DOI:10.1587/transinf.2024MUP0003

Publicized:2024/10/07

PAPER

Video Watermarking Method Based on 3D U-Net Robust Against Re-shooting

Takaharu TSUBOYAMA[†], **Ryota TAKAHASHI**[†], ***Nonmembers***, **Motoi IWATA**[††a)], ***Senior Member***, ***and* Koichi KISE**[††b)], ***Fellow***

SUMMARY In recent years, digital signage has become popular as a means of information dissemination to the general public. However, unlike advertisements displayed on PCs or smartphones, it is impossible to directly acquire information displayed on such signages even if the content is interesting. Mizushima et al. proposed a video watermarking method that is robust against re-shooting so that the watermark can be extracted from watermarked videos displayed on digital signage. Conventional methods have the problem of limited information capacity. In recent years, watermarking methods based on deep learning have attracted attention for embedding large watermarks. In this paper, we implemented a video electronic watermark based on 3D U-Net, which makes it possible to embed larger watermarks than existing methods. In addition, the proposed method was able to extract the watermark from the re-shot video, and the shortest average processing time is 1.85 seconds to extract the correct watermark.

key words: *Digital signage, 3D U-Net, Video watermarking, Re-shooting*

1. Introduction

In recent years, digital signage, a display type information dissemination media, has become increasingly popular and can be seen in train stations and on the streets. Since digital signage can transmit information to the masses, it easily attracts users' interest. However, users cannot directly obtain detailed information even if they are interested in the content. Therefore, a method to embed watermarks in videos to get information is being considered. Digital watermarking is a technology for embedding additional information in multimedia content such as images, videos, and music by making changes too small to be perceived by humans[1]. The embedded information is called a watermark, and a video with an embedded watermark is called a watermarked video. It is convenient for users that they can extract watermarks from watermarked videos on signage by re-shooting them by smartphone. As a watermark, the shortened URL of a website that provides detailed information on the video is useful because users can obtain detailed information from the watermarked video. For example, a shortened URL of a travel reservation website could be embedded as a watermark in a travel video advertisement. Considering that the response time for a user is reported to be 2 to 8 seconds[2], extracting the watermark should be fast. We assume that there is no malicious attack on the watermarked video since there is no advantage for the party displaying the video and the party wishing to obtain detailed information.

Mizushima et al. proposed a method to extract a watermark from frames re-shot by a smartphone by estimating the watermarked area using traces of the embedded watermark[3]. This method can embed a 22 bits watermark and work in real-time on smartphones. However, it has the disadvantage of degrading the image quality of watermarked videos to obtain a reliable watermark. Other problems are the perception of block noises in watermarked videos and the low capacity. The block-based embedding process is the cause of them. Independent embedding for each block makes perceiving boundaries of them and the tradeoff between the amount of embedding bits, that is the number of blocks, and the robustness which is related to the size of blocks. As an alternative method, Hui et al. proposed a watermarking method for camcorder recording that uses the similarity between adjacent video frames to embed a watermark. This method is robust against geometric and frame rate changes, especially for camcorder recordings[4].

In recent years, watermarking methods based on deep learning have attracted attention for embedding large watermarks. Tancik et al. proposed an image watermark called StegaStamp[5]. StegaStamp is robust against re-shooting for still images. It uses deep learning to embed watermarks, detect watermark areas from re-shot images, and extract watermarks. It used U-Net[6] to embed the watermark. The U-Net is trained to embed a watermark in the image so as to be robust against re-shooting. However, since the video is composed of sequential frames, the 2D U-Net model cannot use the temporal features.

In this study, we use 3D U-Net, the 3D extension of U-Net, to use temporal features of the video as a model for watermarking. In this paper, we describe related work in Sect. 2, embedding a watermark in video and extracting the watermark using deep learning in Sect. 3, the performance of the proposed method in extracting watermarks from re-shot videos in Sect. 4, and conclude in Sect. 5.

2. Related Work

2.1 Video watermarking methods robust against re-shooting

Mizushima et al. proposed a method to extract watermarks from watermarked videos by extracting watermarks from

[†]Osaka Prefecture University
[††]Osaka Metropolitan University
a) E-mail: imotoi@omu.ac.jp
b) E-mail: kise@omu.ac.jp

accumulated difference frames to improve image quality[3]. Mizushima et al.'s method is implemented on a smartphone and correctly extracts the watermark within one second in most cases. It is also stated to be robust against pixel changes due to re-shooting by using different frames. However, the watermarks used in the experiments were 16 bits and 22 bits in length, and watermarks of longer lengths have not been tested.

Hui et al. proposed a watermarking scheme for camcorder recording that uses the similarity between adjacent video frames to embed a watermark. A feature-based positioning algorithm is proposed to determine the embedding region and embedding strength so that the watermark is robust against geometric transformations during camcorder recording. For extracting the watermark, a synchronization algorithm and a cross-validation algorithm are used, and a local matching algorithm is proposed to reduce the difference between neighboring frames of the watermark. Experimental results show that the proposed algorithm is robust against geometric transformations, compression, etc., and is particularly robust against camcorder recordings. The weak point of Hui et al.'s method is that it requires a few frames to embed only 1-bit watermark. They said in the paper that their scheme is not suitable for short videos, because for videos of tens of seconds, it may not have enough frames to ensure the watermark embedding.

2.2 Video watermarking method using deep learning

Weng et al. have proposed a method of embedding video as a watermark in a video, focusing on the temporal redundancy of video[7]. Since consecutive frames of a video are often similar to each other, the difference between frames is considered to be small. Therefore, Weng et al. proposed a method to embed reference frames and differences alternately. The reference frame is the frame on which the differences are calculated. When extracting the embedded video, the reference frame and the difference are used to compute the frames of the embedded video. This method can reduce degradation compared to embedding frames directly in each frame. The weak point of Weng et al.'s method is low robustness against video processing. They investigated only the robustness against MP4 compression.

2.3 Image watermarking method based on deep learning robust against re-shooting

Tancik et al. use deep learning to embed a watermark in an image, detect the watermarked area from a re-shot image, and extract the watermark[5]. The system used for embedding and extracting watermarks is called StegaStamp. It is capable of embedding a string of text into an image and extracting the watermark from a printed or displayed image that has been re-shot. First, the encoder is trained to embed the message into the image while minimizing the perceptual differences between the input image and the watermarked image. Here, Tancik et al. employ U-Net[6] as the deep learning model, which receives four input channels, a 400×400 red, green, and blue channels of an input image, and one channel for the message, and produces the watermarked image. The decoder is a network trained to recover the message from the watermarked image. The decoder also uses a spatial transformation network [8] to enable extraction from geometrically transformed watermarked images. Our research is similar to Tancik et al.'s research in that it uses a deep learning model robust against re-shooting, but the difference is that the target of this research is a video, not an image.

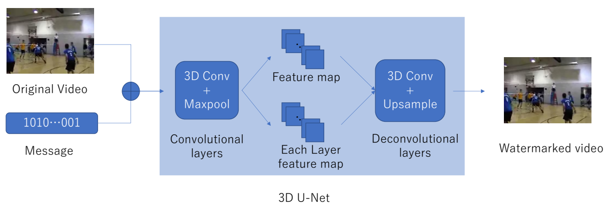


Fig. 1 Embedding a watermark.

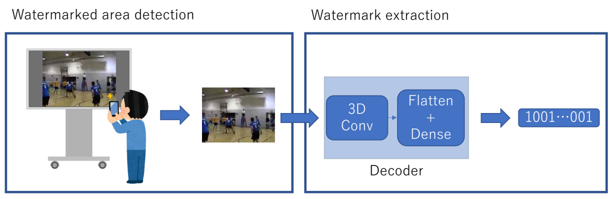


Fig. 2 Detecting a watermarked area in a re-shot watermarked video and extracting the watermark.

3. Proposed Method

Figure 1 shows the flow of embedding a watermark. Figure 2 shows the flow from watermarked area detection in a re-shot watermarked video to watermark extraction. In this section, we explain each of the steps in Fig. 1 and Fig. 2. Here, we have not implemented automatic detection of watermarked video areas yet, so the detection is done manually.

3.1 Encoder for embedding a watermark and decoder for extracting the watermark

The proposed method uses a 3D U-Net[9] with four convolutional layers and four deconvolutional layers. First, the following preprocessing is performed on the original video and watermark. The N_c-bit binary sequence $\boldsymbol{c}$ is processed through a fully-connected layer to form a $1 \times 15 \times 20 \times 3$ tensors, where the binary sequence $\boldsymbol{c}$ is the target of embedding (hereinafter called embedding sequence). Then $\boldsymbol{c}$ is upsampled to produce an $8 \times 240 \times 320 \times 3$ tensors. It has been reported by Tancik et al. that applying similar preprocessing to messages aids convergence. The $8 \times 240 \times 320 \times 6$ tensor, which is the combination of preprocessed $\boldsymbol{c}$ and the original 8-frames sequence of size $8 \times 240 \times 320 \times 3$, is input to the 3D U-Net to output an $8 \times 240 \times 320 \times 3$ watermarked frame sequence. Here, the size of the frame sequences $8 \times 240 \times 320 \times 3$

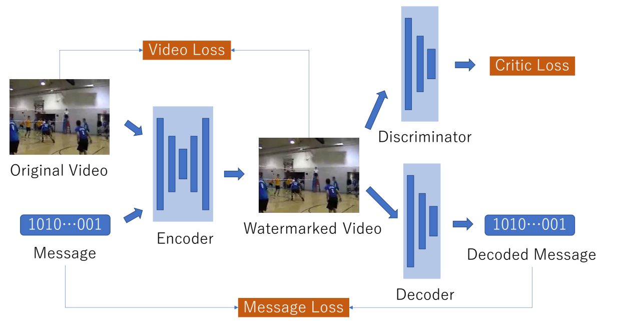


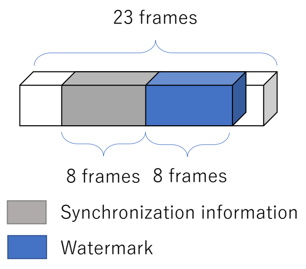
Fig. 3 Encoder for embedding a watermark and decoder for extracting the watermark training.

means [the number of frames] × [height] × [width] × [the number of color channels].

Next, a 3D CNN is used as the decoder for extracting the watermark, taking a $240 \times 320 \times 3$ pixels video as input, and outputting a N_c-bit binary sequence $\boldsymbol{c}'$ via 7 convolutional layers followed by a fully-connected layer. The cross entropy error between the extracted binary sequence $\boldsymbol{c}'$ and the correct embedding sequence $\boldsymbol{c}$ is the watermark loss. Figure 3 shows the training flow of the encoder for embedding a watermark and the decoder for extracting the watermark. We use a discriminator that predicts whether a message is encoded in the input video or not to obtain a critic loss. Lower critic loss means higher image quality of an input video. The network consists of 5 convolutional layers. For training the discriminator, we use Wasserstein loss[10]. As the losses related to the degradation of the image quality of the watermarked video (hereinafter called video loss), we employ the perceptual differences between the original and the watermarked video, that is, the mean squared error of RGB, the mean squared error of YUV, the LPIPS perceptual loss[11], and the critic loss. All these losses are used to train the encoder for embedding a watermark and the decoder for extracting the watermark simultaneously. The weight of the watermark loss is fixed at 1.0, and the weight of others is linearly increased with each training epoch. Hereafter, the encoder for watermark embedding and the decoder for watermark extraction are referred to as the embedder and extractor, respectively.

3.2 Noise for the training of the embedder and extractor

Noise by re-shooting includes white noise caused by reflections on a signage and camera systems. Hue, luminance, and contrast shifts are employed to reproduce the white noise as follows: First, we apply hue shift by adding a random color offset to each of the RGB channels sampled uniformly from $[-0.1, 0.1]$. Next, we apply luminance and contrast shifts based on affine histogram rescaling $mx + b$ with $m \sim U[0.5, 1.0]$ and $b \sim U[-0.3, 0.3]$. Various camera noises are possible, but a Gaussian noise model (standard deviation $\sigma \sim U[0, 0.02]$) is applied assuming standard non-photon-starved imaging conditions as in Stegastamp.



▨ Synchronization information

🟦 Watermark

Fig. 4 Construction of embedding sequences in watermarked video.

3.3 Construction of embedding sequences

To extract a watermark from the re-shot video, it is necessary to synchronize the start and end points of the 8 frames of watermarked video. The embedding sequence of N_c-bits of all zeros is used as the synchronization information. Figure 4 shows the composition of the embedding sequence in a watermarked video. First, N_c-bit synchronization information is embedded into the eight original frames for synchronization. Then, N_c-bit code word of watermark is embedded into the following eight original frames, where the code word of watermark is obtained by BCH encoding of N_W-bit watermark. Thus, the synchronization information and the code word of watermark are embedded alternately. As shown in Fig. 4, any 23 frames of re-shot video always contain frame sequences with both the synchronization information and the watermark. Synchronization is performed with respect to the frame sequence from which the highest number of zeros are extracted by extracting eight frames at a time in a sliding window from the 23 frames.

3.4 Embedding watermark

The watermark is embedded using the embedder described in Sect. 3.1. Eight frames of an original video and N_c-bit embedding sequence described in Sect. 3.3 is input to the embedder. This embedding process is repeated until the end of the original video, selecting eight frames of the original video without overlapping.

3.5 Extracting watermark

The extractor described in Sect. 3.1 is used to extract the watermark. Figure 5 shows the flowchart of extracting the watermark. The watermark extracting process consists of the synchronization of watermarked frames and a reliability check. First, successive 23 frames in a watermarked video are extracted and used for synchronization. The eight frames where a watermark was embedded can be found before or after the frame sequence with the most zeros. The reliability check is performed to confirm that the watermark is extracted correctly. The reliability check is based on the number of corrected error bits when BCH decoding the code word. The extracted watermark is considered reliable if the number of

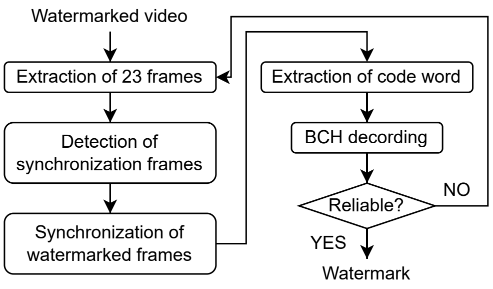


Fig. 5 Flowchart of extracting the watermark.

corrected error bits is less than or equal to the error correction capability of the BCH code. Otherwise, the extracted watermark is discarded, and then the process returns to the first step in Fig. 5, i.e., the extraction of the 23 frames from the re-shot video. The next 23 frames are extracted in such a way that they do not overlap with frames that have already been processed for extracting the watermark.

4. Experiment

4.1 Experimental conditions

In this study, we used the video dataset UCF101[12], which is used in other video watermarking method research[13], [14]. This dataset is classified into 101 classes of human motion and consists of more than 13,000 files, totaling 27 hours of video. The frame size is all 240 × 320 pixels. The dataset was divided into training and test videos at a ratio of 9:1 based on the number of files. Watermark embedding and extracting were performed on a GPU. The used GPU was an NVIDIA Corporation GP102 [TITAN Xp] with 128 GB of memory.

4.2 Learning embedders and extractors

We trained the embedder and extractor using the training video dataset described in Sect. 4.1. The length of the embedding sequence N_c was set to 100 bits, and the number of training epochs was 100,000. In model training, 100 bits binary random number sequence was used as the embedding sequence. Eight original video frames were extracted from the training video data set and trained with a batch size of 4, along with the binary random number sequence. The loss used for training is the sum L of the mean squared error of RGB L_R, the mean squared error of YUV L_Y, the LPIPS perceptual loss L_L, the critic loss in Discriminator L_D and the watermark loss L_W.

$$L = \lambda_R L_R + \lambda_Y L_Y + \lambda_L L_L + \lambda_D L_D + \lambda_W L_W \tag{1}$$

where λ_R, λ_Y, λ_L, λ_D, and λ_W are the weight of the mean squared error of RGB, the mean squared error of YUV, the LPIPS perceptual loss, the critic loss in the discriminator, and the watermark loss, respectively. Here, in the calculation of the mean squared error of YUV L_Y, the squared errors of UV components were multiplied by 100. The watermark loss weight λ_W is fixed at 1.0. For the first 10,000 training epochs, the weights other than the watermark loss λ_W were set to 0 to improve the accuracy of extracting the watermark. After 10,000 epochs, the weights other than the watermark loss were linearly increased with each training epoch. The function to determine each video loss weight was based on the current number of training n as the following equations:

$$\lambda_R = \lambda_Y = \min\left(1.5 \times \frac{n}{20000}, 1.5\right) \tag{2}$$

$$\lambda_L = \min\left(0.3 \times \frac{n}{20000}, 0.3\right) \tag{3}$$

$$\lambda_D = \min\left(0.5 \times \frac{n}{20000}, 0.5\right) \tag{4}$$

The number of training epochs using only watermark loss (that is, 10,000 epochs in this experiment) was set as the number of times so that the accuracy of extracting the watermark for validation data in the training phase is achieved to be at least 95%, where the accuracy is concordance rate between the original embedding sequence and the extracted embedding sequence. In this experiment, we created three models with different losses and noise for training. The first model was trained with only the mean squared error of RGB and without adding noise for training (hereinafter called "RGB loss noiseless model"). The second model was trained with all video loss and without adding noise for training (hereinafter called "All loss noiseless model"). The third model was trained with all video loss and with adding noise for training (hereinafter called "All loss noisy model"). In the experiment, "RGB loss noiseless model" and "All loss noiseless model" are compared for investigating the effect of training loss, while "All loss noiseless model" and "All loss noisy model" are compared for investigating the effect of the noise for the training of the embedder and extractor.

In addition, a V-Net [15] based model was prepared for the comparison. Since the watermark could not be extracted in the as-is V-Net implementation, we used a model trained without Element Wise Sum, using only the RGB mean squared error, and without adding noise (hereafter called "V-Net RGB noiseless model"). The watermarked videos created with this model had a watermark extraction rate of more than 90% without re-shooting.

4.3 Experiments on embedding

One hundred videos were randomly selected from the test dataset described in Sect. 4.1 for embedding watermarks into them in this experiment. The selected videos were total 615 seconds in 320 × 240 pixels at 30 fps. The average duration of the 100 selected videos was 6.3 seconds, with a standard deviation of fewer than 3 seconds. The BCH coding

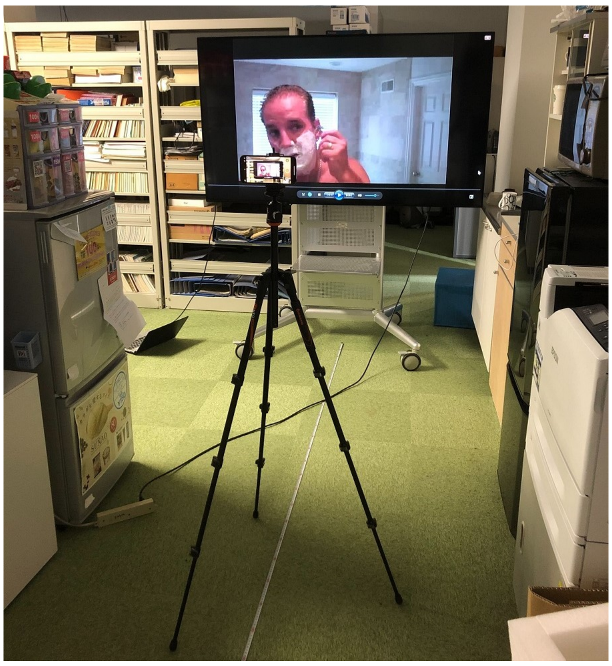


Fig. 6 Setting for re-shooting.

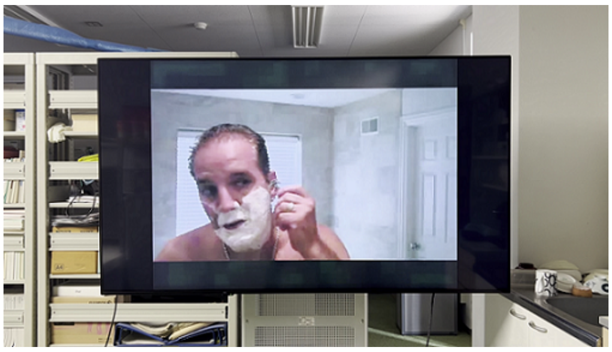


Fig. 7 Re-shot video.

parameters used for encoding the watermark consisted of 56 data bits and 40 checksum bits, and the remaining 4 bits were filled with zeros to achieve the embedding sequence of length 100 bits. The error correction capability is 5 bits.

4.4 Image quality evaluation of watermarked video

This section describes the image quality of the watermarked videos in Sect. 4.3. Figure 8 shows the original frame and the watermarked frames for "RGB loss noiseless model", "All loss noiseless model", and "All loss noisy model", and "V-Net RGB loss noiseless model". Each watermarked video was shown on the signage for the qualitative evaluation. Comparing the watermarked videos of "RGB loss noiseless model" and "All loss noiseless model", the watermarked video of "RGB loss noiseless model" was a flicker throughout the watermarked video than that of "All loss noiseless model". Comparing the watermarked videos of "RGB loss noiseless model" and "V-Net RGB loss noiseless model", "V-Net RGB loss noiseless model" showed more severe flicker noise than "RGB loss noiseless model". This flicker noise is caused by the reason that some parts of the 8 frames have more watermarks and some less. Flicker noise in "RGB loss noiseless model" and "V-Net RGB loss noiseless model" is a reddish or greenish noise that appears over the entire frame. Flicker noise in "All loss noiseless model" and "All loss noisy model" is like a mild desaturation, but in addition, "All loss noisy model" also shows noticeable black flicker noise only in the frames with mostly white backgrounds, as shown in Fig. 8. Table 1 shows the average PSNR and SSIM, where they are the objective image quality evaluation indices. Among the three 3D U-Net models, the average PSNR and SSIM are higher using all training losses than using only the RGB loss. Comparing the average PSNR and SSIM of"RGB loss noiseless model" and "V-Net RGB loss noiseless model", the score of "V-Net RGB loss noiseless model" was higher than that of "RGB loss noiseless model", however this is the opposite of the qualitative evaluation. In the V-Net model, one of the 8 frames was mainly changed for embedding, while in the 3D U-Net models, 8 frames were changed to a similar extent. Therefore, the average PSNR and SSIM of the V-Net RGB noiseless model were high though the watermarked videos include severe flicker noise.

Table 1 The average PSNR and SSIM for each model.

Model name	avg. PSNR [dB]	avg. SSIM
RGB loss noiseless model	37.4	0.961
All loss noiseless model	40.8	0.980
All loss noisy model	39.5	0.976
V-Net RGB noiseless model	40.4	0.974

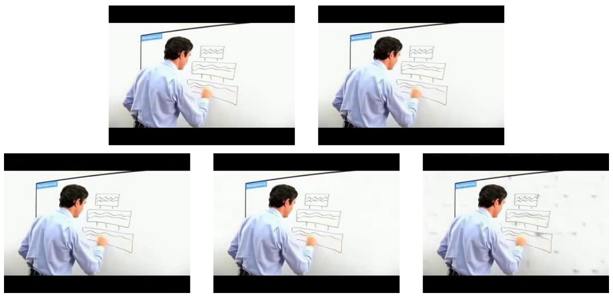


Fig. 8 Original frame (top left) and watermarked video frames using "V-Net RGB loss noiseless model" (top right), "RGB loss noiseless model" (bottom left), "All loss noiseless model" (bottom middle), "All loss noisy model" (bottom right).

Table 2 Results of reliability check and correct watermark extraction in 802 synchronizations when re-shooting.

Model name	Reliability check	Correct watermark
RGB loss noiseless model	563	563
All loss noiseless model	485	483
All loss noisy model	611	611
V-Net RGB noiseless model	420	420

Table 3 Average processing time for each model when re-shooting. (a) Average interval between correct watermark extractions [sec]. (b) Average processing time from synchronization to reliability check[sec].

Model name	(a)	(b)
RGB loss noiseless model	1.09	0.76
All loss noiseless model	1.27	0.91
All loss noisy model	1.01	0.91
V-Net RGB noiseless model	1.46	0.76

4.5 Experiments on extracting against re-shooting

The watermark extraction experiments were conducted on the re-shot videos of the watermarked videos of the selected videos described in Sect. 4.3. Figure 6 shows the setting for re-shooting the watermarked video. Since the proposed method does not implement automatic detection of a watermarked area in the re-shot video, the video was shot with a tripod so as to fix the coordinates of the watermarked area. The watermarked video was played on 70 inches signage (SHARP PH-H701) with a resolution of 3840×1920 pixels and was re-shot from a distance of 200 centimeters for about 10 minutes with an iPhone13 Pro fixed on a tripod. Figure 7 is an example of a frame of the re-shot video. The format of the video image stored in the iPhone 13 Pro after re-shooting is mov format with 1920×1080 pixels at 30 fps. In this experiment, the area detection of the watermarked video in the re-shot video was done manually to define the four corner points. The four corner points were (472, 139), (1455, 139), (1453, 868), and (482, 878) in the first frame of the re-shot video. They were projected to the image at coordinates (0, 0), (984, 0), (984, 738), and (0, 738). This projection is applied to all frames.

Finally, the watermark is extracted from the clipped and projected watermarked area in the re-shot video as described in Sect. 3.5. The same experiments were conducted on watermarked videos generated by the three models. Table 2 shows the number of times the reliability check was satisfied in 802 synchronizations (hereinafter called Reliability check), and the number of times the correct watermark after BCH error correction is extracted in 802 synchronizations (hereinafter called Correct watermark). Comparing "RGB loss noiseless model" and "All loss noiseless model", Correct watermark on "RGB loss noiseless model" is larger than that on "All loss noiseless model". Meanwhile, the image degradation of the watermarked video on "RGB loss noiseless model" is more visible than that on "All loss noiseless model". Comparing "All loss noiseless model" and "All loss noisy model", both Reliability check and Correct watermark on "All loss noisy model" is larger than those on "All loss noiseless model". This shows that adding noise during the training of the embedder and extractor is effective to achieve the robustness against re-shooting. Comparing "RGB loss noiseless model" and "V-Net RGB loss noiseless model", Correct watermark on "RGB loss noiseless model" is larger than that on "V-Net RGB loss noiseless model". Of the 100 videos used in the experiment, 14 videos could not be extracted at all in "V-Net RGB loss noiseless model" but could be extracted completely in "RGB loss noiseless model", for a total of 81 seconds. These videos included sports such as gymnastics, volleyball, and ice skating, as well as videos of musical instrument performances such as drums and sitar. Some of these videos had a lot of movement, some were small, some were from a fixed point, and some had a large background movement as the camera followed the subject. There were no common features, but there was an improvement in performance in a variety of scenes. This shows that 3D U-Net models are more robust against re-shooting than V-Net models.

Table 3 shows the average interval between correct watermark extractions and the average processing time from the synchronization to the reliability check against re-shooting for each model. As shown in Tab. 3, in three 3D U-Net models, RGB loss noiseless model can extract the correct watermark most frequently, while All loss noiseless model do least frequently. This shows that the three models have the robustness against re-shooting not for specific frames but for general frames although the effectivities are various. Comparing "RGB loss noiseless model" and "V-Net RGB loss noiseless model", Correct watermark on "RGB loss noiseless model" can be obtained more frequently than "V-Net RGB loss noiseless model". We can find the average processing time to obtain a correct watermark as the sum of the average interval between correct watermark extractions and the average processing time from synchronization to reliability check. Thus the average processing time for correct watermark extraction of "RGB loss noiseless model", "All loss noiseless model", "All loss noisy model", and "V-Net RGB loss noiseless model" are 1.85, 2.18, 1.92, and 2.22, respectively.

4.6 Experiments on extracting against re-shooting from various angles

Watermarked videos were re-shot from 30 degrees on the left and right sides to investigate the performance of the robustness against re-shooting from various angles. Table 4, Tab. 5, and Tab. 6 show the results for re-shot videos from 30 degrees on the left side and right side. Comparing Tab. 4 with Tab. 2, Correct watermark on "V-Net RGB loss noiseless model" decreased with angle, while Correct watermark on "RGB loss noiseless model" did not decrease. This shows that 3D U-Net model is more robust than V-Net model against re-shooting from various angles. The increase in some Correct watermarks seems to have occurred due to differences in the reflections of the background on the signage. Comparing Tab. 5 and Tab. 6, with Tab. 3, average processing time from synchronization to reliability check is related to the type of loss, not to the angles. Comparing "All loss noiseless model" and "All loss noisy model", adding noise during the training of the embedder and extractor is effective against perspective projection with re-shooting. It is noteworthy that "All loss noiseless model" can extract the correct watermark although the average interval of correct watermark extraction is relatively long. Therefore, "All loss noiseless model" is one candidate when the image quality is important.

4.7 Performance evaluation of watermarked video using StegaStamp

We compared the performance against re-shooting of the proposed method with that of StegaStamp. We embedded a watermark at each frame of the video using the trained model used in StegaStamp. The video and watermark were the same as in Sect. 4.3. The detection is also done manually. Moreover, the synchronization is not necessary and skipped because all frames are individually watermarked. Based on the difference between the proposed method and StegaStamp in our experiment, we can discuss the advantage of using temporal features. Since StegaStamp requires a 400×400 pixels image input, a 320×240 pixels frame was enlarged to a 400×300 pixels frame, and margins with a pixels value of 127 were added to the top and bottom. We extracted the watermark after the re-shot video under the same conditions as in Sect. 4.3. The number of all frames is 18446, and 17564 frames satisfied the reliability check. The correct watermark was extracted from 17560 frames. The average processing time to extract the correct watermark was 0.073 seconds which is the sum of the average interval of correct watermark extractions 0.035 [sec] and the average processing time of extraction 0.038 [sec]. The results confirm that the watermark can be extracted sufficiently by applying StegaStamp to each frame of the video.

Next, we evaluated the image quality of the watermarked videos using StegaStamp. The average PSNR and SSIM were calculated to be 30.7 dB and 0.878, respectively. Our method's average PSNR and SSIM values were higher than that of StegaStamp. In addition, comparing the PSNR and SSIM for each frame of the "All loss noisy model" and Stegastamp model, the PSNR and SSIM were higher for "All loss noisy model" at 18128 and 18423 frames, respectively, out of 18456 total frames. These results confirmed that the PSNR and SSIM were higher by distributing the information to be embedded over 8 frames using the temporal feature of the video. Figure 9 is the original video frame, the watermarked video frame using StegaStamp, and their residual frames. Figure 10 is residual frames using three models for comparison with Fig. 9. Other residual frames than "RGB loss noiseless model" have noise at the edges of the object, while the residual frame of "RGB loss noiseless model" has noise over the entire frame. This result shows that "All loss noiseless model" and "All loss noisy model" better use of temporal features and improved image quality due to the watermark being embedded in the relatively moving areas between frames and less noise in the stationary background. Also, this result shows that the cause for more flicker noises in "RGB loss noiseless model" seems to be noise over the entire frame. The watermarked video using StegaStamp showed a smoky afterimage.

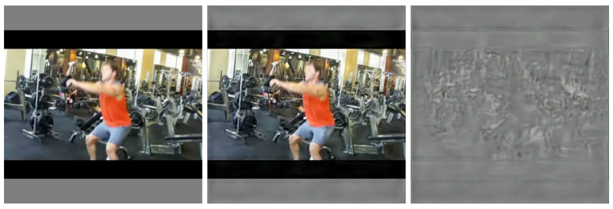


Fig. 9 Original frame (left), watermarked video frames using StegaStamp (center), and residual frames between the original video and watermarked video using StegaStamp (right).

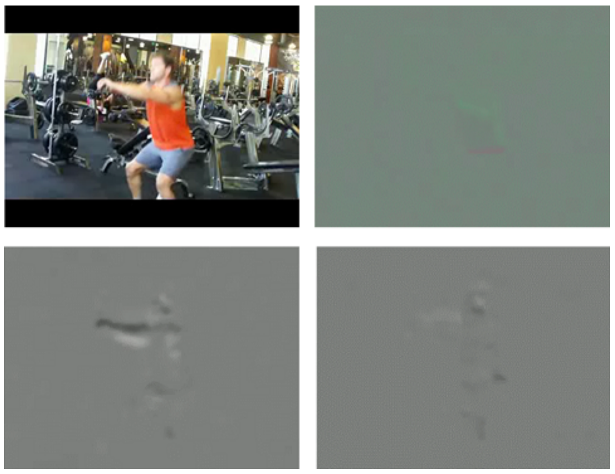


Fig. 10 Residual frames between the original video (top left) and the watermarked video using "RGB loss noiseless model" (top right), "All loss noiseless model" (bottom left), "All loss noisy model" (bottom right).

5. Conclusion

A new video watermarking method using 3D U-Net is proposed. The experimental results confirmed that the proposed method is robust against re-shooting and capable of embedding a watermark with 56 bits of information, which is larger than existing methods. The Challenge of the proposed method is to reduce flicker found by qualitative evaluation in image quality evaluation. In this experiment, the detection of watermarked areas from the re-shot video was done manually, so automatic detection of watermarked areas is a future issue. In addition, more practical experiments that consider image blurring due to camera shake and pixel misalignment from frame to frame are future issue. Another future challenge is to develop an application that runs the proposed method on an actual smartphone and conduct experiments on actual devices.

References

[1] J. Fridrich, Steganography in Digital Media, Cambridge University Press, 2014.

[2] S.S. Krishnan and R.K. Sitaraman, "Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs," IEEE/ACM Transactions on Networking, vol.21, no.6, pp.2001–2014, 2013.

[3] M. IWATA, N. MIZUSHIMA, and K. KISE, "Practical watermarking

Table 4 Results of reliability check and correct watermark extraction in 802 synchronizations when re-shooting from 30 degrees on the left and right side.

Model name	left side		right side	
	Reliability check	Correct watermark	Reliability check	Correct watermark
RGB loss noiseless model	594	592	628	628
All loss noiseless model	411	407	465	463
All loss noisy model	654	653	637	637
V-Net RGB noiseless model	249	247	367	365

Table 5 Average processing time for each model when re-shooting from 30 degrees on the left side. (a) Average interval between correct watermark extractions [sec]. (b) Average processing time from synchronization to reliability check[sec].

Model name	(a)	(b)
RGB loss noiseless model	1.04	0.79
All loss noiseless model	1.51	0.90
All loss noisy model	0.94	0.89
V-Net RGB noiseless model	2.49	0.79

Table 6 Average processing time for each model when re-shooting from 30 degrees on the right side. (a) Average interval between correct watermark extractions [sec]. (b) Average processing time from synchronization to reliability check[sec].

Model name	(a)	(b)
RGB loss noiseless model	0.98	0.79
All loss noiseless model	1.33	0.91
All loss noisy model	0.97	0.91
V-Net RGB noiseless model	1.68	0.80

method estimating watermarked region from recaptured videos on smartphone," IEICE TRANS. INF. & SYST., vol.100, no.1, Jan. 2017.

[4] C. Hui, S. Liu, W. Shi, F. Jiang, and D. Zhao, "Spatio-temporal context based adaptive camcorder recording watermarking," ACM Transactions on Multimedia Computing, Communications and Applications, vol.18, no.3s, pp.1–25, 2022.

[5] M. Tancik, B. Mildenhall, and R. Ng, "Stegastamp: Invisible hyperlinks in physical photographs," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," International Conference on Medical image computing and computer-assisted intervention, pp.234–241, Springer, 2015.

[7] X. Weng, Y. Li, L. Chi, and Y. Mu, "High-capacity convolutional video steganography with temporal residual modeling," Proceedings of the 2019 on International Conference on Multimedia Retrieval, pp.87–95, 2019.

[8] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," Advances in neural information processing systems, vol.28, pp.2017–2025, 2015.

[9] Ö. Çiçek, A. Abdulkadir, S.S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," International conference on medical image computing and computer-assisted intervention, pp.424–432, Springer, 2016.

[10] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," International conference on machine learning, pp.214–223, PMLR, 2017.

[11] R. Zhang, P. Isola, A.A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.586–595, 2018.

[12] K. Soomro, A.R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," CRCV-TR-12-01, November 2012.

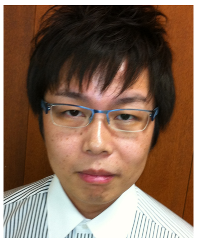
[13] A. Mishra, S. Kumar, A. Nigam, and S. Islam, "Vstegnet: Video steganography network using spatio-temporal features and micro-bottleneck.," BMVC, p.274, 2019.

[14] X. Shen, H. Yao, S. Tan, *et al.*, "Vhnet: A video hiding network with robustness to video coding," Journal of Information Security and Applications, vol.75, p.103515, 2023.

[15] F. Milletari, N. Navab, and S.A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," 2016 fourth international conference on 3D vision (3DV), pp.565–571, Ieee, 2016.

Takaharu Tsuboyama Currently enrolled at Osaka Prefecture University. His research interests include digital watermarking for video.

Ryota Takahashi received the B.S. degree in Engineering from Osaka Prefecture University in 2021. His research interests include digital watermarking for video.



Motoi Iwata received the M.E. and D.E. degrees in computer and systems sciences from Osaka Prefecture University, Osaka, Japan, in 1999 and 2005, respectively. From 1999 to 2016 and 2016 to 2022, he was an Assistant Professor / Associate Professor at the Graduate School of Engineering, Osaka Prefecture University, respectively. Since 2022, he has been an Associate Professor at the Graduate School of Informatics, Osaka Metropolitan University. His current research focuses on digital watermarking, data hiding, image retrieval, comic computing, and educational technology. He is a member of the Imaging Society of Japan, the Institute of Image Information and Television Engineers, ACM, and IEEE.

Koichi Kise received the B.E., M.E. and Ph.D. degrees in communication engineering from Osaka University, Osaka, Japan in 1986, 1988 and 1991, respectively. From 2000 to 2001, he was a visiting professor at German Research Center for Artificial Intelligence (DFKI), Germany. He is now a Professor of the Department of Core Informatics, Graduate School of Informatics, Osaka Metropolitan University, Japan. He received awards including the best paper award of IEICE in 2008 and 2022, the IAPR/ICDAR best paper awards in 2007 and 2013, the IAPR Nakano award in 2010, the ICFHR best paper award in 2010 and the ACPR best paper award in 2011. He worked as the chair of the IAPR technical committee 11 (reading systems), a member of the IAPR conferences and meetings committee. He is an editor-in-chief of the international journal of document analysis and recognition. His major research activities are in analysis, recognition and retrieval of documents, images and human activities. He is a member of IEEE, ACM, IPSJ, IEEJ, ANLP and HIS.