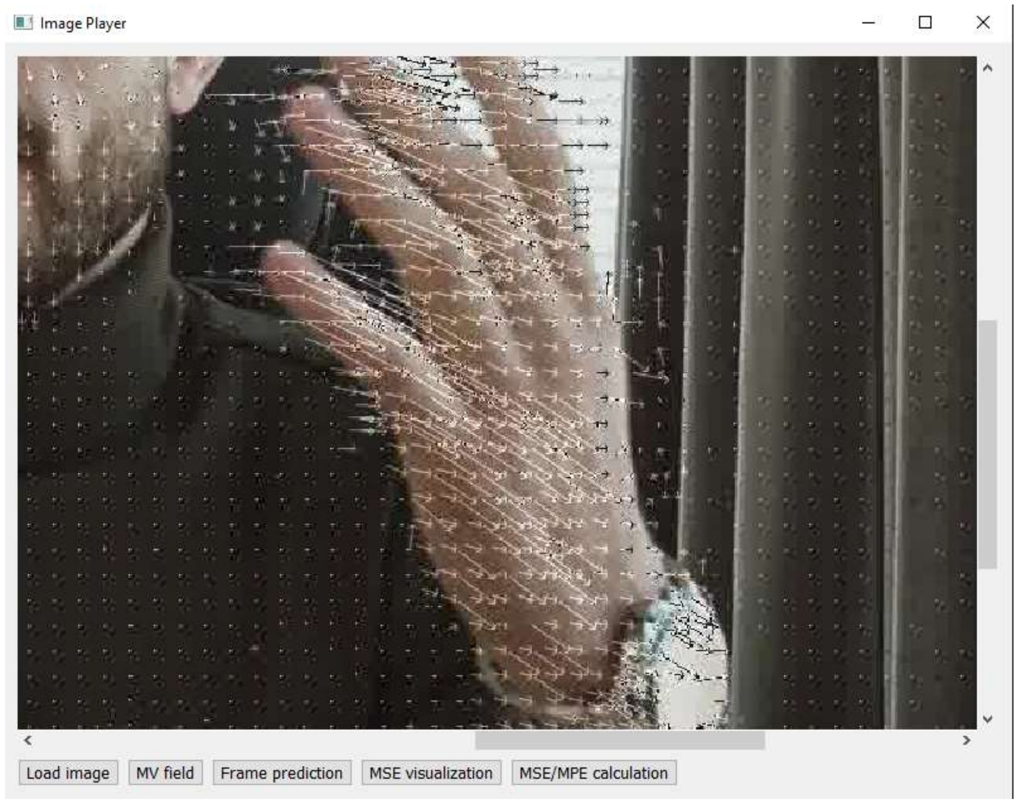


Figure15 Example of load image function 3

4.2.2 MV field

The MV Field function calculates the gradient of adjacent pixel data in a frame and displays its magnitude and direction in a vector field. This function visualizes the coding mechanism by intra-frame prediction, which shows the correlation of pixel values in a particular space within a frame. It can also zoom in on an arbitrary area. Figure 16 and Figure 17 show an example of a motion vector field. The motion from the reference pixel is shown as a red vector, which allows the user to obtain a motion vector field with high spatial resolution.

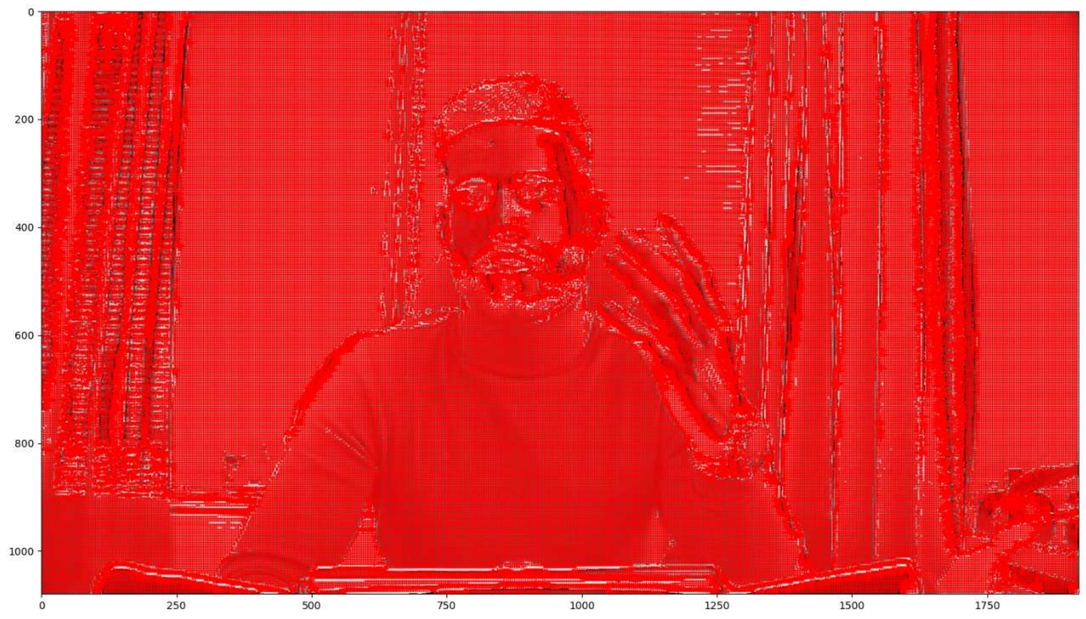


Figure16 Example of motion vector field function 1

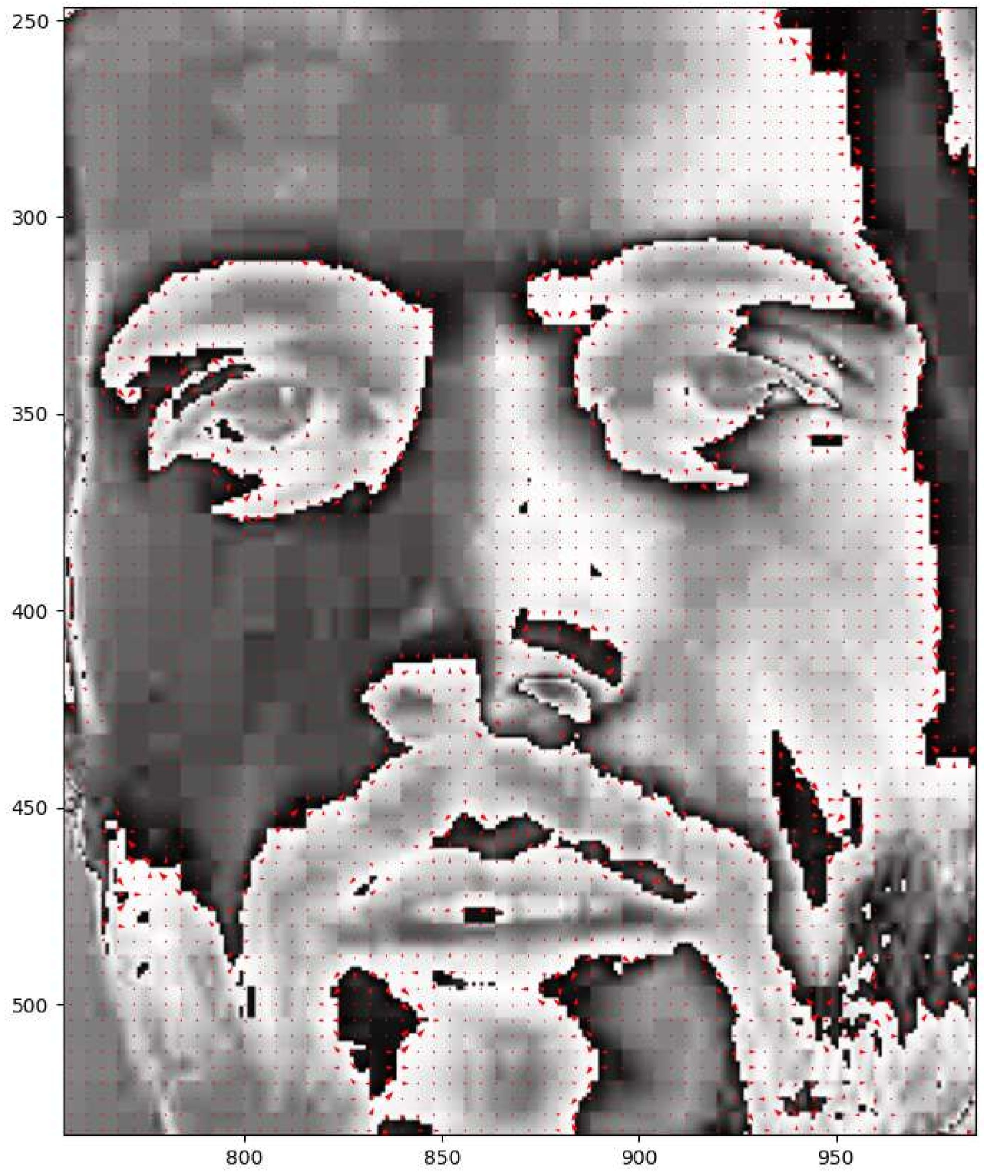
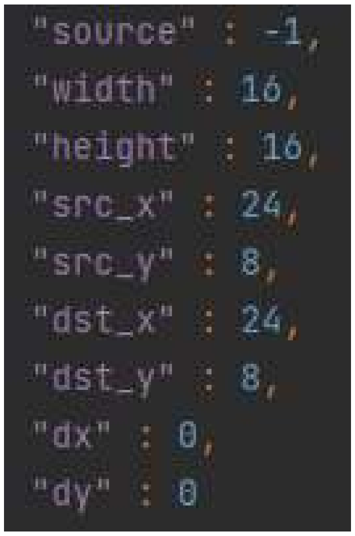


Figure17 Example of motion vector field function 2

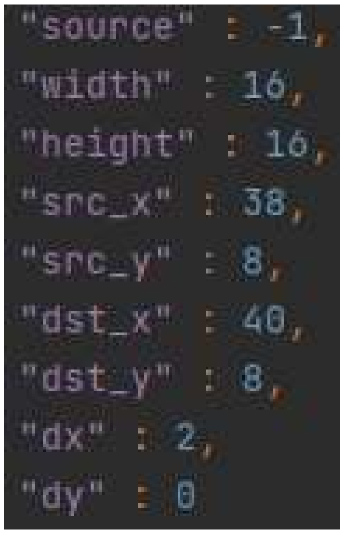
4.2.3 Frame prediction

The Frame Prediction feature allows users to generate a predictive image of the target frame from a reference frame. The user can analyze the differences found between the vector-encoded approximate frame and the original frame. However, this function requires the user to extract the motion vector data between the target frame and the reference frame using an application called [2]MV-Tractus. [31]

Figure 18 is an example of the frame-to-frame motion vector data output by MV-Tractus. The motion vector data on the left shows that the 16x16 macroblock region centered at (x,y)=(24,8) has not changed. On the other hand, in another motion vector data, we can see that the 16x16 macroblock region centered at (x,y)=(38,8) in the reference frame has changed to (x,y)=(40,8) in the source frame. For "Source," -1 indicates a move from a past frame, and 1 indicates a move from a future frame.



```
"source" : -1,
"width" : 16,
"height" : 16,
"src_x" : 24,
"src_y" : 8,
"dst_x" : 24,
"dst_y" : 8,
"dx" : 0,
"dy" : 0
```



```
"source" : -1,
"width" : 16,
"height" : 16,
"src_x" : 38,
"src_y" : 8,
"dst_x" : 40,
"dst_y" : 8,
"dx" : 2,
"dy" : 0
```

Figure18 Motion vector data

The prediction frame is generated by moving all the macroblocks in the reference frame based on the motion vector coordinates described above. Since the prediction frame is not adjusted for residual errors, features that are not present in the original image may appear. Figure 19 shows the original frame, and Figure 20 shows the predicted frame.

[2] https://github.com/jishnujayakumar/MV-Tractus

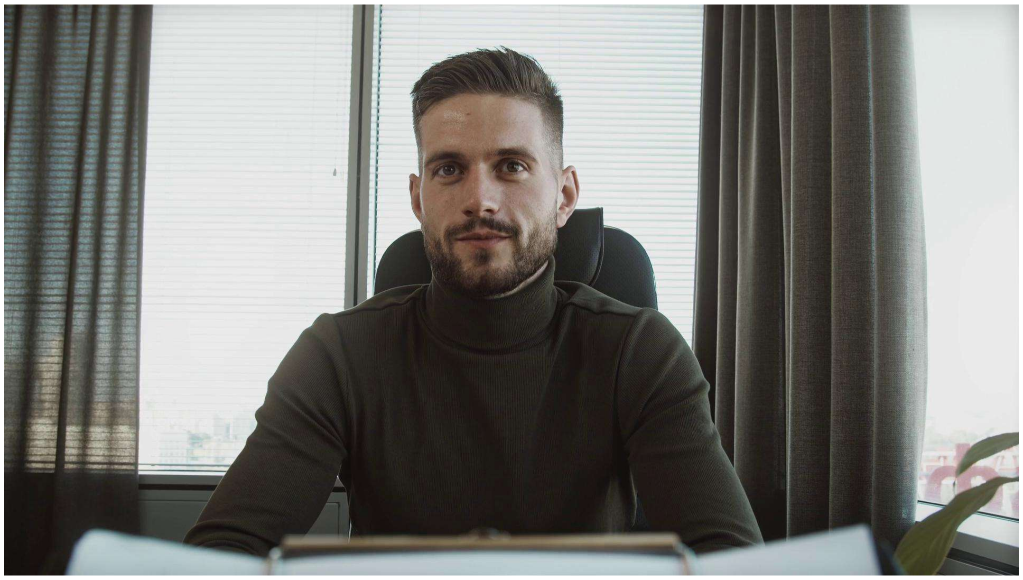


Figure19 Example of original frame

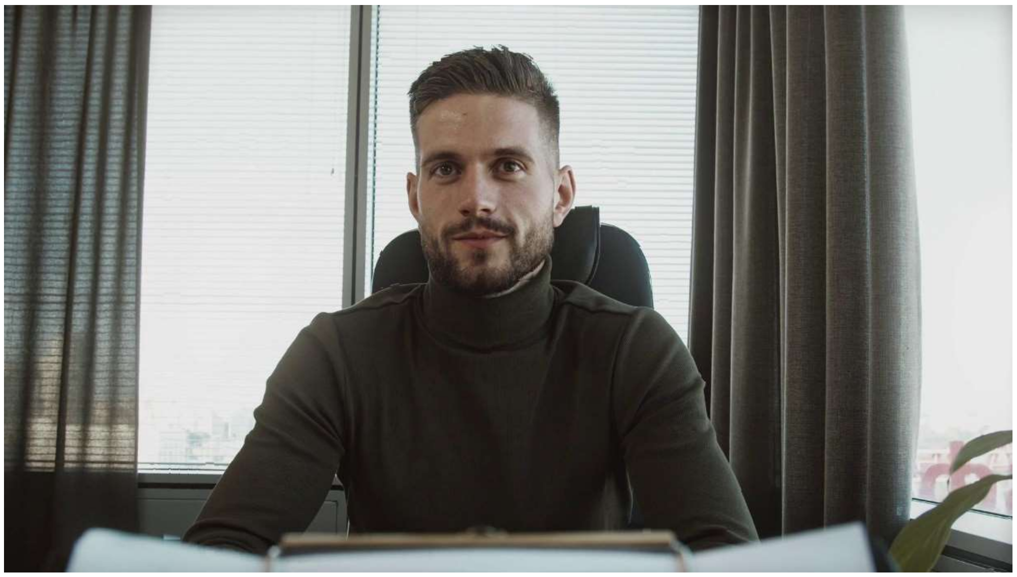


Figure20 Example of predicted frame

4.2.4 MSE visualization

The MSE visualization function numerically evaluates the similarity between the predicted frame and the original image by comparing two frames and calculating the MSE of the pixel values of the macroblocks at the same coordinates.

The MSE is visualized on the prediction frame according to its value. The smaller the MSE, the closer the image is to the original. Figure 21 shows an example of visualizing the MSE by comparing Figure 19 with Figure 20: blue if the MSE is 0, green if the MSE is between 1 and 500, and red otherwise. The red areas have discontinuities in the pixel values for reasons that are not immediately clear.

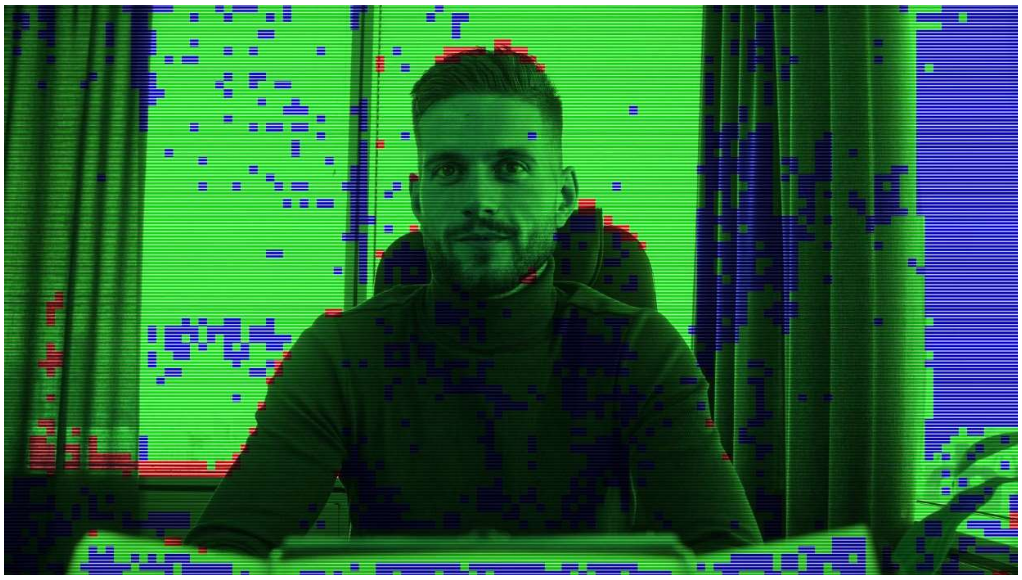


Figure21 Example of MSE visualization function

4.2.5 MSE/MPE calculation

The MSE/MPE calculation function outputs three types of information to a text file: macroblock coding information, MSE, and maximum pixel error (MPE). The maximum pixel error is the value obtained by subtracting the minimum pixel value from the maximum pixel value; for a 16x16 macroblock, the MPE is calculated by comparing 256 pixel values.

5 Experimental results and discussion

The previous chapter described the functions of the prototype tool. This chapter presents the procedure for analyzing frames using the prototype tool.

5.1 Target Frame Selection

First, the video was divided into frames using the video division function, and the target frame was selected from the frame information. (Figure 22)

Next, macroblocks were visualized in the same video using the MB and MB information functions, and the coding information of the macroblocks was output as a text file. The video segmentation function was used to divide the video in which macroblocks were visualized into frames. Figure 23 shows the target frame with the macroblock visualized.

Finally, the information about the target frame was extracted from the text file containing the macro block's coding information. (Figure 24)

Figure 22 Example of target frame

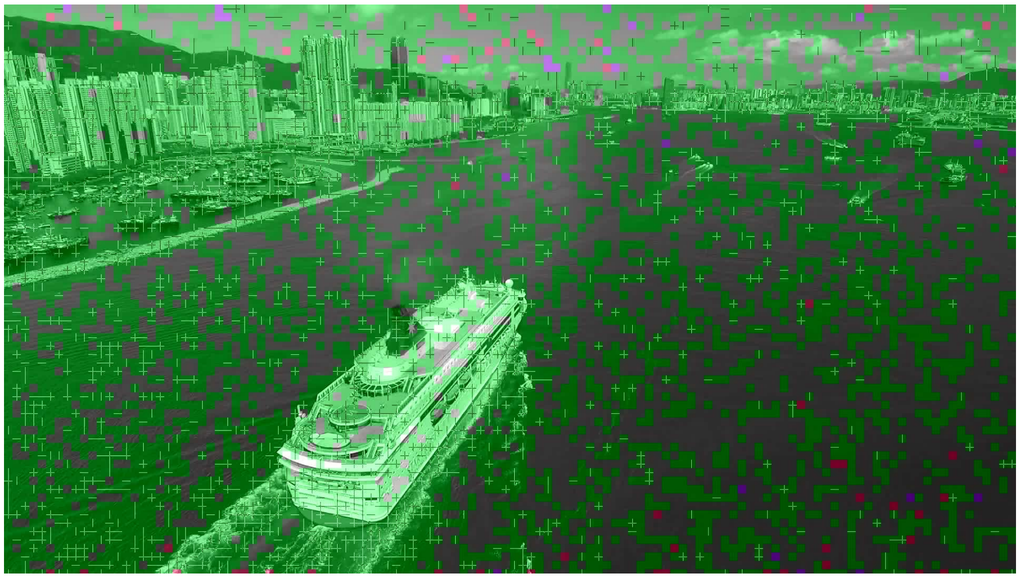
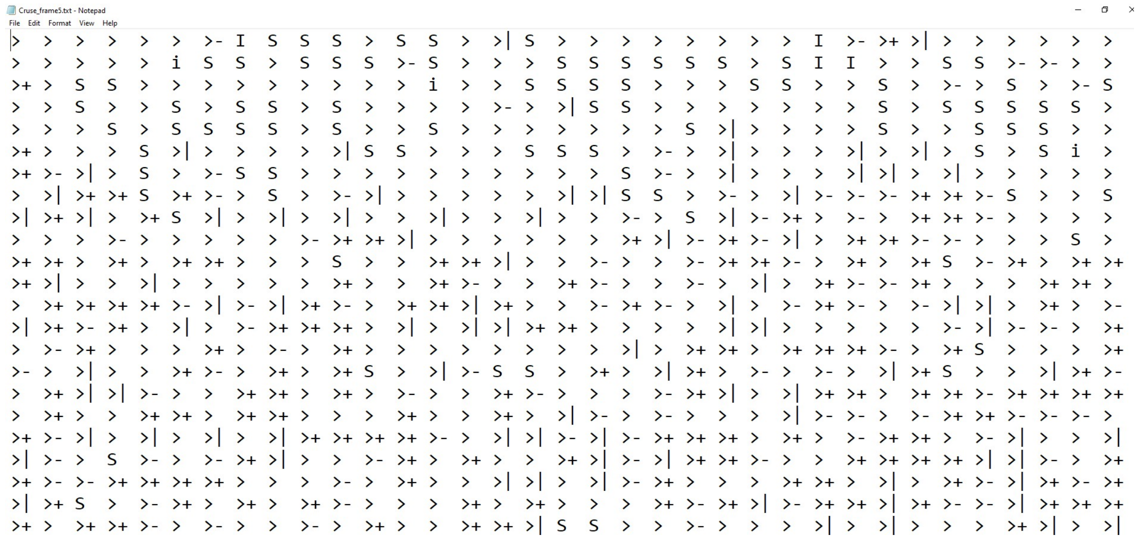


Figure 23 Example of target frame with the macroblock visualized.



```
>  >  >  >  >  >  >- I  S  S  S  >  S  S  >  >| S  >  >  >  >  >  >  >  >  I  >- >+ >| >  >  >  >  >  >
>  >  >  >  >  i  S  S  >  S  S  S  >- S  >  >  >  S  S  S  S  S  S  >  S  I  I  >  >  S  S  >- >- >  >
>+ >  S  S  >  >  >  >  >  >  >  >  >  i  >  >  S  S  S  S  S  >  >  >  S  S  >  >  S  >  >- >  S  >  >- S
>  >  S  >  >  S  >  S  S  >  S  >  >  >  >  >- >  >| S  S  S  >  >  >  >  >  >  >  S  >  S  S  S  S  S  >
>  >  >  S  >  S  S  S  S  >  S  >  >  S  >  >  >  >  >  >  >  S  >| >  >  >  >  S  >  >  S  S  S  >  >
>+ >  >  >  S  >| >  >  >  >  >| S  S  >  >  >  S  S  S  >  >- >  >| >  >  >  >| >  >| >  S  >  S  i  >
>+ >- >| >  S  >  >- S  S  >  >  >  >  >  >  >  >  >  >  S  >- >  >| >  >  >  >| >| >  >| >  >  >  >  >
>  >| >+ >+ S  >+ >- >  S  >  >- >| >  >  >  >  >  >| >| S  S  >  >- >  >| >- >- >- >+ >+ >- S  >  >  S
>| >+ >| >  >+ S  >| >  >| >  >| >  >  >| >  >  >| >  >  >- >  S  >| >- >+ >  >- >  >+ >+ >- >  >  >  >
>  >  >  >- >  >  >  >  >  >- >+ >+ >| >  >  >  >  >  >  >+ >| >- >+ >- >| >  >+ >+ >- >- >  >  >  S  >
>+ >+ >  >+ >  >+ >+ >  >  >  S  >  >  >+ >+ >| >  >  >- >  >  >- >+ >+ >- >  >+ >  >+ S  >- >+ >  >+ >+
>+ >| >  >  >| >  >  >  >  >  >+ >  >  >+ >- >  >  >+ >- >  >  >- >  >| >  >+ >- >- >+ >  >  >  >+ >+ >
>  >+ >+ >+ >+ >- >| >- >| >+ >- >  >+ >+ >| >+ >  >  >- >+ >- >  >| >  >- >+ >- >  >- >| >| >  >+ >  >-
>| >+ >- >+ >  >| >  >- >+ >+ >+ >  >| >  >| >| >+ >+ >  >  >  >  >| >| >  >  >  >  >  >- >| >- >- >  >+
>  >- >+ >  >  >  >+ >  >- >  >+ >  >  >  >  >  >  >  >  >| >  >+ >+ >  >+ >+ >+ >- >  >+ S  >  >  >  >+
>- >  >| >  >  >+ >- >  >+ >  >+ S  >  >| >- S  S  >  >+ >  >| >+ >  >- >  >- >  >| >+ S  >  >  >| >+ >-
>  >+ >| >| >- >  >  >+ >+ >  >+ >  >- >  >  >+ >- >  >  >  >- >+ >| >  >| >+ >+ >  >+ >+ >- >+ >+ >+ >+
>  >+ >  >  >+ >+ >  >+ >+ >  >  >  >+ >  >  >+ >  >| >- >  >- >  >  >  >| >- >- >  >- >+ >+ >- >- >- >
>+ >- >| >  >| >  >| >  >| >+ >+ >+ >+ >- >  >| >| >- >| >- >+ >+ >+ >  >+ >  >- >+ >+ >  >- >| >  >  >|
>| >- >  S  >- >  >- >+ >| >  >  >- >+ >  >+ >  >  >+ >| >- >| >+ >+ >- >  >  >+ >+ >+ >+ >| >| >- >  >+
>+ >- >- >+ >+ >+ >+ >  >  >  >- >  >+ >  >  >| >| >  >| >- >+ >  >  >  >+ >+ >  >| >  >+ >- >| >+ >- >+
>| >+ S  >  >- >+ >  >+ >  >+ >- >  >  >  >+ >  >+ >  >  >  >+ >- >+ >| >- >+ >+ >| >+ >- >- >| >+ >+ >+
>+ >  >+ >+ >- >  >- >  >  >- >  >+ >  >  >+ >+ >| S  S  >  >  >- >  >  >  >| >  >| >  >  >  >+ >| >  >|
```

Figure 24 Example of coding information of the target frame

5.2 MSE visualization

After extracting the motion vector data, the Frame prediction function is used to generate the target frame's predicted image. Then, to get the difference between the original frame and the predicted image, the MSE visualization function checks where the difference in pixel values between the two frames is large. Figure 25 shows the area where the MSE is 501 or higher in red. Figure 26 shows the area where the MSE is 1001 or higher in red. The two frames show that the predicted frame differs from the original frame in the background buildings, the small ship, the ferry hull, and near the waves at the hull's rear.

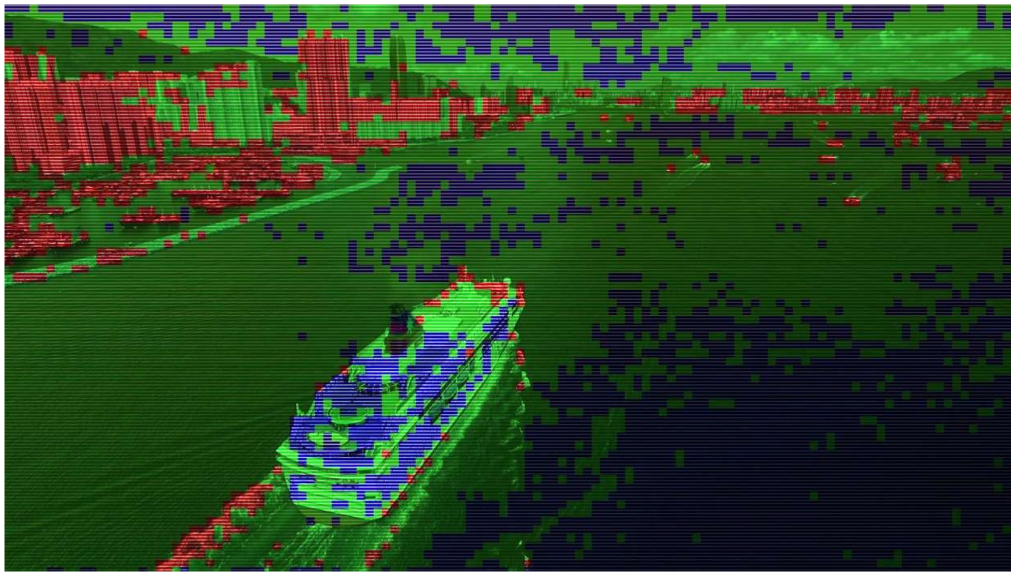


Figure 25 Example of MSE visualization 1

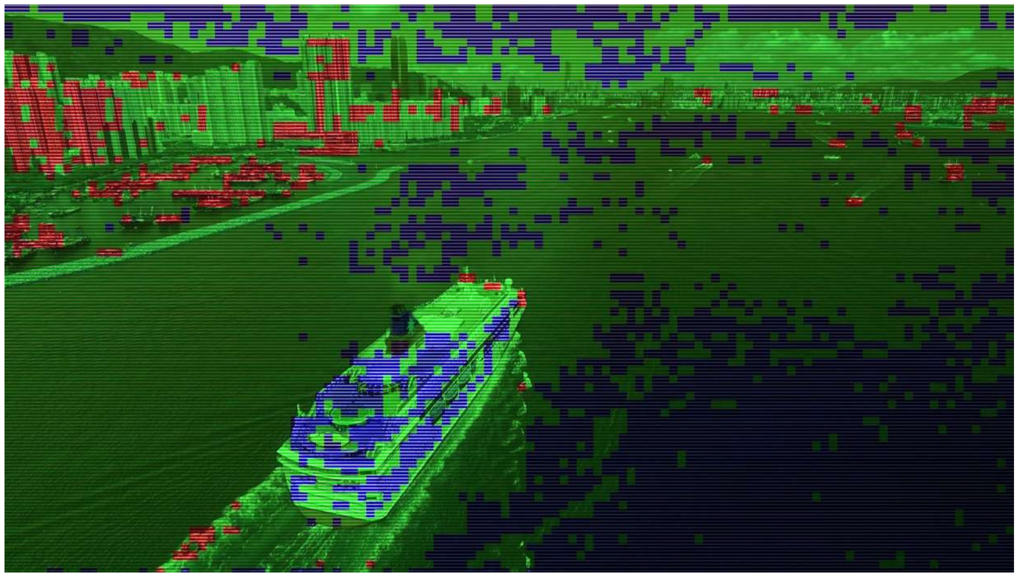


Figure 26 Example of MSE visualization 2

5.3 Analysis of MSE and MPE

The information of MSE and MPE between the reference frame and the prediction frame was output, and the relationship between them was analyzed.

The MSE and MPE values are expected to be large in areas where pixel value discontinuities exist, and we analyzed how the macroblocks in those areas are coded. Figure 27 and Figure 28 are a cumulative scatter plot that classifies MSE and MPE, the difference between the target frame and its predicted frame, based on the coding information of the macroblock. The two scatter plots have the number of elements on the vertical axis and MSE or MPE on the horizontal axis. From these scatter plots, we can see that the number of macroblocks encoded by interframe prediction with sub-blocks is less than that of intra-frame prediction or standard interframe prediction, but the MSE and MPE are large.

Figure 23 shows that many inter-frame predictions using sub-blocks are used in areas where the MSE is large. Therefore, we can assume that the discrepancy between the pixel values and the predicted image occurred because a more precise inter-frame prediction was used in the original frame.

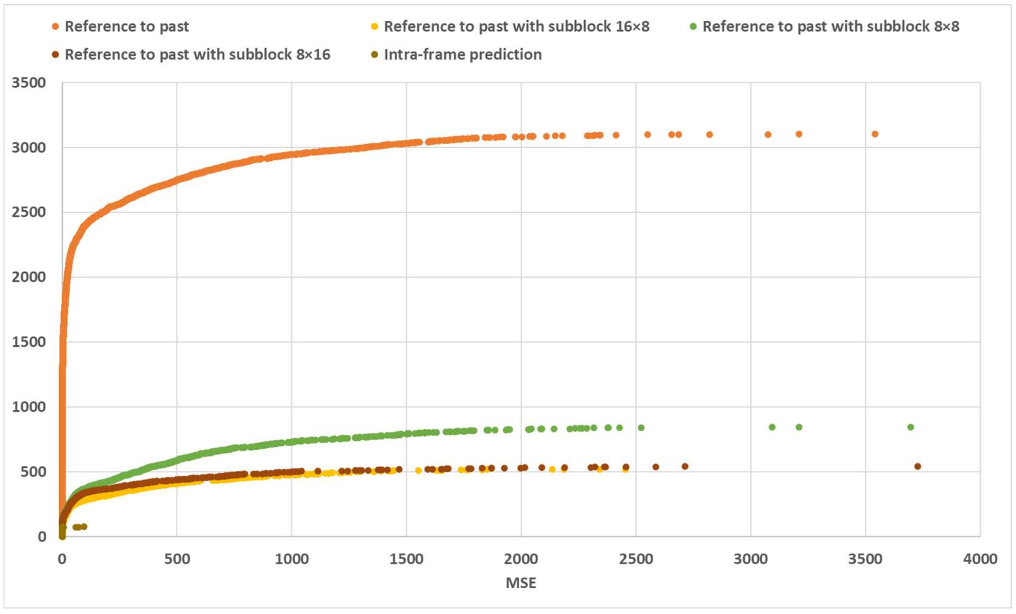


Figure 27 Cumulative scatter plot of MSE classified by macroblock symbols.

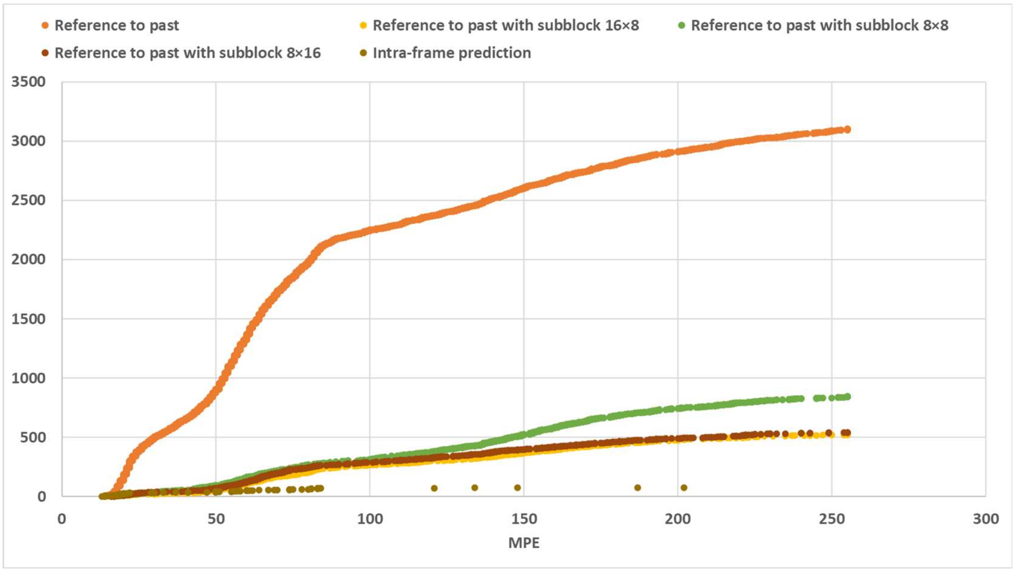


Figure 28 Cumulative scatter plot of MPE classified by macroblock symbols.

6 Deepfake analysis

Deepfake videos may produce inconsistent points in the frame.[32] This chapter examines the ability to identify tampered areas in an image when analyzing a Deepfake video using a prototype tool. Using [3]DeepFaceLab, a software for creating Deepfake, we created a Deepfake video and split it into frames. The analysis procedures are the same as those presented in the previous chapter.

Comparing the frame structure of the original video and the deep fake video, the total number of frames was the same, but there were differences in the B-frames at the first and last GOP. (Table 2, 3)

	Frame 1	Frame 2	Frame 3	Frame 4	Frame5
Original	I-frame	B-frame	B-frame	P-frame	-
Deepfake	I-frame	B-frame	B-frame	B-frame	P-frame

Table 2 Comparison of the first GOP.

	Frame 459	Frame 460	Frame 461	Frame 462	Frame463
Original	-	-	P-frame	B-frame	P-frame
Deepfake	P-frame	B-frame	B-frame	B-frame	P-frame

Table 3 Comparison of the last GOP

The following shows the original frame (Figure 29), the predicted frame (Figure 30), the original frame with macroblock visualization (Figure 31), and the frame with MSE visualization (Figure 32). Figure 32 visualizes the areas with MSEs of 501 or higher in red.

[3] https://github.com/iperov/DeepFaceLab

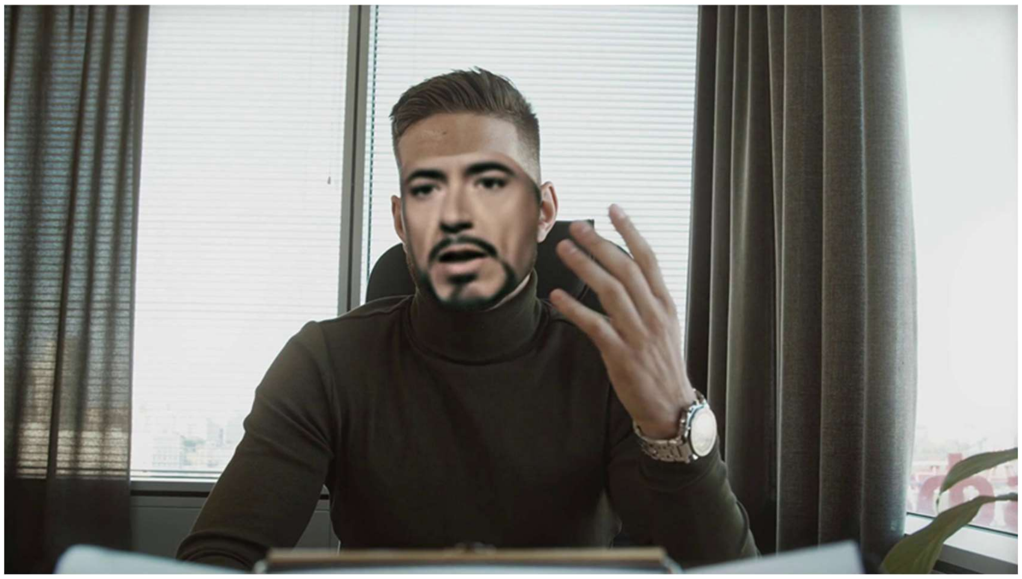


Figure 29 Example of original frame (Deepfake)

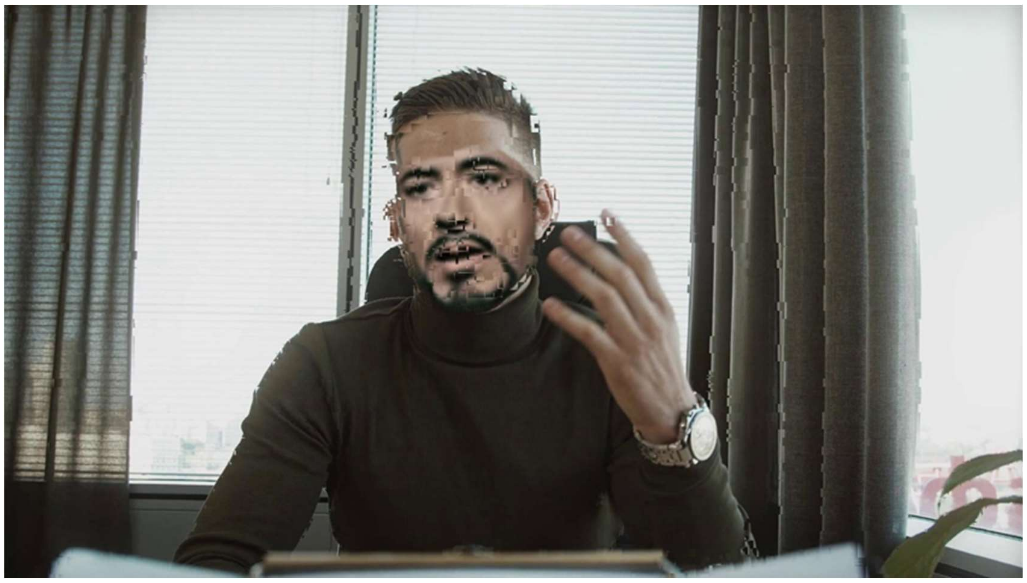


Figure 30 Example of predicted frame (Deepfake)

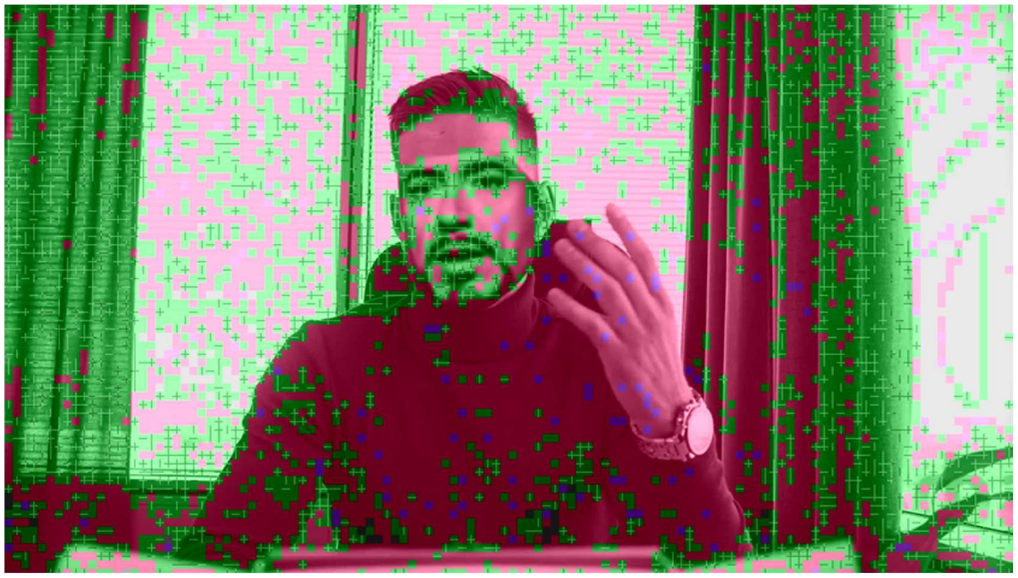


Figure 31 Example of macroblock visualization (Deepfake)

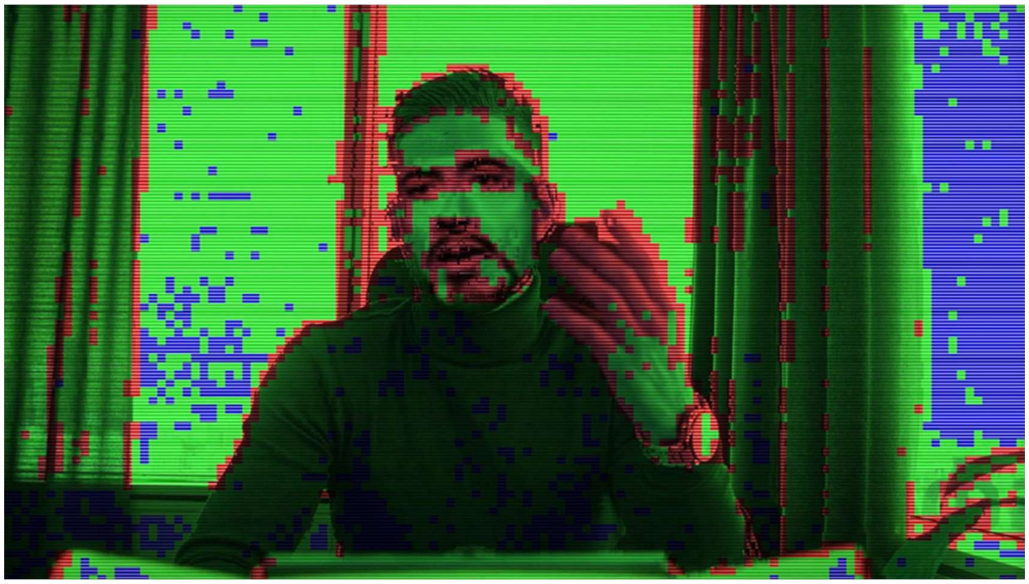


Figure 32 Example of MSE visualization (Deepfake)

Interestingly, while the number of elements in the region predicted using sub-blocks of past frames was 1773, the number of macroblocks using intra-frame prediction was 4036, accounting for half of the macroblocks in the frame. Compared to Fig. 27, the MSE is exceptionally large for all macroblock coding, and the maximum MSE and MPE of the macroblock using intra-frame prediction is higher than the other macroblocks, suggesting that some discontinuity occurs in the area using intra-frame. Since this original frame is a P-frame, it is unnatural that intra-frames are used for most of the male subject's face and body. (Figure 33, 34) In this analysis, the anomaly was inferred from the unnatural bias of the frame's coding information.

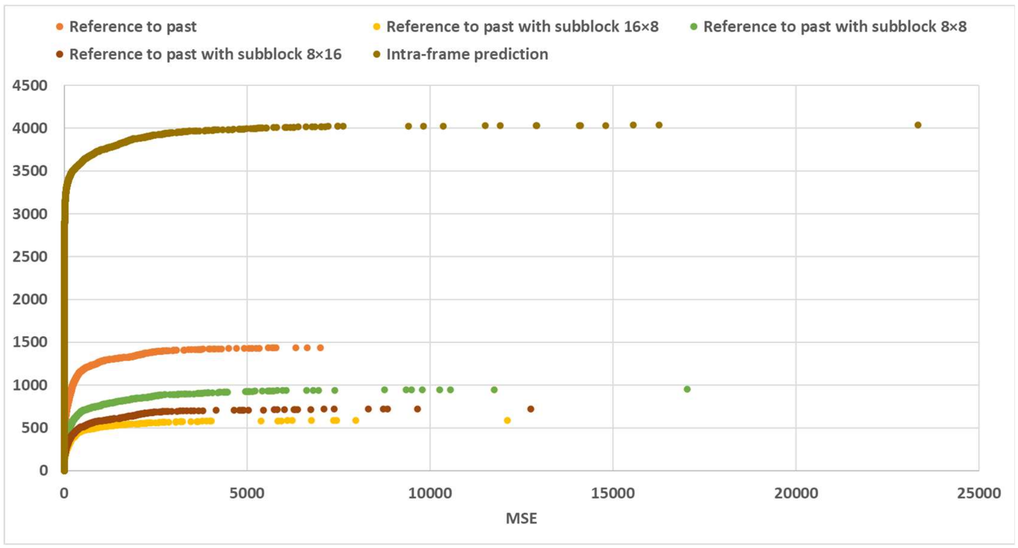


Figure 33 Cumulative scatter plot of MSE classified by macroblock symbols (Deepfake)

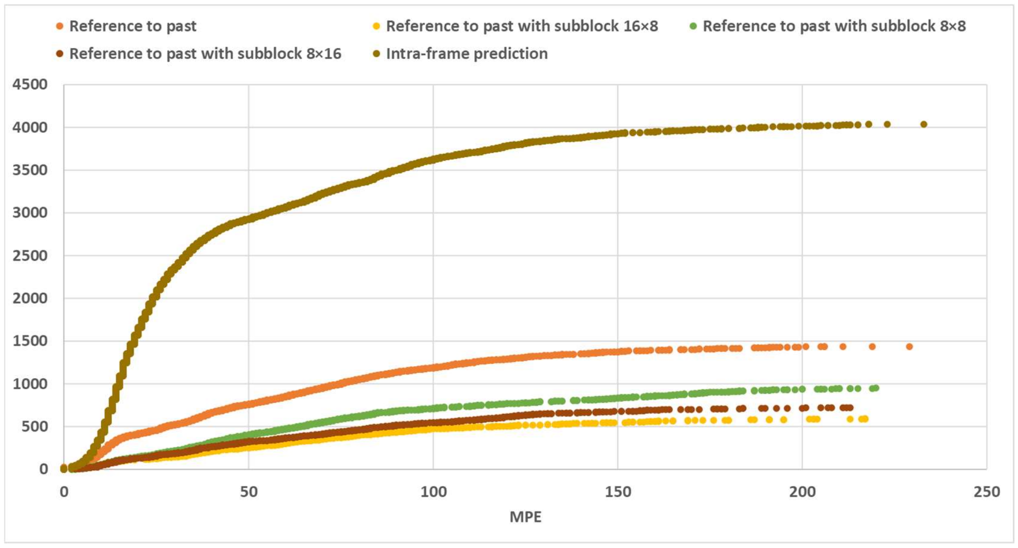


Figure 34 Cumulative scatter plot of MPE classified by macroblock symbols (Deepfake)

The important point in this chapter is that the data structure of the original video and the Deepfake video is different. If an adversary can characterize the source video data structure, he/she can mimic the source video data structure. It has significant implications for source codec identification, such as provenance analysis. Similarly, if the Deepfake video can be encoded with the same structure as the original video, it will be harder to detect the modified video.

For future work, I will consider using my tool to analyze the structure of the source video and generate a Deepfake that is encoded in a way that is consistent with the source video. If successful, this technique may be used as an anti-forensic tool by bypassing the detection of video structure changes.

7 Conclusion

This paper proposed a new environment for analyzing how a particular area in a video was coded. Video forensics experts will be able to analyze and explain whether the features in a video are authentic or contain compression artifacts. PyQt was chosen as the development platform because of its advanced features, ease of programming, and easily modify it. The prototype tool developed provides different types of data (still images, video, macroblock processing, motion vectors), automatic creation of text files, and another function. However, it currently relies on other software to extract motion vectors between frames and cannot automatically analyze videos or images.

Gloe et al. showed that the structure of video file containers differed among media devices such as digital cameras and cell phones.[33] Luliani et al. also showed that the video container structure changes when any manipulation is applied to the original video in a video file container such as MP4.[34]

If the prototype tool can visualize the structure of video file containers, it could be used for video provenance analysis and Deepfake detection.

References

[1] Zeng K, Zhao T, Rehman A, Wang Z. Characterizing perceptual artifacts in compressed video streams. In: Rogowitz BE, Pappas TN, de Ridder H, editors. San Francisco, California, USA; 2014

[2] Milani S, Fontani M, Bestagini P, Barni M, Piva A, Tagliasacchi M, et al. An overview on video forensics. APSIPA trans signal inf process. 2012 Aug;1:e2.

[3] Jiang X, Wang W, Sun T, Shi YQ, Wang S. Detection of Double Compression in MPEG-4 Videos Based on Markov Statistics. IEEE Signal Process Lett. 2013 May;20(5):447–50.

[4] Vazquez-Padin D, Fontani M, Bianchi T, Comesana P, Piva A, Barni M. Detection of video double encoding with GOP size estimation. In: 2012 IEEE International Workshop on Information Forensics and Security (WIFS) [Internet]. Costa Adeje - Tenerife, Spain: IEEE; 2012 [cited 2021 Apr 22]. p. 151–6. Available from: http://ieeexplore.ieee.org/document/6412641/

[5] Su Y, Xu J. Detection of Double-Compression in MPEG-2 Videos. In: 2010 2nd International Workshop on Intelligent Systems and Applications [Internet]. Wuhan, China: IEEE; 2010 [cited 2021 Apr 22]. p. 1–4. Available from: http://ieeexplore.ieee.org/document/5473474/

[6] Wang W, Farid H. Exposing digital forgeries in video by detecting double MPEG compression. In: Proceeding of the 8th workshop on Multimedia and security - MM&Sec '06 [Internet]. Geneva, Switzerland: ACM Press; 2006 [cited 2021 Apr 22]. p. 37. Available from: http://portal.acm.org/citation.cfm?doid=1161366.1161375

[7] Stamm MC, Lin WS, Liu KJR. Temporal Forensics and Anti-Forensics for Motion Compensated Video. IEEE TransInformForensic Secur. 2012 Aug;7(4):1315–29.

[8] Kobayashi M, Okabe T, Sato Y. Detecting Forgery From Static-Scene Video Based on Inconsistency in Noise Level Functions. IEEE TransInformForensic Secur. 2010 Dec;5(4):883–92.

[9] Yan Zhou, Fan-Zhi Zeng, Guang-Fa Yang. The research for tamper forensics on MPEG-2 video based on compressed sensing. In: 2012 International Conference on Machine Learning and Cybernetics [Internet]. Xian, Shaanxi, China: IEEE; 2012 [cited 2021 Apr 22]. p. 1080–4. Available from: http://ieeexplore.ieee.org/document/6359505/

[10] Dong Q, Yang G, Zhu N. A MCEA based passive forensics scheme for detecting frame-based video tampering. Digital Investigation. 2012 Nov;9(2):151–9.

[11] Su Y, Zhang J, Liu J. Exposing Digital Video Forgery by Detecting Motion-Compensated Edge Artifact. :4.

[12] Jerian M, Paolino S, Cervelli F, Carrato S, Mattei A, Garofano L. A forensic image processing environment for investigation of surveillance video. Forensic Science International. 2007 Apr;167(2–3):207–12.

[13] Al-Ameen Z, Sulong GB, Johar G. Computer Forensics and Image Deblurring: An Inclusive Investigation. 2013;7.

[14] Al-Ameen Z, Al-Atroshi C. Modern Visibility Enhancement and Tampering Detection Tools of Digital Image Forensics: A Laconic Review. Journal of Arts and Imaging Science. 2017;4(1):6.

[15] Lei X, Jiang X, Wang C. Design and Implementation of a Real-Time Video Stream Analysis System Based on FFMPEG. In: 2013 Fourth World Congress on Software Engineering [Internet]. Hong Kong, China: IEEE; 2013 [cited 2021 Apr 22]. p. 212–6. Available from: http://ieeexplore.ieee.org/document/6754288/

[16] Ryu C, Lee D, Jang M, Kim C, Seo E. Extensible Video Processing Framework in Apache Hadoop. In: 2013 IEEE 5th International Conference on Cloud Computing Technology and Science [Internet]. Bristol, United Kingdom: IEEE; 2013 [cited 2021 Apr 22]. p. 305–10. Available from: http://ieeexplore.ieee.org/document/6735441/

[17] Zeng H, Shi L, Zhang Z. Research and Implementation of Video Codec Based on FFmpeg. :5.

[18] https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/PCAST/pcast_forensic_science_report_final.pdf

[19] https://www.justice.gov/olp/page/file/1352496/download

[20] Kalva H. The H.264 Video Coding Standard. 2006;5.

[21] Sorell M. Video provenance by motion vector analysis: A feasibility study. In: 2012 5th International Symposium on Communications, Control and Signal Processing [Internet]. Roma, Italy: IEEE; 2012

[22] Puri A, Chen X, Luthra A. Video coding using the H.264/MPEG-4 AVC compression standard. Signal Processing: Image Communication. 2004 Oct;19(9):793–849.

[23] Fitzpatrick, Martin. Create GUI Applications with Python & Qt5 (PyQt5 Edition): The hands-on guide to making apps with Python

[24] Moore, Alan D.. Mastering GUI Programming with Python: Develop impressive cross-platform GUI applications with PyQt

[25] Riselvato, John. FFmpeg Quick Reference of 100+ Scripts for Video, Audio and Streaming

[26] Weed, Chris. Introduction to FFmpeg: Encode Video for the Web and Mobile Devices

[27] https://ffmpeg.org/documentation.html

[28] https://github.com/Yoshihisa90/Research_Project

[29] http://trac.ffmpeg.org/wiki/Debug/MacroblocksAndMotionVectors

[30] https://github.com/FFmpeg/FFmpeg/blob/918de766f545008cb1ab3d62febe71fa064f8ca7/libavcodec/mpegutils.c#L196

[31] Jishnu P, & Singh, Praneet. (2018, October 21). MV-Tractus: A simple tool to extract motion vectors from H264 encoded video sources (Version 2.0). Zenodo.

[32] Amerini I, Galteri L, Caldelli R, Del Bimbo A. Deepfake Video Detection through Optical Flow Based CNN. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)

[33] Gloe T, Fischer A, Kirchner M. Forensic analysis of video file formats. Digital Investigation. 2014 May;11:S68–76

[34] Iuliani M, Shullani D, Fontani M, Meucci S, Piva A. A Video Forensic Framework for the Unsupervised Analysis of MP4-Like File Container. IEEE Transactions on Information Forensics and Security. 2019 Mar;14(3):635–45.