A Study of Embedding Operations and Locations for Steganography in H.264 Video

Andreas Neufeld and Andrew D. Ker

The Department of Computer Science, Parks Road, Oxford OX1 3QD, England.

ABSTRACT

This work studies the fundamental building blocks for steganography in H.264 compressed video: the embedding operation and the choice of embedding locations. Our aim is to inform the design of better video steganography, a topic on which there has been relatively little publication so far. We determine the best embedding option, from a small menu of embedding operations and locations, as benchmarked by an empirical estimate of Maximum Mean Discrepancy (MMD) for first- and second-order features extracted from a video corpus. A highly-stable estimate of MMD can be formed because of the large sample size. The best embedding operation (so-called F5) is identical to that found by a recent study of still compressed image steganography, but in video the options for embedding location are richer: we show that the least detectable option, of those studied, is to spread payload *unequally* between the Luma and the two Chroma channels.

Keywords: Video Steganography, Security Benchmarking, Embedding Operations, MMD

1. INTRODUCTION

The medium of compressed video has an unusual position in the information hiding literature. Motived by commercial applications, there is a large amount of work on digital watermarking for compressed video (see Refs. 1–4 for a survey), but very little on steganography.[5–10] This is despite a steganographer having many reasons – ubiquity, as well as huge potential capacity – to try hiding in video, and despite copious work on steganography in compressed still images.[11–13] In this paper we examine basic building blocks for steganography in compressed video: the embedding operation and the choice of embedding locations.

Our aim is to determine the best choice from a small menu of embedding operations and locations. In the context of steganography, this means the least detectable by statistical analysis (steganalysis). In this empirical study, we take some common embedding operations from compressed images, adapted as need be for the residuals of prediction blocks in H.264 video, and benchmark their security as measured by the information-theoretic quantity Maximum Mean Discrepancy (MMD)[14] over a corpus of videos. MMD has been advocated as a measure of steganographic security in Ref. 15. The MMD, and its rate of growth, is estimated using first- and second-order features extracted from slightly more than 26 hours of video, after simulation of the various embedding options. Because of the large sample size, the estimates are highly stable.

A related study, for still compressed images, is performed in Ref. 16 but using the error rate of a particular detector, instead of a theoretical quantity like MMD, as the benchmark. Our results show some commonalities between compressed images and compressed video, in that the embedding operation commonly known as F5[11] is superior to the alternatives. As with still-image steganography, one dare not alter zero coefficients by embedding, and it remains the case with compressed video that *only* the zeros should be avoided in order to achieve. We can also shed light on the best choice of colour channel for video steganography. Interestingly, the least detectable option is to spread payload *unequally* between the Luma and the two Chroma channels, although the advantage over Luma-only embedding is small. Our results are only as general as the corpus of videos from which they are produced, as the relevance of MMD to detectability, and in the context of a detector which consider first-

Further author information: (Send correspondence to ADK):

A. Neufeld is now with the Image and Pattern Analysis Group, University of Heidelberg, Germany. E-mail: neufeld@math.uni-heidelberg.de

A. D. Ker: E-mail: adk@cs.ox.ac.uk, Telephone: +44 1865 283530

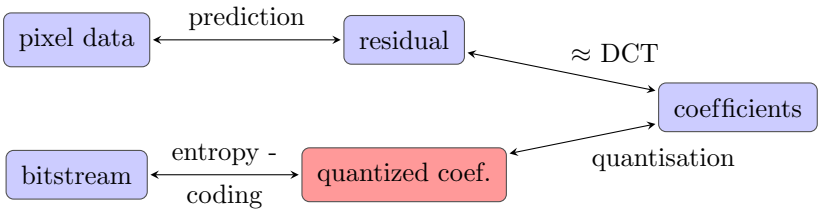


Figure 1: Video coding workflow. We modify only the quantized coefficients in the compressed video stream when simulating embedding.

and second-order histogram features, but provide at least some evidence to inform the design of better video steganography.

The structure of the paper is as follows. In section 2 we will briefly outline the structure of H.264 video, and the parts which we consider to be good candidates for steganographic embedding (prediction blocks only). We also describe the quantization process, which is richer than in still images because different parts of the video, and different parts of each frame, can be quantized differently. In section 3 we will describe the experimental study: the options for embedding operation and embedding location which we test, the implicit adversary modelled by a feature vector, and the estimation of MMD. The results appear in section 4 and we draw conclusions in section 5.

2. VIDEO CODING

A video is a collection of images, called *frames*, shot at a constant time interval. A *codec* is a coder/decoder pair of which the decoding process is uniquely specified. There can be several encoders that produce different bitstrings when encoding the same input video but there is only one decoder available.[17] Most encoders offer a wide range of settings to trade off quality against compressed file size and encoding speed.

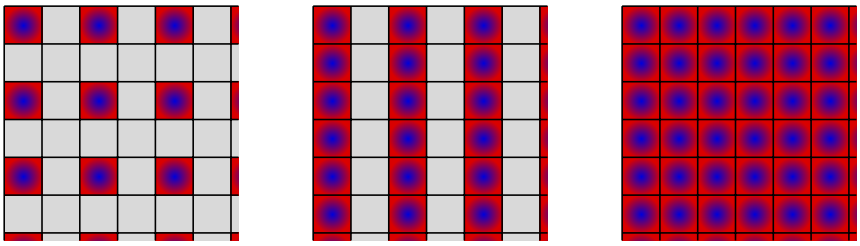
There are four steps involved in encoding a video. The frames are divided into blocks and each block is predicted based on surrounding blocks ore block in past and/or future frames. The prediction is subtracted from the actual pixel data and the residual is transformed using an integer approximation of the DCT. The DCT coefficients are quantized and finally entropy coded into a bitstream.

The decoding process is precisely the inverse operation. The encoder needs to make sure that the decoder makes the same predictions as the encoder, therefore the encoder uses decoded blocks for prediction only. Figure 1 shows the general video coding workflow.

2.1 Colour Channels

There are three channels present in an H.264 video stream: Y (Luminance or Luma), Cr (red Chrominance or Chroma) and Cb (blue Chroma). Cr and Cb are also called U and V.

The human eye is less sensitive to Chroma than it is to Luma, and this motivates to sample Chroma information at a lower resolution. There are three different sampling patterns available in H.264: 4:2:0, 4:2:2 and 4:4:4,



(a) 4:2:0 (b) 4:2:2 (c) 4:4:4

Figure 2: Different sampling patterns. Each cell represents a luma sample and coloured cells represent chroma samples.

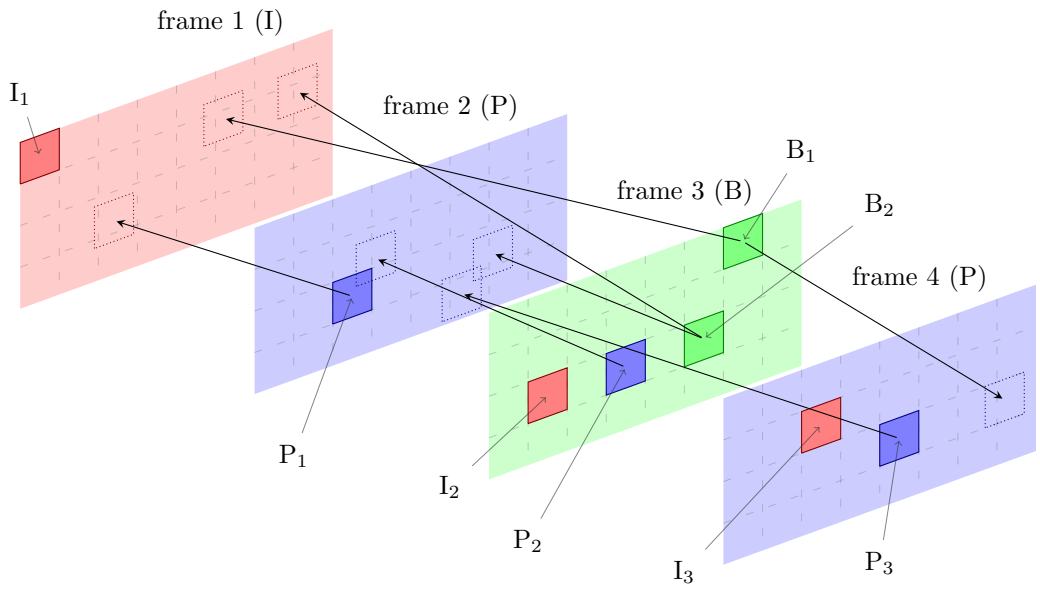


Figure 3: Inter prediction. Showing I-, P- and B-macroblocks. The P macroblocks in use frame 1 and frame 2 as reference frames. Frame 3 has a future reference on frame 4, hence frame 4 has to be decoded before frame 3. The macroblock P_3 is not allowed to reference frame 3 even though frame 3 is displayed before frame 4.

illustrated in Figure 2. Our video corpus consists of the most commonly-used option, 4:2:0 sampling, which means that the Chroma information is stored at half the resolution of Luma.

2.2 Prediction

H.264 partitions a frame into 16×16 pixel *macroblocks* which can then be further partitioned down to size 4×4 pixels, and each partition receives its own *prediction mode*. The pixel values are subtracted from the prediction and the *residual* of each 4×4 block is transformed using an integer approximation of the DCT.

There are different frame types available in H.264: I-, P- and B-frames. I-frames are independent pictures using *intra-prediction* (prediction referencing the same frame) only. P- and B-frames use surrounding frames as reference anchors to make accurate predictions of the current frame. P-frames use a single *motion vector* pointing on a past frame. B-frames use two motion vectors and are allowed to reference both past and future frames. This implies that frames are not encoded in display order. Each macroblock is type I, P or B as well and a frame can be divided into horizontal slices which are again of type I, P or B. Slices can be decoded intependently, allowing the decoder to use parallel hardware efficiently. Figure 3 illustrates the different macroblock types.

P-macroblocks use the point they reference to as prediction where coordinates have half- or quarter-pixel precision. B-macroblocks combine both reference frames to a prediction using a weighted average or other combination.

There is a special kind of I-frame, the *IDR (Instantaneous Decoder Refresh)* frame. It signals to the decoder that it can release all memory used for reference frames. Therefore motion vectors cannot pass an IDR frame. IDR frames divide the video into parts the decoder can easily switch between.

2.3 Transformation and Quantization

Independently of the macroblock type or prediction mode the residual of each 4×4 pixel block gets transformed using an integer approximation of the DCT. (An 8×8 transform is available as an option as well but the 4×4 transform is much more common, and our corpus is created entirely using the 4×4 transform). In addition to the core transform another transform is used for the DC coefficients of the Chroma blocks, as shown in Figure 4

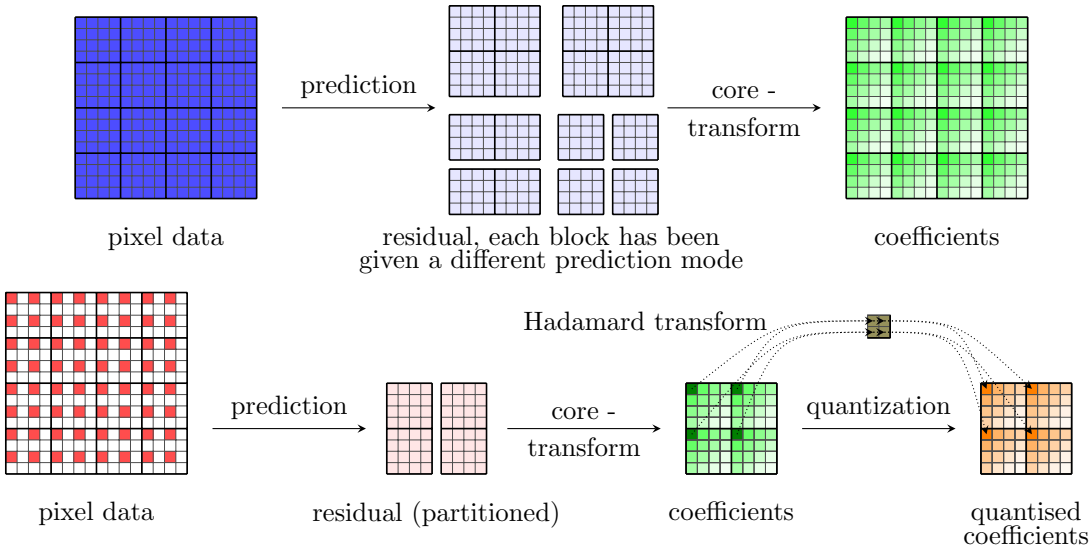


Figure 4: Luma (above) and Chroma (below) transforms, quantization is omitted for Luma. Lower intensity indicates smaller average magnitude of coefficients.

The level of quantization is determined by the Quantisation Parameter (QP). Each 4×4 block has its own QP. The quantization step size s is approximately exponential in QP: it doubles as QP increases by 6.[17] QP can take values from 0 to 51: when QP is 0 we have $s = 0.625$; at QP= 24, $s = 10$; when QP is 51, $s = 224$.

On input coefficient x we compute quantized coefficient $y = \text{round}(\frac{x}{s})$. When decoding the video x will be approximated as $x' = y \cdot s$. Pixel values are integers, and H.264 uses an integer approximation to the DCT as core transform. The Hadamard transform, used for DC coefficients, is an integer transform as well. Therefore quantization in H.264 takes integers as input and produces integers as output.

There are two entropy coding modes available: *Context Adaptive Variable Length Coding (CAVLC)* and *Context Adaptive Binary Arithmetic Coding (CABAC)*. CAVLC picks the codeword table based on the number of non-zero coefficients in surrounding blocks. CABAC estimates a probability distribution over coefficients based on previously observed data to offer better codewords. We use CAVLC in all our experiments to avoid a desync in the decoder.

3. EXPERIMENTS CONDUCTED

3.1 Embedding Operations

We simulate video steganography at the lowest level, by performing random embedding operations on the residuals in the encoded video stream. Our embedding operations are taken from the literature on still-image steganography: the traditional Least Significant Bit (LSB) Replacement, LSB Matching (LSBM), and the F5 embedding operation.[11]

It is folklore wisdom that, for still JPEG steganography, one should not change a zero coefficient, and there may be merit in avoiding changes to other coefficients of small absolute value. So we implemented different varieties of the LSBR, LSBM, and F5 embedding operations with different "thresholds", as illustrated in Figure 5. For LSBM the options are to refuse to change zeros, or coefficients in the range $-1..1$. For F5 we also tried avoiding changes to $-2..2$. For LSBR we can avoid changes to 0 and 1 (in still images this is the "JSteg" embedding operation) or coefficients in the range $-2..3$.

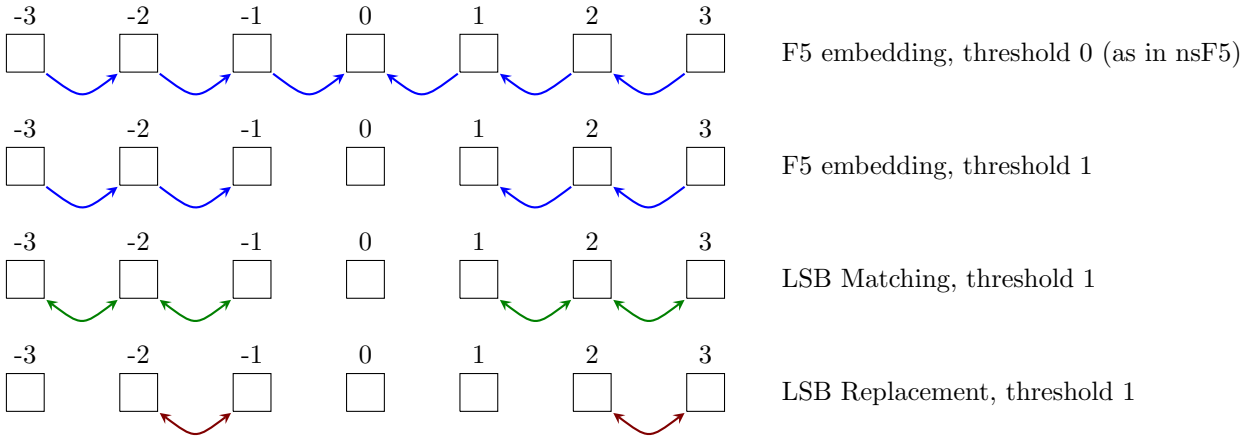


Figure 5: All embedding methods that are simulated. Threshold 2 for each method is included as well.

3.2 Embedding Locations

Having chosen the type of changes to make, we should then decide where they take place. There are a number of options:

Slice type There are three types of frames available: I-, P- and B-frames, similarly there are three types of slices and macroblocks. Different slice types may be present in a frame and different macroblock types may be present in a slice. We make decisions based on the slice type. We do not use I slices since these are very rare. P-slices are the most frequently appearing: there more P-slices than B-slices since B-slices reference two P- or I-frames. B-blocks store two vectors while P-blocks store only one, therefore residuals in B-blocks must be much smaller than residuals in P-blocks and the overall space needed to store B-blocks must be less than the cost of a P-block (otherwise the encoder would not have chosen to use a B type). Therefore we may expect the residuals of P- and B-blocks to have different shapes.

Except for a very small number of cases B-slices are accompanied by P-slices next to them, therefore we have two options: to embed in P-slices or both B- and P-slices.

Channel In most literature about image steganography only the luminance is used for embedding. We are interested whether chrominance could be better suited for embedding, or a combination of both channels.

DC coefficient The $(0, 0)$ DCT mode is also called the DC coefficient. It is the most visually perceptible coefficient and also most frquently non-zero. We will test one variety of embedding which excludes changes to DC coefficients, and one which uses the DC coefficient in the same way as any other.

3.3 Feature Vector

Our aim is to find the least detectable embedding operations and best suited embedding locations. It is practically impossible to give a definitive answer, so we try to be as general as possible.

Our opponent, the steganalyst, will extract features from our videos and use these for classification. We do not know exactly which features they will use, but we assume that they are guided by designs of steganalysis in still images, using features which count occurrencs and higher-order histograms of the quantized coefficients. In our case we will concatenate both histograms and 2nd-order co-occurence histograms from both Luma and Chroma coefficients. If our embedding is not detectable with our features, it will not be detectable with any subset either.

Note that the features are from DCT transformed *residuals*, since the P- and B-blocks store what is left after prediction. This means that the coefficients are effectively already filtered, and there is no need to consider differences between/within blocks in the way that is done for JPEG features.[18]

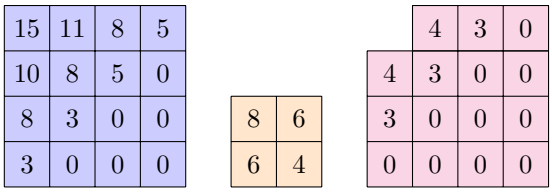


Figure 6: The ranges we used in our experiments. Value r indicates that histogram range $[-r, r]$ was included.

We look at the *histogram* of coefficient in each 4×4 pixel block. There is no reason to include the entire histogram, because very high values will occur rarely (if at all), and give little or no information about the presence of payload, while increasing the dimensionality of the data. Therefore we only count coefficients in the range $[-r, r]$, for values of r which capture most of the information in the object: for highly quantized coefficients, smaller values of r are sufficient, and in fact we discard information from the highest frequencies altogether. Figure 6 shows the ranges we used in our experiments, which were determined from manual inspection of the data.

We include *co-occurrences* of coefficients. The number of co-occurrences we measure depends on the maximum range that appears in the histograms. Therefore the co-occurrences may contain coefficients that were not included in the histograms. Histograms and co-occurences are extracted separately for each channel, and separating Chroma DC from Chroma AC.

The last part of our feature vector is the co-occurrence between the two Chroma channels ('U×V'). We count co-occurences for each coefficient separately. It is possible that we do not have both Chroma channels available in a block, in which case these blocks are ignored for U×V features. Thus there are four parts to our feature vector: Luma, Chroma DC, Chroma AC, and U×V. The total length of the feature vector is 1882.

Recall that each macroblock can have a different QP, which means that its histogram bins represent different ranges of unquantized coefficients, and this means that feature counts from different QP blocks will have a different meaning. This is a similar problem to feature mismatch from images with different quantization factors in JPEG files, with the additional difficulty that the variation can be within each frame. To eliminate this effect, we first tried concatenating features from all different values of QP, but the dimensionality was simply too high. Instead, we fixed on the most common QP and performed steganalysis only on those blocks. We must acknowledge the (weak) possibility that this fails to capture certain properties of the covers.

One feature vector is extracted per section between two IDR frames: in our video collection this means an average of 115.25 frames form each feature vector, but the number of frames between IDR frames does differ widely. Scaling of the features (subsection 3.5) standardizes the scales of the histograms, so the number of frames (and macroblocks within those frames) does not affect the magnitude of the feature vector.

3.4 Security Benchmark

The security of an embedding operation is its ability to avoid detection. There are different detectors for video available. One possibility would be to train a classifier on features extracted from the DCT coefficients.[19,20] Another approach is to detect peculiarities in the processing history of an image or video.[19,21] Alternatively, the noise of a video can be explicitly modelled and then classified using its entropy.[22] Informally speaking, modified videos tend to have noise which is more random than it should be, and this can be detected.

We are concerned that our conclusions should be as general as possible. Benchmarking against a particular detector gives easy-to-interpret results (detection accuracy) but runs the risk of being dominated by peculiarities of the detector(s) used rather than the fundamental security of the embedding operations. In contrast to the still-image approach of Ref. 16, therefore, we try to take an information-theoretic approach. We assume that clean and modified feature vectors follow different probability distributions, and want to make an empirical estimate of how "different" these distributions are.

3.4.1 The Kullback-Leibler divergence (KL-D)

The *Kullback-Leibler divergence (KL-D)* is advertised in Ref. 23 as measure of steganographic security. It has the following definition:

$$D_{KL}(p_c, p_s) = \sum_x p_c(x) \log \frac{p_c(x)}{p_s(x)}. \tag{1}$$

The KL-D is an information theoretic distance measure on probability distributions and gives bounds on detectability, including:

$$D_{KL}(p_c, p_s) < \epsilon \quad \Rightarrow \quad P_{\text{missed det.}} > 2^{-\epsilon} \tag{2}$$

if there are no false positives and the log in (1) is to base 2.

However, the KL-D has a serious disadvantage: it is very hard to estimate empirically. For example, the work of Ref. 24 shows that an enormous corpus is required to estimate KL-D (in this case for uncompressed images) for feature vectors of dimensionality 9. We modelled clean and stego sets as Gaussians (already a strong assumption) and tried to estimate the KL-D using the following expression:

$$D_{KL}(\mathcal{N}_0 \| \mathcal{N}_1) = \frac{1}{2}\left(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) - \ln\left(\frac{\det \Sigma_0}{\det \Sigma_1}\right) - d \right), \tag{3}$$

where d is the dimensionality, but the covariance matrices turned out to be singular. Another KL-D estimate based on the k-nearest neighbourhood is shown to be highly unstable in Ref. 25.

3.4.2 The Maximum Mean Discrepancy (MMD)

The MMD was designed to solve the Two-Sample Problem, that is to decide if two sets X and Y are sampled (indipendently and identically distributed) from two different distributions.[14] In Ref. 15 it was proposed to use MMD, instead of measures like KL-D, as a fundamental benchmark for steganography.

The MMD has theoretical foundations, it can be linked to Parzen Window estimates.[25] In addition to this, it is highly stable, fast in computation and convergence, and works well with large dimensions. The disadvantage is that it does not have the same fundamental connection with detectability as does KL-D, but it should still be more universal than the behaviour of any particular detector.

The MMD of two distributions p, q and function class $\mathcal{F}$ is defined to be:

$$MMD(\mathcal{F}, p, q) = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{x \sim q} f(x) \right). \tag{4}$$

In case of two finite sample sets X and Y of size N this is equivalent to:

$$MMD(\mathcal{F}, X, Y) = \sup_{f \in \mathcal{F}} \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i) - \frac{1}{N}\sum_{i=1}^{N} f(y_i) \right). \tag{5}$$

We restrict the function class to *reproducing kernel Hilbert spaces (RKHS)*. In an RKHS with kernel $k(\cdot,\cdot)$, function evaluation can be written as $f(x) = \langle k(x,\cdot), f \rangle$.[14] We restrict the kernel function class to Gaussian kernels since these have been shown to perform well when used in SVMs.[25] The Gaussian kernel is defined as:

$$k(x, y) = \exp(-\gamma \|x - y\|^2). \tag{6}$$

The final MMD estimate is found using the following unbiased empirical estimate which is based on the U-Statistic:

$$MMD(X, Y) = \sqrt{\frac{1}{N(N-1)} \sum_{i \neq j} k(x_i, x_j) - 2k(x_i, y_j) + k(y_i, y_j)}. \tag{7}$$

We see that the computation of MMD is quadratic in the number of feature vectors, while an SVM would require cubic running time and a grid search for hyper parameters.[15]

A useful property of MMD is that it scales locally linearly with changes, such as payload, as long as the changes as reasonably well-behaved.[26] In the experiments we will estimate MMD for a variety of small payload sizes, and make a least-squares estimate for the rate at which it increases with payload, reducing the benchmark to a single number (MMD per bpnc).

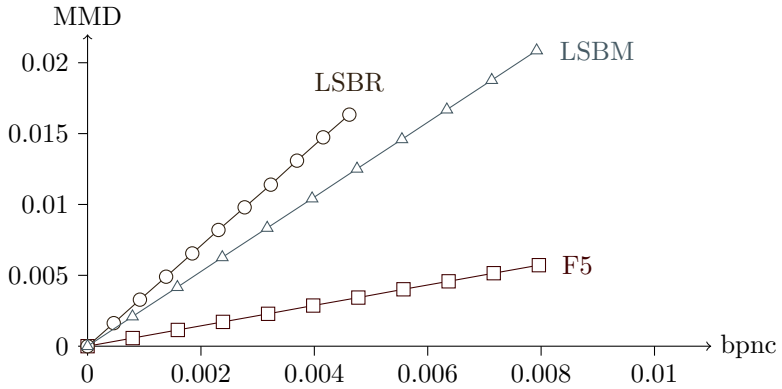


Figure 7: MMD as a function of payload (bpnc). In this graph, the embedding operations exclude 0 (LSBR also excludes 1). Embedding was done in Luma AC and DC coefficients in 3000kbit/s video. We use the slope of the graphs for our benchmark.

3.5 Normalization and Hyperparameters

The features must be *rescaled* before the benchmark is computed. The minimum and maximum values in the clean set are found for each element of the feature vector (because they are counts, the minimum turns out to be zero for each feature) and features are linearly scaled so that the maximum element is mapped to one. This ensures that all features have equal impact on the resulting MMD value.

We set $\gamma = \eta^{-2}$ where η is the median of L_2 distances of all pairs of clean feature vectors. This traditional choice is motivated by the desire for the exponents in $k(x, y)$ to be close to unity.

3.6 Cover Source

We use 16 deinterlaced DVD movies, more than 26 hours of video: DVDs are encoded at up to 5000kbit/s in the mpeg2 format, at a resolution of 1024×576. They were ripped with `mencoder` and transcoded into the H.264 format with `x264`.

We use two encoding bitrates to compare low and high quality encodings, 500kbit/s and 3000kbit/s. The resolution was unchanged. Both cases produced approximated 20000 feature vectors.

4. RESULTS

Our experiments are aimed at answering the following questions:

1. Which embedding operation is least detectable?
2. Which locations are best suited for embedding?
   (a) Channels: Luma, Chroma or both?
   (b) Frame types: P-slices or P- and B-slices?
   (c) Coeffcients: Include the DC coefficient?
3. Do we prefer low- or high-quality videos for embedding?

We simulate the embedding in all possible combinations of embedding operations and locations, applying the embedding operation with probability $p = 0.001, 0.002, \ldots, 0.01$, computing the bits per nonzero coefficient $b(p)$ in each case, and estimating the MMD $f(p)$ via (7). Figure 7 shows the MMDs for our embedding operations in Luma (AC+DC) coefficients. We fit a least-squares approximation of $f(p) = a \cdot b(p)$. A high value of a indicates that MMD grows more quickly with payload: the embedding operation is more detectable. Table 1 shows the

Embedding operation	Unchanged values	P-slices (20577 vectors)						P+B-slices (20584 vectors)					
		Luma		*Chroma*		$L+C$		*Luma*		*Chroma*		$L+C$	
		AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC
F5	0*	1.18	**0.72**	3.62	12.05	1.07	2.55	1.28	0.79	4.06	16.02	1.23	3.72
	-1..1	2.64	1.48	7.35	5.45	2.53	1.47	2.93	1.54	8.48	5.49	2.81	1.53
	-2..2	5.14	3.00	13.83	10.96	4.98	2.89	5.47	3.31	15.21	12.71	5.27	3.26
LSBM	0	4.49	2.63	8.80	6.84	4.22	2.50	4.31	2.42	10.00	6.68	4.08	**2.39**
	-1..1	6.34	3.65	16.95	14.40	6.06	3.63	6.51	3.93	19.45	17.19	6.22	4.04
LSBR	0,1	6.20	3.54	12.06	9.61	5.84	3.42	5.19	**2.77**	13.44	9.55	4.93	3.08
	-2..3	8.63	5.07	21.62	20.44	8.63	5.05	10.17	6.21	25.60	24.62	9.95	6.16

Embedding operation	Unchanged values	P-slices (20262 vectors)						P+B-slices (20267 vectors)					
		Luma		*Chroma*		$L+C$		*Luma*		*Chroma*		$L+C$	
		AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC	AC	AC+DC
F5	0*	1.63	**0.97**	3.13	4.18	1.61	1.74	1.67	1.02	3.00	4.81	1.64	2.02
	-1..1	2.99	1.29	11.07	2.88	2.73	1.24	3.18	1.29	10.55	2.93	2.89	1.25
	-2..2	6.57	2.97	3.89	6.76	6.43	2.95	6.83	2.85	7.55	6.76	6.60	2.82
LSBM	0	4.41	1.87	9.76	3.46	4.31	**1.75**	4.64	1.94	9.59	3.71	4.53	1.86
	-1..1	8.30	3.64	14.52	10.15	8.04	3.50	8.41	2.93	13.88	8.12	8.16	2.82
LSBR	0,1	6.46	2.70	12.98	4.40	6.47	**2.42**	7.04	2.88	10.14	4.27	7.18	2.46
	-2..3	15.35	6.61	6.83	10.62	14.60	6.86	26115	375.4	762440	15167	25842	371.7

$*$ corresponds to the traditional F5 embedding operation.

Table 1: MMD growth rates (above: 3000kbit/s video, QP=20; below: 500kbit/s video, QP=28)

growth rates for all embedding operations in the high- and low-quality video corpuses. For high-quality videos the features were extracted from blocks with QP=28; from low-quality videos with QP=20.

As well as computing the least-squares fit for a, we can also use standard techniques to estimate confidence intervals for it. Because our corpus is so large, with tens of thousands of feature vectors, we found that the MMD estimates were highly stable, to the point where the error is too small to display. For example, the smallest value in Table 1 is 0.72, and the standard error of this estimate is $3 \cdot 10^{-4}$. The largest value in the top half of the table is 25.60, and the standard error of this estimate is 0.63.

We draw the following conclusions from these tables:

Embedding operation We can see that LSBR is the most detectable and F5 is the least detectable method among those tested. A similar observation was made in Ref. 16 for still compressed images. We also observe that shortening the unchanged interval (using coefficients with low absolute value, except for zero) leads to a lower detectability in most cases, the only exception being F5 when embedding in Chroma AC+DC.

Embedding location

Channel Chroma-only embeddings are most detectable across all embedding operations. In some cases Luma performs better than L+C, and sometimes it does not. In the next experiments we will vary payload balance between the two channels to get a clearer picture of the behaviour in between.

Slice type We did not test I-slices, since they are rare and less interesting for steganography. The majority of slices are P-slices, also P-slices are larger than B-slices in general, a video file mainly consists of P-slice data, this makes P-slices most attractive for embedding. In our experiments we see that including B-slices for embedding increases detectability. B-slices have smaller resuduals than P-slices in general, this may give an intuitive reason for increased detectability in B-slices.

Coeffcients In most cases including the DC coefficient decreases detectability. But for the best performing operation, F5 embedding, the Chroma DC coefficient increases detectability.

Video quality LSBM and LSBR tend to be less detectable in low quality videos. F5 achieves minimum detectability in higher quality encodings, out-performing low bitrate LSBM and LSBR. However, this observation is particularly limited by our use of data from a single QP factor and one must bear in mind that it could be an artefact of the QP choice for the two bit rate videos.

4.1 Embedding Channel

In Table 1, the locations were either the Luma or Chroma channels only, or equal use of all of them, in the sense that the embedding *probability* was the same in all channels. (This does not mean that equal data would be hidden in both channels, because the Luma channel contains more nonzero coefficients to work with). We examine further how to balance payload between the Luma and Chroma channels by embedding with different probabilities in the different channels.

Figure 8 shows the resulting graph: on the left of the graph the payload is entirely in the Luma channel, on the right entirely in the Chroma channels. We have best performance for embedding in Luma only. But in Table 1 we saw that detectability of Chroma DC embeddings was very high, therefore we will drop this and embed in Luma AC+DC and Chroma AC. The resulting graph is shown in Figure 9. This embedding operation achieves lowest detectability among those tested.

In our least detectable scheme, there is little variation in the detectability, except for embedding almost entirely in Chroma, where MMD growth is much higher. We introduced a logarithmic scale to enhance visibility of line shape. With 3000kbit/s videos, we get a minimum at approximately equal use of Luma and Chroma chanells, and with 500kbit/s an embedding ratio of $0.1 : 1$ Luma:Chroma is the global minimum. Finding the minimum would be rather delicate, however, because detectability rapidly increases if the Chroma channel is used a fraction too much.

We can see that a combination of Luma and Chroma embedding achieve best performance. The benefit compared with using only Luma is small, but it is safe to include the Chroma AC channel (as long as not exclusively), and gives a slight improvement.

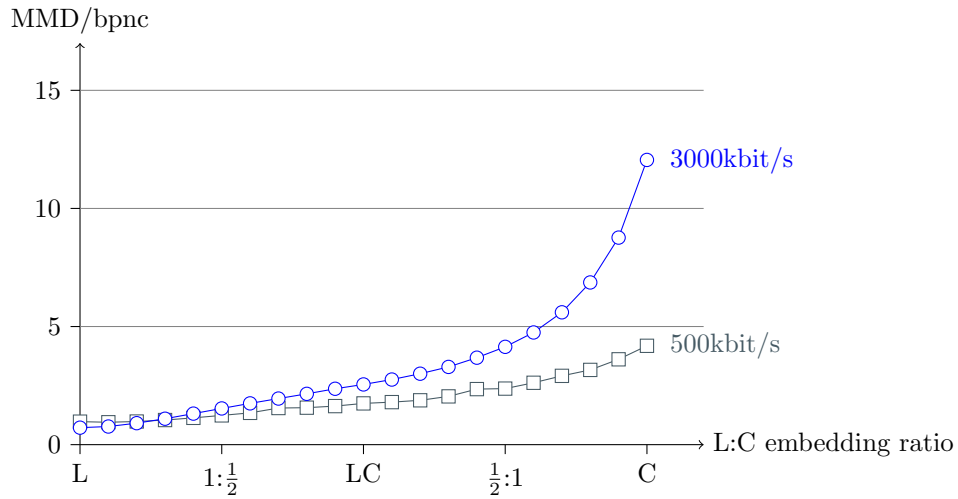


Figure 8: Detectability as payload is varied between Luma and Chroma channels, using the F5 embedding operation in AC+DC coefficients.

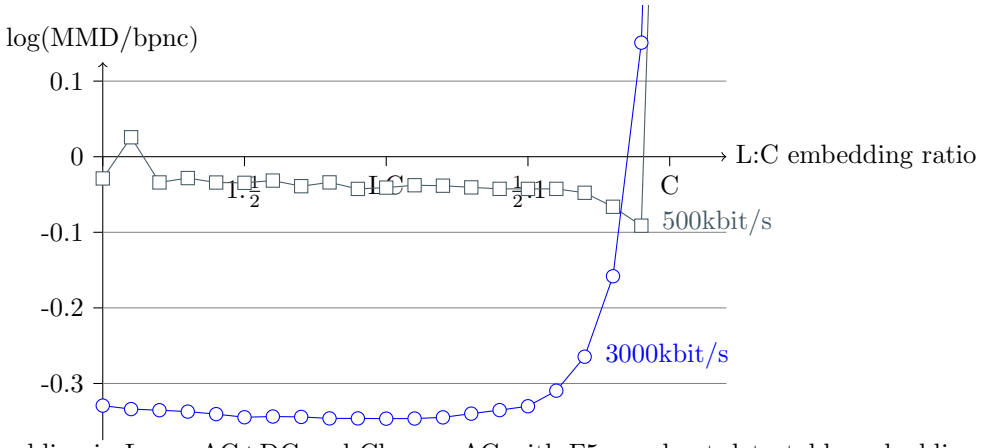


Figure 9: Embedding in Luma AC+DC and Chroma AC with F5, our least detectable embedding scheme. The y-axis is on a log-scale to enhance the details of the graph. You can find the same graph for 3000kbit/s on a linear scale in Figure 10.

4.2 Subsets of features

We observe that Chroma-only embeddings are most detectable, and this also holds when the DC coefficient is not included. We ask the following question: does the detection power come from the use of the U×V features? To answer this, we performed the same experiments with subsets of the full feature vector, using a) only histogram information ('H'), b) co-occurences but excluding U×V features ('HC'), and c) the full vector ('HCU'). Figure 10 shows the detectability of our least detectable embedding scheme (F5 operation in P-slices, Luma DC+AC and Chroma AC) with these sets of features. MMD values of different dimensions cannot be compared directly, but we can see that Chroma-only embeddings are vulnerable to extra information given by richer feature sets.

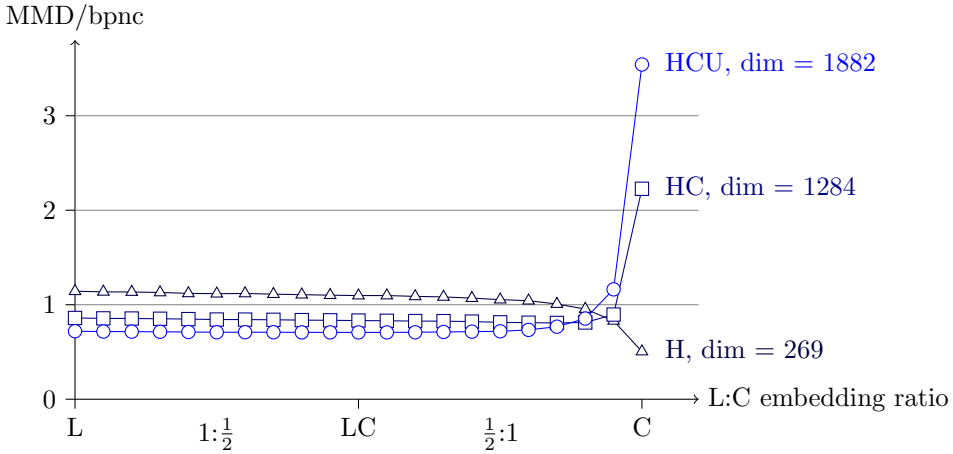


Figure 10: MMD growth of our least detectable embedding scheme for 3000k video with varying subsets of features. 'H' are the histograms, 'C' the co-occurrences and 'U' the U×V features. We cannot compare values between the graphs directly since the different feature subsets are of different dimensions but we can compare the shapes of graphs.

5. CONCLUSION

We have adapted some commonly used algorithms for image steganography to the H.264 video codec, extracted first and second order DCT features from a large video collection and benchmarked different embedding operations and locations. As benchmark we used the Maximum Mean Discrepancy (MMD), which has recently been proposed as benchmark for image steganography.

For the embedding location, we used different macroblock types, both Luma and Chroma channels with varying payload and included or excluded the first (DC) coefficient. For the embedding operations, we introduced an embedding threshold, which allowed us to vary the range of unchanged coefficients. Experiments were conducted on low- and high-quality encodings of DVD movies.

Our results show that the different embedding methods behave similarly to still compressed (JPEG) images: overall LSBR is more detectable than LSBM, which is more detectable than F5. Across all tested methods, excluding small coefficients worstens performance, but zeros are still excluded. In some cases, Chroma-only embeddings do not show the same behaviour, but these are unstable in other apects as well. We get lowest detectability using the F5 embedding operation (presumably implemented as nsF5[27]) in P-macroblocks of high quality videos, using all nonzero coefficients and balancing payload with a slight favour towards Chroma between the Luma and the Chroma AC channels.

The MMD measures detectability as difference of the estimated distribution of clean and stego sets. It has a number of drawbacks: it is difficult to interpret the results concretely, and it is questionable whether MMD truly reflects the performance of (optimal) detectors. We leave it as future work to benchmark against one or more particular detectors to see whether concrete results confirm our conclusions.

Our features are bound to part of the H.264 codec: motion vectors or QP deltas could be used as well for embedding, but we use DCT coefficients only. H.264 is a widespread codec, but not the only one available on the internet. Its successor H.265 (High Efficiency Video Coding) is announced for this year, which will again increase the choice of video codecs. It is desirable to work on features extracted from the video content itself rather than syntax elements of the codec, so that modifications possibly could be detected even after recompression into another format.

This work, however, is designed to draw attention to the fundamental detectability of different embedding operations. Not all choices are equal, and it is premature to design a steganographic system before choosing the best types of location and the optimal embedding operation.

ACKNOWLEDGMENTS

The work was performed as part of the first author's undergraduate dissertation at Oxford University Department of Computer Science.

REFERENCES

[1] Hartung, F. and Girod, B., "Watermarking of uncompressed and compressed video," *Signal Proc.* **66**(3), 283–301 (1998).

[2] Doërr, G. and Dugelay, J.-L., "A guide tour of video watermarking," *Signal Proc.: Image Communication* **18**(4), 263–282 (2003).

[3] Zhang, J., Ho, A. T. S., Qiu, G., and Marziliano, P., "Robust video watermarking of H.264/AVC," *IEEE Trans. Circuits Syst. II, Exp. Briefs* **54**(2), 205–209 (2007).

[4] Mansouri, A., Aznaveh, A. M., Torkamani-Azar, F., and Kurugollu, F., "A low complexity video watermarking in H.264 compressed domain," *IEEE Trans. Info. For. Sec.* **5**(4), 649–657 (2010).

[5] Wang, C.-M., Wu, N.-I., Tsai, C.-S., and Hwang, M.-S., "A high quality steganographic method with pixel-value differencing and modulus function.," *J. Systems and Software* **81**(1), 150–158 (2008).

[6] Westfeld, A. and Wolf, G., "Steganography in a video conferencing system," in [*Proc. 2nd Information Hiding Workshop*], 15–17, Springer (1998).

[7] Xu, C., Ping, X., and Zhang, T., "Steganography in compressed video stream," in [*International Conference on Innovative Computing, Information and Control*], 269–272, IEEE Computer Society, Los Alamitos, CA, USA (2006).

[8] Liu, B., Liu, F., Yang, C., and Sun, Y., "Secure steganography in compressed video bitstreams," in [*Proc. 3rd International Conference on Availability, Reliability and Security*], *ARES '08*, 1382–1387, IEEE Computer Society (2008).

[9] Robie, D. L. and Mersereau, R. M., "Video error correction using steganography," *EURASIP J. Appl. Signal Process.* **2002**(2), 164–173 (2002).

[10] Sherly, A. P. and Amritha, P. P., "A compressed video steganography using TPVD," *Int. J. Database Management Systems* **2**(3) (2010).

[11] Westfeld, A., "F5-a steganographic algorithm," in [*Proc. 4th Information Hiding Workshop*], *LNCS* **2137**, 289–302, Springer (2001).

[12] Fridrich, J., Pevný, T., and Kodovský, J., "Statistically undetectable JPEG steganography: Dead ends challenges, and opportunities," in [*Proc. 9th ACM Workshop on Multimedia and Security*], *MM&Sec*, 3–14, ACM (2007).

[13] Filler, T., Judas, J., and Fridrich, J., "Minimizing additive distortion in steganography using syndrome-trellis codes," *IEEE Trans. Inf. Forensics Security* **6**(3), 920–935 (2011).

[14] Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. J., "A kernel method for the two-sample-problem," in [*Advances in Neural Information Processing Systems 19*], Schölkopf, B., Platt, J., and Hoffman, T., eds., 513–520, MIT Press (2007).

[15] Pevný, T. and Fridrich, J., "Benchmarking for steganography," in [*Proc. 10th Information Hiding Workshop*], *LNCS* **5284**, Springer (2008).

[16] Kodovský, J. and Fridrich, J., "Influence of embedding strategies on security of steganographic methods in the JPEG domain," in [*Proc. SPIE: Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*], **6819** (2008).
[17] Richardson, I. E., [*The H.264 Advanced Video Compression Standard*], John Wiley and Sons Ltd (2010).
[18] Fridrich, J. and Kodovsky, J., "Rich models for steganalysis of digital images," *IEEE Trans. Inf. Forensics Security* **7**(3), 868–882 (2012).
[19] Pevný, T. and Fridrich, J., "Detection of double-compression in JPEG images for applications in steganography," *IEEE Trans. Inf. Forensics Security* **3**, 247–258 (2008).
[20] Pevný, T. and Fridrich, J., "Merging Markov and DCT features for multi-class JPEG steganalysis," in [*Proc. SPIE: Security, Steganography, and Watermarking of Multimedia Contents IX*], **6505**, 03–14 (2007).
[21] Fridrich, J., Goljan, M., and Du, R., "Steganalysis based on JPEG compatibility," in [*Proc. SPIE: Multimedia Systems and Applications IV*], **4518**, 275–280 (2001).
[22] Budhia, U. and Kundur, D., "Digital video steganalysis exploiting collusion sensitivity," in [*Proc. SPIE: Sensors, Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense*], **5403**, 210–221 (2004).
[23] Cachin, C., "An information-theoretic model for steganography," in [*Proc. 2nd Information Hiding Workshop*], *LNCS* **1525**, 306–318, Springer (1998).
[24] Ker, A. D., "Estimating steganographic Fisher Information in real images," in [*Proc. 11th Information Hiding Workshop*], *LNCS* **5806**, 73–88, Springer (2009).
[25] Pevný, T., *Kernel Methods in Steganalysis*, PhD thesis, Binghamton University, SUNY (2008).
[26] Pevný, T., "MMD in the context of Ker's benchmark," (2008). Personal communication.
[27] Kodovský, J., "Simulator of the nsF5 algorithm with wet paper codes (released 2008)." `http://dde.binghamton.edu/download/nsf5simulator/` (last accessed April 2012).