Challenges and Pitfalls in Video Watermarking using Motion Vectors

BART WESTENENK, University of Twente, The Netherlands

As universities are moving to online learning with lecture videos, some lecturers have concerns that their videos can be copied and claimed as the property of someone else. To ensure their videos are protected, lecturers can add watermarks. By watermarking videos, viewers cannot copy the video and claim it as their own. Visible watermarks are watermarks that viewers can see with the naked eye. Although effective, they also taint the video and influence the viewer's experience. Invisible watermarking using motion vectors can provide the protection while not tainting the original video. Literature on invisible watermarking in video using motion vectors currently exists but gloss over the implementation details and the influence the chosen codec has. Due to this oversight, implementation of these algorithms may seem trivial, while it is not. Due to codec complexity and lack of available tooling. In some cases, using the most modern codec imposes challenges that do not appear when using older codecs. This paper provides a novel overview of the most commonly used codecs and their advancements through time which can be used in future research. It finds challenges and pitfalls in implementation such as lack of tooling and complex codec specifications. Which, in turn, makes the field less approachable. This paper contributes to the approachability of the video watermarking research field by providing novel insights into these challenges and pitfalls, provides recommendations how these can be tackled in future research and which currently available tooling can be used. There is probable use for future research aiming to implement novel algorithms.

Additional Key Words and Phrases: Data Hiding, Video Watermarking, Codec, Motion Vectors, Copyright protection

1 INTRODUCTION

In the last few years, lecturers at universities are publishing more videos on their Learning Management Systems (LMS) as universities are moving more into hybrid learning due to the COVID-19 pandemic. Some lecturers may not want to publish videos because they have concerns whether their videos can be copied and shared outside the LMS. They have copyright on those videos [7] and they have the right to protect that copyright. There are various ways of protecting that copyright. One way is by watermarking the video. Watermarks alter the original video to visibly show that the video has been downloaded or copied from elsewhere. An example can be seen in fig. 1, this could be limiting for the student, as some part of the video is now unusable and the video is tainted.

To solve this problem, invisible watermarking can be used. Invisible watermarking has the same goal as visible watermarking namely, embedding a message visible for all viewers and is hard to remove. Invisible watermarking hides the watermark by altering properties of the video, this way there is a minimal disturbance to the original video while the copyright is still protected. If students know the videos are invisibly watermarked, they are discouraged in sharing the video outside of the LMS.

TScIT 41, July 5, 2024, Enschede, The Netherlands

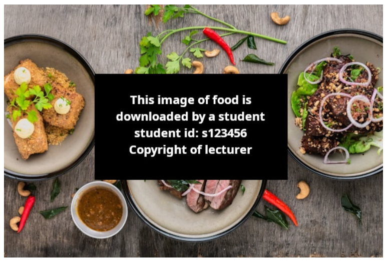

Fig. 1. Example of visible watermarking, showing who has downloaded the image and who the original author is. Source: Adapted from [14]

In general, videos are constructed by using intra frames (I-frames), predicted frames (P-frames) and bi-directional predicted frames (B-frames) [17]. I-frames are frames that do not depend on other frames and are standalone images. P- and B-frames depend on I-frames and other P- and B-frames by describing the changes in the frame using motion vectors. Motion vectors describe the movement of a block of pixels called a macroblock. These motion vectors can be used to watermarking videos by modifying them slightly.

1.1 Motivation and novelty

Invisible watermarking has many applications, including the aforementioned copyright protection use, fingerprinting of videos and content filtering [3]. A small review of existing literature was conducted. In this, it appeared that existing literature gave minimal insight in the way algorithms are implemented and why a certain coding standard was chosen. This makes this field of research less approachable for future researchers. This paper provides recommendations and directions on how to make this field more approachable.

By also reviewing and analysing the available coding standards, this paper makes the field more approachable. Previous reviews of video watermarking [8, 15, 20] did not consider implementation details nor the selection of a codec. This paper will provide a novel overview on the differences between video coding standards with respect to motion vectors. This paper will also provide a list of requirements that video editing tools should implement to make it useful in this field of research.

1.2 Scope

Although I-frames exist and they can also be used for invisible watermarking, many research already exists on the topic. Motion vector based watermarking has been investigated less in literature. Due to time limitations this paper will not focus on I-frame based watermarking. Instead, this research has been scoped to focus on invisible watermarking based on motion vectors. In terms of the challenges and pitfalls that can occur during implementation, the focus has been given to the most commonly used codec H.264 [4]. An analysis is done to give insight in the video watermarking tools available for this codec.

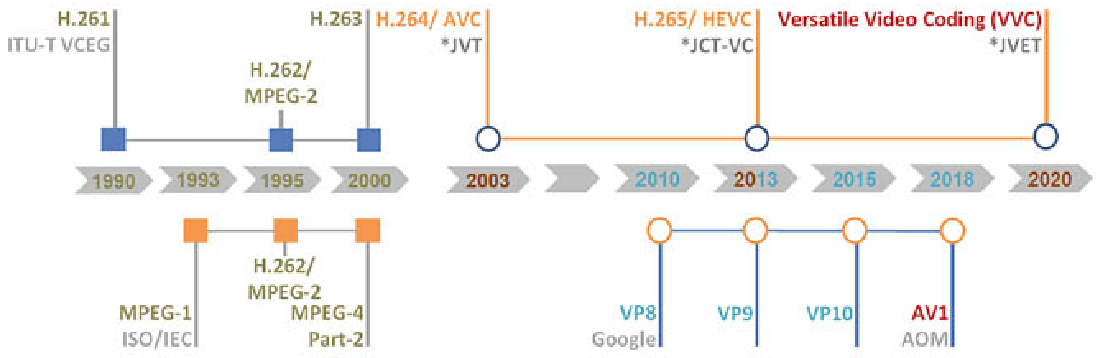

Fig. 2. Video coding standards development timeline. Source: [18]

This paper is structured as follows. First, the available video coding standards, also called codecs, which can be used to implement motion vector based watermarking algorithms are discussed. By analyzing the development of features in the codecs that relate to motion vectors, an overview will be given of the coding standards. Second, some of the existing algorithms that use motion vectors are discussed in section 4.2 and give a conclusion to RQ1 and RQ2 in section 4.3. Furthermore, recommendations are provided on how to choose a codec for an algorithm. Third, in section 5, requirements for tooling to use in video watermarking research will be constructed and currently available tooling will be analysed based on those requirements.

2 PROBLEM STATEMENT

In this section, the goal of the research and the research questions are discussed. As mentioned earlier, the goal of this paper is to identify potential challenges in the implementation phase of research. This is done by answering the research questions below. These questions will be used to reach the aforementioned goal.

Goal: Identifying hidden challenges and pitfalls in implementing motion vector based watermarking algorithms in H.264.

- **RQ1:** To what extent can motion vectors in B-frames and P-frames be used for watermarking in video?
- **RQ2:** To what extent does the choice of codec influence the possible watermarking algorithms?
- **RQ3:** What challenges could be faced while implementing watermarking algorithms in H.264 video?

3 METHOD

To answer RQ1, a small literature survey has been done in order to gather resources on video steganography and specifically motion vector-based video steganography. This literature survey includes papers that are deemed most influential according to review papers in existing literature [8, 15, 20] and propose algorithms that are based on motion vectors. These papers are then analyzed in which way they use motion vectors, and how it can be used to watermark videos.

To answer RQ2, existing codecs are analyzed on their significant improvements since their predecessors. This analysis also includes an overview of how video codecs are generally structured to give insight in what properties can be used in future research. Afterwards, the previously analyzed papers from the literature survey will be analyzed once more on which codec is used as well as the reasoning behind it. If there is no codec or reasoning given, an analysis will be done as to which codecs would be suitable to implement the algorithm.

To answer RQ3, an analysis into the tooling available for modifying video in the compressed domain of H.264 video is provided. H.264 was chosen because it is the most commonly implemented codec and is still widely used today [4]. The results from RQ3 aim to provide insight in what requirements future tooling should fulfill to be useful in terms of video watermarking, in particular with respect to motion vectors.

4 RELATED WORKS

In this section, some of the currently available codecs, the general idea of video codecs and the technological advances with regards to motion vectors are discussed. After discussing the available codecs, some existing data hiding algorithms that use motion vectors are also discussed as well as their respectively used codecs. Thereafter, a conclusion to RQ1 and RQ2 is drawn and a recommendation is provided on choosing a video codec in section 4.3.

4.1 The general structure of digital video

Since the release of H.261 (in 1990) [11], the design of all major video coding standards follow the so-called block-based hybrid video coding approach. A frame is divided into blocks of $x \times y$ called macroblocks. Each macroblock of a frame picture is either intra predicted (I-frame) without referencing other pictures in the video, or is temporally predicted, where the signal is formed by

Coding standard	Release	Max. resolution ∗	Frame-independent coding ∗∗	GRF	Merge mode	Royalties
H.262	1996	1920×1080@30	No	No	No	Expired
H.264	2003	8192×4320@120.9	Yes	No	No	Yes
H.265	2013	8192×4320@120.9	Yes	No	Yes	Yes
VP8	2010	16383×16383 ∗∗∗	No	Yes	No	None

∗ This assumes the usage of the highest profiles, lower level profiles may support lower resolutions.
∗∗ This means that macroblocks can have different prediction modes, independent of the type of frame, introduced in H.264.
∗∗∗ VP8 does not place restrictions on framerate.

Table 1. Overview of codecs and important properties

moving a block of an already coded picture. Temporally predicted pictures referencing previous pictures are called P-frames. When also referencing future pictures, these are called B-frames. Sometimes research also mentions a unit called Group of Pictures (GOP) to denote a set of frames consisting of at least 1 I-frame and some temporally predicted frames. [17]

Figure 2 shows that since the release of H.261, a lot of newer codecs have emerged. A selection of these codecs is discussed on their origin, usage, reasoning for introduction and licensing. Some differences that make the codecs more efficient are not considered in this paper. For example, H.264 uses CABAC encoding and H.265 and VP8 optimized this. This compression technique can influence the choice of codec, but does not provide any extra properties in motion vectors that can be used to hide data in a video and are thus out of scope for this research. Video codecs like H.262 [13], H.264 [10], H.265 [12] and VP8 [25] all stem from this original coding approach. Although they all stem from the same general idea, there are still many differences in implementations and some of these codecs are specialized in certain areas. Due to time constraints put on this research, only these codecs are discussed. These are the most commonly implemented codecs [4] and were most influential in the development of more modern codecs.

4.1.1 H.261. The original H.261 codec [11] as introduced in 1990 was introduced to make it possible to send video over telephone lines, they foresaw that a number of audiovisual services would be likely to appear if there was a standardized way of transmitting video over telephone lines. Although H.261 is not in use anymore due to the introduction of newer codecs, it can be seen as the basis of all modern codecs.

4.1.2 H.262. The oldest codec still in use to date is H.262 [13], officially launched in 1996. It was developed as a joint project of ITU-T and ISO/IEC JTC 1 [17]. Its original purpose was to respond to the growing need for a coding method for various applications such as digital storage media and television broadcasting. It served as a useful standard to store video digitally and in transferring video over digital networks. Compared to H.261, there are not many differences. H.262 for example supports HDTV and interlacing, a technique used in older analog television systems, where H.261 does not support this. In terms of performance H.262 should, depending on the codec implementation, perform better than H.261. Although originally H.262 was not royalty-free, due to its age the royalty rights have expired.

4.1.3 H.264. H.264 is the most used codec according to Bitmovin's 2020 video developer report [4]. It is used in 86% of production environments and supported by all major browsers like Google Chrome, Firefox and Microsoft Explorer [5]. In contrast to H.262, its purpose was also to target videoconferencing and internet streaming. It also allows for even an even higher resolution of 8192×4320@120.9, although not all decoders support it, and is more efficient than its predecessors.

An obvious difference from older standards is its increased flexibility in intercoding. For the purpose of motion-compensated prediction, a macroblock can now be partitioned in square and rectangular block shapes with sizes ranging from 4×4 and 16×16. H.264 coded video contains slices instead of pictures. Pictures can contain multiple slices, those slices are not required to be of the same type. A picture can be used as a reference frame independently of the type of the slices contained in that picture. To be allowed to use H.264 in either software or hardware, a licensing fee must be paid to Via LA licensing [23]. Cisco influenced the popularity positively by releasing a free H.264 codec called OpenH264 [6], Cisco pays the license costs for their provided binaries, making H.264 essentially a free coded. In response, Firefox was able to implement H.264 in their browser.

4.1.4 H.265. The codec that the joint project ITU-T and ISO/IEC MPEG developed after H.264 is H.265, this codec again increases the resolution. H.265 also supports resolutions up to 8192×4320@120.9, but due to the advances in H.265 more decoders can actually decode at that speed. Compared to former codecs, it has improved the motion vectors by a lot. It supports a so-called merge mode, where no motion vectors are coded. Instead, they are derived from the corresponding prediction unit (PU), which consists of the chroma and luma prediction blocks. Each PU contains one or two motion vectors used for unipredictive or bipredictive coding. No research was found that uses the merge mode in H.265 to embed data. To use H.265, a company needs a license agreement similarly to H.264. This had implications on the adaption of H.265 as a successor to H.264, making it still not as widely used as H.264 [4].

4.1.5 VP8 and VP9. Google released the VP8 codec [25] as an open source codec aimed to provide a way to transfer video in low bandwidth environments. It was focused on web-based video applications and thus, due to the nature of the web, it must be possible to implement the codec on a wide range of devices. Compared to the existing codecs like H.262 and H.264, Google mainly focused on optimizing the use of I-frames. VP8 stores a reference frame in a buffer from arbitrary points in the past of the video. They call these reference frames Golden Reference Frames (GRF). Golden Reference Frames are useful when objects or backgrounds disappear for a moment and come back. This greatly helps compression efficiency. This is, for example, applicable in a TV show switching between 2 camera angles. VP9 succeeded this standard which is a more efficient version of VP8. VP8 and VP9 are provided for royalty-free.

4.2 Existing data hiding algorithms in P- and B-frames

The selection of the candidate motion vectors (CMV) is the key difference in existing data hiding algorithms. The selection method influences the method of embedding greatly as embedding may not influence the selection criteria as this may lead to the motion vector not being selected during extraction. In general, all of these algorithms either operate on the complete video, or operate on a per GOP basis. This means that each GOP will hide its own message and can be independently decoded.

4.2.1 Magnitude. Zhang et al. [26] proposed an algorithm that selects the CMV based on the magnitude. This algorithm is meant for steganography instead of watermarking. It uses a key to determine the required threshold. The secret bits are embedded in the CMV by modifying the x and y component based on the current angle of the motion vector. Zhang et al. did not specify for which specific codec they implemented the algorithm. At their point in time, only H.261 and H.262 were available, thus they probably used H.261 as they didn't use any features of H.262 in their algorithm. [26]

4.2.2 Phase angle. Hao-bin et al. [9] proposed an algorithm based on the algorithm by Zhang et al. They also based the candidate selection algorithm on the magnitude of the CMV. However, they also considered the angle in which the motion vector points. They claim that if a vector is vertical and the horizontal distance is zero, then a slight change in the horizontal component has greater influence than a change in the vertical component. Both Hao-bin et al. and Zhang et al. determine the CMV by a predefined threshold, but Hao-bin et al. lets the phase angle determine in which component the data will be embedded. They have used the JM17.2 reference encoder to implement their algorithm in H.264 video. [9]

4.2.3 Prediction error. Aly et al. [2] proposed an algorithm that selects the motion vectors based on their so called prediction error. According to them, algorithms selecting CMV on magnitude choose to do so because the chance that those motion vectors accurately represent the motion is smaller and changes are less notable. Aly et al. claimed that this is not always correct, and that selecting based on their associated prediction error is more accurate, resulting in less disturbance to the video quality. Aly et al. implemented their algorithm into the H.262 reference encoder and decoder. [2]

4.2.4 Low and high texture areas. Liu et al. [16] introduced a way of improving the embedding performance by increasing the amount of embedded bits based on the texture of the macro block that is affected by the motion vector. The texture of a macroblock is the amount of different pixels in a macroblock. For example, a macroblock containing the sky is a low texture area and macroblock containing a face is a high texture area. In low texture areas, the embedded bits can be increased compared to high texture areas as the effect of the change is less visible for low texture areas. They did not specify the codec they used nor the way of implementation. [16]

4.3 Recommendations and conclusions

In this section, some of the codecs available and the technical differences in terms of motion vectors are discussed. Table 1 shows a list of the discussed codecs and their properties. Future researchers that are interested in using motion vectors for video watermarking, are recommended to choose a codec that implements the properties that they require. All previously mentioned algorithms can be implemented in H.262 as well as modern codecs. Future research should take the complexity of the codec that it uses into account as modifying a H.262 video is easier compared to modifying H.264 videos and VP8 videos. These recommendations are given with the aim to limit the time spend on implementations, and increase the time spend constructing algorithms.

In conclusion, based on the literature discussed earlier, motion vectors can be used extensively for data hiding in P- and B-frames. Furthermore, although the codec does influence the complexity of modifications, it does not have a great influence on the possible watermarking algorithms. Moreover, it depends on which properties of the motion vectors will be modified and whether those properties exist in the targeted codec. For example, if the research will go into using motion vectors based on the golden reference frames of VP8, then only VP8 may be considered as target codecs for the implementation. This has implications on the availability of tools and libraries, as will be discussed in the following section.

5 CHALLENGES AND PITFALLS

One challenge of working with video codecs is understanding how the actual bitstreams of videos are structured. Although an overview of how the most common video codecs work is provided in section 4, it does not cover how one can read or modify the bitstream of a video. It must be noted that modifying the bitstream of any codec is not a trivial task and that may be a pitfall for researchers new to this field of research. This section will go over some of the tools that have been used by other researchers to implement watermarking algorithms and others that one may find when lookin for tools and libraries to use for a watermarking algorithm. Most of these tools and libraries can be found online, but unfortunately, most algorithms considered in section 4.2 did not provide any source code or an overview of how their implementation works.

5.1 Modification tool requirements

To determine whether a tool is useful for video watermarking research, it must fulfill some requirements. After considering existing tools and analyzing existing watermarking algorithms, the following list of requirements has been constructed:

(1) *Must* produce a syntactically correct bitstream. The implementation must block the user from writing syntactically incorrect bitstreams.
Reason: If a bitstream is not syntactically correct, standard conforming decoders will not load the video.

(2) *Must* produce a semantically correct bitstream. The implementation must block the user from writing semantically incorrect bitstreams.
Reason: If a bitstream is not semantically correct, standard conforming decoders will either determine the video as corrupted or show undocumented behaviour. This removes the mental overhead to think about the side effects on the bitstream when modifying a value.
(3) *Must* be able to modify P- and B-frames (motion vectors) without re-encoding, in a determinant manner.
Reason: To implement motion vector based algorithms, it is required to be able to modify specific, predetermined motion vectors.
(4) *Should* be able to modify I-frames without re-encoding.
Reason: To implement algorithms that also use I-frames, it would be useful to also be able to modify I-frame properties without re-encoding.
(5) *Should* be capable of modifying videos coded using different codecs.
Reason: To make sure the tool stays relevant after new codecs are released, it needs to be able to adapt to new codecs.

These requirements are written with the following goal in mind: Ease the implementation of watermarking algorithms and improve the approachability of the video watermarking field. Although this paper mainly aimed to provide useful insights with respect to motion vectors, I-frame based watermarking should not be forgotten.

5.2 Previous research on altering video bitstreams

In the past, other researchers have also tried to find and develop methods to modify the video bitstream in a consistent and reliable manner. Unfortunately, the amount of research carried out in this field is limited. The research that has been carried out is not fitted for watermarking applications on a higher level.

5.2.1 H26Forge. H26Forge [22] is a project specifically tailored to making the H.264 bitstream editable on a low level by Vasquez et al. The goal of this project was generating syntactically correct bitstreams that are not necessarily semantically correct. Because motion vectors are not directly encoded into the bitstream, it would need extensive knowledge of the semantics of the H.264 codec to edit a specific motion vector. Due to the time limit of this research, it is not possible to have such in depth knowledge about the codec. The authors of H26Forge took multiple years to develop H26Forge. H26Forge also supports H.265 in some way, but does not have full support. Future research can go into using the basis H26Forge laid to provide a more standardized way to modify specific properties that are not directly encoded in the bitstream.

5.2.2 Compressed domain processing. Wee et al. [24] wrote a book chapter analyzing and describing various algorithms on efficiently processing video in the compressed domain. It contains a comprehensive amount of knowledge on how a bitstream is formatted and how it could be altered on a high level. Unfortunately, they have only described their algorithms in a theoretical sense and did not provide any implementation or, recommendations for an implementation. For future works it would be useful if the theories they have laid out in their book, could be implemented into a framework that can be used for future research.

5.3 Custom encoders

Most of the papers found modified existing encoders to implement their algorithms. This includes the reference encoder for H.264 called the JM software encoder [21], currently at version 18.0. Hao-bin et al. [9] used version 17.2 to implement their algorithm. Besides the reference implementation, also Cisco's OpenH264 implementation [6] is an option to use for implementing an algorithm. Unfortunately, due to the way the bitstream is structured, motion vectors are not directly encoded into the bitstream. This makes it difficult to find where and when the motion vectors are calculated and where the modification of the motion vectors should happen. It is a difficult task to modify an existing encoder to fulfill the needs for a watermarking algorithm.

5.4 PyAV

This project implements Python bindings for most of FFmpeg's low level libraries like `libav`, `libavcodec` and others. Due to the way this project is setup, it is not possible to alter the motion vectors or make alterations on a macro block level. This is due to python not being able to directly access the c++ buffer that is used by the underlying c libraries. This meant, in practice that altering a frame using PyAV, required full decoding of the frame and reencoding after the modifications. Meaning that the motion vectors are recalculated and the motion vectors cannot be reliably modified, making it not suitable for motion vector based watermarking algorithms.

5.5 FFmpeg

FFmpeg [1] is a tool meant for converting, editing and transcoding various video files and codecs. It is meant to work with various different codecs and is thus built with many abstraction techniques. The possible types of FFmpeg filters are: audio/video filters (AVFilters) and bitstreamfilters (BSFilters). Unfortunately, due to the aim of FFmpeg to be compatible with as many codecs as possible, it is not possible to write an AVFilter or BSFilter which edits the low level properties of a specific codec. It suffers from a different problem than PyAV, as it is possible using a BSFilter to alter the direct buffer. Unfortunately, it is still non-trivial to alter a specific property like the motion vectors as it requires an implementation to first decode the motion vectors in a frame, and then modify them and re-encode them correctly into an existing bitstream. However, the bitstream consists of more properties than only motion vectors, so other properties that depend on the encoding used in the motion vectors may be affected, resulting in a possibly invalid bitstream.

5.5.1 FFglitch. A project by Polla based on FFmpeg called FFglitch [19] was made with the aim to modify the motion vectors of a H.262 video. This tool may be useful to implement algorithms that only use motion vectors in its implementation. It currently only supports H.262 video and H.264 video support is on its roadmap. Future research could extend this project to also allow other codecs to be modified and other properties of video like, for example, properties of I-frames to allow for combining the two properties.

Name	(1)	(2)	(3)	(4)	(5)
H26Forge	Yes	No	No	No	No
PyAV	Yes	Yes	No	No	Yes
FFMpeg	Yes	Yes	No	No	Yes
FFGlitch	Yes	Yes	Yes	No	Yes

Table 2. Fulfillment of the aforementioned requirements by the available tools discussed in this section.

5.6 Recommendations and conclusions

The goal of this research is to recommend requirements a tool should fulfill and investigate which tools already fulfill these requirements. Table 2 shows an overview of these requirements. In conclusion, based on the performed analysis, there is still a lack of proper tooling to consistently implement video watermarking algorithms based on motion vectors. To propose an algorithm at this moment, the researcher must spend more time on the implementation, than on the design of the algorithm. In the opinion of the authors, the work spent on the implementation can be reduced by providing more tools. The authors recommend future researchers aiming to implement watermarking algorithms using motion vectors in H.262, to use FFglitch. Future research into new tooling or building a framework for video watermarking research, are recommended to take the aforementioned requirements into account. FFglitch and H26Forge are recommended as a basis and/or inspiration for future works.

In conclusion, some recommendations for future works that can be undertaken by future researchers are provided. Most notably, in this field of research, there is a lack of proper tooling to change specific properties of the video. This greatly influences the time needed to implement an algorithm and the deep understanding of the specific codec that is used for the implementation. This may result in a limited interest by future researchers that are new to this field. All in all, the biggest pitfall and challenge for newer researchers are the complexity of the coding standards and the absence of proper tooling.

6 CONCLUSION AND FUTURE SCOPE

In this research, we have explored the various codecs available, the viability of using motion vectors for video watermarking and the hidden challenges and pitfalls researchers may be confronted with during implementation. We have shown that although resources on video watermarking using motion vectors exist, that there is a lack of adequate tooling to implement these algorithms. Currently, implementing these algorithms require a deep understanding of very complex codecs, making it difficult to implement trivial algorithms. Making video watermarking a field that is not easy to step into. This research has mainly focused on implementation challenges and pitfalls for the H.264 codec. Furthermore, this research has mainly focused on motion vectors and did not take into account other properties in these codecs. Future research can be done in a similar direction, but aimed at implementation challenges in I-frame based watermarking, or watermarking in H.265 and VP8. As the coding used in I-frames is largely based on regular image coding, research in invisible image watermarking can be translated to invisible video watermarking as has already been done in the past. However, this field will also have various challenges and pitfalls that can be investigated. Future research can also focus on providing more tooling or to develop a framework that tooling should adhere to altering compressed videos in a reliable way. Researchers that want to undertake developing a novel algorithm using motion vectors, are advised to choose their codec wisely as the complexity of the codec can greatly increase the time needed to properly implement the algorithm for acquiring experimental results. The merge mode in H.265 and Golden Reference Frames in VP8 can also be considered as potential targets to implement novel algorithms.

REFERENCES

[1] 2024. FFmpeg/FFmpeg. https://github.com/FFmpeg/FFmpeg original-date: 2011-04-14T14:12:38Z.

[2] Hussein A. Aly. 2011. Data Hiding in Motion Vectors of Compressed Video Based on Their Associated Prediction Error. *IEEE Transactions on Information Forensics and Security* 6, 1 (March 2011), 14–18. https://doi.org/10.1109/TIFS.2010.2090520 Conference Name: IEEE Transactions on Information Forensics and Security.

[3] Md. Asikuzzaman and Mark R. Pickering. 2018. An Overview of Digital Video Watermarking. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 9 (Sept. 2018), 2131–2153. https://doi.org/10.1109/TCSVT.2017.2712162 Conference Name: IEEE Transactions on Circuits and Systems for Video Technology.

[4] Bitmovin. 2023. *The 7th Annual Bitmovin Video Developer Report.* Technical Report. https://bitmovin.com/downloads/assets/bitmovin-7th-video-developer-report-2023-2024.pdf

[5] BrowserStack. 2024. MPEG-4/H.264 video format | Can I use... Support tables for HTML5, CSS3, etc. https://caniuse.com/mpeg4

[6] Cisco. [n. d.]. OpenH264. https://github.com/cisco/openh264/releases

[7] Council of European Union. [n. d.]. DIRECTIVE (EU) 2019/790 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. https://eur-lex.europa.eu/eli/dir/2019/790/oj Doc ID: 32019L0790 Doc Sector: 3 Doc Title: Directive (EU) 2019/790 of the European Parliament and of the Council of 17 April 2019 on copyright and related rights in the Digital Single Market and amending Directives 96/9/EC and 2001/29/EC (Text with EEA relevance.) Doc Type: L Usr_lan: en.

[8] Mukesh Dalal and Mamta Juneja. 2021. A survey on information hiding using video steganography. *Artificial Intelligence Review* 54, 8 (Dec. 2021), 5831–5895. https://doi.org/10.1007/s10462-021-09968-0

[9] Hao-Bin, Zhao Li-Yi, and Zhong Wei-Dong. 2011. A novel steganography algorithm based on motion vector and matrix encoding. In *2011 IEEE 3rd International Conference on Communication Software and Networks.* 406–409. https://doi.org/10.1109/ICCSN.2011.6013622

[10] ISO. [n. d.]. Information technology - Coding of audio-visual objects - Part 10: Advanced video coding. https://www.iso.org/standard/83529.html

[11] ITU-T. 1993. H.261 : Video codec for audiovisual services at p x 64 kbit/s. https://www.itu.int/rec/T-REC-H.261-199303-I/en

[12] ITU-T. 2023. H.265 : High efficiency video coding. https://www.itu.int/rec/T-REC-H.261-199303-I/en

[13] ITU-T and ISO/IEC. [n. d.]. H.262 : Information technology - Generic coding of moving pictures and associated audio information: Video.

[14] Banse Lily. 2024. Cooked dish on gray bowl. https://unsplash.com/photos/cooked-dish-on-gray-bowl--YHSwy6uqvk

[15] Yunxia Liu, Shuyang Liu, Yonghao Wang, Hongguo Zhao, and Si Liu. 2019. Video steganography: A review. *Neurocomputing* 335 (March 2019), 238–250. https://doi.org/10.1016/j.neucom.2018.09.091

[16] Zina Liu, Huaqing Liang, Xinxin Niu, and YixianYang. 2004. A robust video watermarking in motion vectors. In *Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP '04. 2004.*, Vol. 3. 2358–2361 vol.3. https://doi.org/10.1109/ICOSP.2004.1442254

[17] Jens-Rainer Ohm, Gary J. Sullivan, Heiko Schwarz, Thiow Keng Tan, and Thomas Wiegand. 2012. Comparison of the Coding Efficiency of Video Coding Standards—Including High Efficiency Video Coding (HEVC). *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (Dec. 2012), 1669–1684. https://doi.org/10.1109/TCSVT.2012.2221192

[18] Andreas S. Panayides, Marios S. Pattichis, Marios Pantziaris, Anthony G. Constantinides, and Constantinos S. Pattichis. 2020. The Battle of the Video Codecs in the Healthcare Domain - A Comparative Performance Evaluation Study Leveraging VVC and AV1. *IEEE Access* 8 (2020), 11469–11481. https://doi.org/10.1109/ACCESS.2020.2965325 Conference Name: IEEE Access.

[19] Ramiro Polla. 2024. ramiropolla/ffglitch-core. https://github.com/ramiropolla/ffglitch-core original-date: 2014-11-04T13:28:59Z.

[20] Mennatallah M. Sadek, Amal S. Khalifa, and Mostafa G. M. Mostafa. 2015. Video steganography: a comprehensive review. *Multimedia Tools and Applications* 74,

17 (Sept. 2015), 7063–7094. https://doi.org/10.1007/s11042-014-1952-z

[21] Alexandros Tourapis. [n. d.]. H.264/14496-10 AVC Reference Software. . *General Information* ([n. d.]).

[22] Willy R Vasquez, Stephen Checkoway, and Hovav Shacham. [n. d.]. The Most Dangerous Codec in the World: Finding and Exploiting Vulnerabilities in H.264 Decoders. ([n. d.]).

[23] ViaLa. 2024. AVC/H.264 License Fees. https://www.via-la.com/licensing-2/avc-h-264/avc-h-264-license-fees/

[24] Susie Wee, Bo Shen, and John Apostolopoulos. [n. d.]. Compressed-Domain Video Processing. ([n. d.]).

[25] Paul Wilkins, Yaowu Xu, Lou Quillio, James Bankoski, Janne Salonen, and John Koleszar. 2011. *VP8 Data Format and Decoding Guide.* Request for Comments RFC 6386. Internet Engineering Task Force. https://doi.org/10.17487/RFC6386 Num Pages: 304.

[26] Jun Zhang, Jiegu Li, and Ling Zhang. 2001. Video watermark technique in motion vector. In *Proceedings XIV Brazilian Symposium on Computer Graphics and Image Processing*. 179–182. https://doi.org/10.1109/SIBGRAPI.2001.963053

ACKNOWLEDGMENTS

During the preparation of this work the author(s) used ChatGPT in order to: provide boilerplate for implementation, aid in understanding the various coding standards and improve grammar. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work.