Multimedia Tools and Applications
https://doi.org/10.1007/s11042-024-18755-2

Robust and efficient airplane cockpit video coding leveraging temporal redundancy

Iulia Mitrica[1] · Attilio Fiandrotti[1,2] · Christophe Ruellan[3] · Marco Cagnazzo[1,4]

Received: 22 August 2023 / Revised: 25 January 2024 / Accepted: 24 February 2024

Abstract
Airplane cockpit screens consist of virtual instruments where characters, numbers, and graphics are overlaid on a black or natural background. Recording the cockpit screen allows one to log vital plane data, as aircraft manufacturers do not offer direct access to raw data. However, traditional video codecs struggle at preserving character readability at the required low bit-rates. We showed in a previous work that large rate-distortion gains can be achieved if the characters are encoded as text rather than as pixels. We now leverage temporal redundancy to both achieve robust character recognition and improve encoding efficiency. A convolutional neural network is trained for character classification over synthetic samples augmented with occlusions to gain robustness against overlapping graphics. Further robustness to background occlusions is brought by a probabilistic framework that error-corrects the output of the convolutional neural network. Next, we propose a predictive text coding technique specifically tailored for text in cockpit videos that achieves competitive performance over commodity lossless methods. Experiments with real cockpit video footage show large rate-distortion gains for the proposed method with respect to three different video compression standards. Notably, the H.264/AVC codec retrofitted with our method outperforms H.265/HEVC-SCC and is competitive with the much more complex H.266/VVC while preserving text and graphics. The entire pipeline described in this work has been implemented at Safran Electronics as an embedded avionics system drawing just 2W of power thanks to a combination of software and FPGA implementation.

✉ Attilio Fiandrotti
attilio.fiandrotti@telecom-paris.fr

Iulia Mitrica
iulia.mitrica@gmail.com

Christophe Ruellan
christophe.ruellan@safrangroup.com

Marco Cagnazzo
marco.cagnazzo@unipd.it

1 LTCI, Télécom Paris, Palaiseau 91120, France

2 Universitá di Torino, Torino 10124, Italy

3 Safran Electronics, Paris 91940, France

4 Universitá di Padova, Padova 35122, Italy

Published online: 04 April 2024

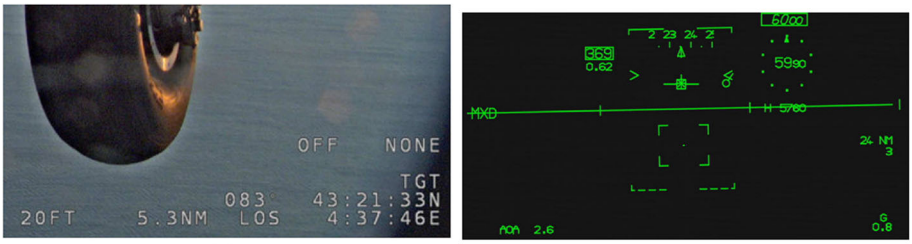
Keywords Airplane cockpit · Video coding · Semantic coding · ConvNets

1 Introduction

In modern airplanes cockpits, gauges, and dials gave way to computer screens where Text (plane coordinates, altitude, aircraft speed, etc.) and Graphical Objects (virtual horizon, heading, route waypoints, etc.) are rendered (over a video streams) as in Fig. 1. Fleet operators commonly record the cockpit screens to log cockpit screens since aircraft manufacturers usually prevent the operators from accessing the plane raw data due to security concerns. Cockpit videos are commonly recorded onboard and records are periodically downloaded when the plane undergoes maintenance and later on visually inspected in the case of malfunctions, accidents, etc. Clearly, such recordings are only as useful as long as the TGOs and text (characters, digits) in particular remain readable, i.e. if high-frequency components corresponding to sharp color transitions at the boundaries between text and background are preserved, whereas the background can be sacrificed to some extent. Preserving high frequencies in video requires in principle low-to-mild quantization, i.e. high encoding rates, or specialized video coding tools, i.e. high encoding complexity. However, the available storage for video onboard is usually limited and the constraints on power consumption (and heat dissipation) in typical avionic gears are typically tight. The tension between the requirements for encoding the video at low bitrates preserving text readability on one side, and constraints on complexity on the other side, call for ad-hoc video coding schemes beyond off-the-shelf video coding standards.

In our previous work [16] we introduced the idea of *semantic* encoding of cockpit screens. First, on-screen text and graphical objects are detected and encoded by their labels (character or digit value, shape) and attributes (positions, color, etc.). Then, the corresponding pixels are inpainted, yielding a *residual* cockpit screen images with few high frequencies. The residual video can then be encoded at a low bitrate by any off-the-shelf codec and stored together with the encoded text. The decoding process consists simply in recovering the residual video and overlaying the decoded characters and objects. In our previous work, we considered a simple *all-Intra* approach, i.e., text and residual video are independently encoded frame by frame. Yet, we achieved large rate-distortion (RD) gains with respect to the state-of-the-art SCC (Screen Content Coding) extension H.265/HEVC standard while preserving character readability. Next, in [15], we showed that if the residual video is encoded exploiting temporal prediction, an H.264/AVC [26] encoder retrofitted with our scheme becomes competitive with H.265/SCC, enabling lower encoder complexity. However, our experiments also showed two directions for further improving our scheme performance: i) character recognition accuracy needs improvements when on-screen characters are temporarily occluded by moving objects on the screen like the virtual horizon; ii) in some cases the rate of text approaches or even exceeds the rate of the residual video. This especially happens when high QPs are used or when the video has a flat (black) background, like in Fig. 1(b).

In this article, we vastly improve over our previous work [16] with a higher character recognition accuracy and more efficient compression technique.
First, we achieve robustness to character occlusions via a twofold strategy. To start with, we train the convolutional network in charge of character recognition over synthetic text samples augmented with noise in the form of occlusions with lines and other shapes that typically appear in cockpit videos. Next, we state character recognition as a maximum a posteriori (MAP) classification problem given the past video, so to take into account temporal



(a) (b)

Fig. 1 Examples of airplane cockpit videos. On the left, text is overlaid on natural background. On the right, text is displayed against a black background and some characters are partially occluded by the virtual horizon line

redundancy. We show that, under some mild hypotheses, the optimal yet costly video-based estimator is equivalent to a much simpler image-based estimator. The MAP approach slashes the character recognition error rate up to 18 times for simple sequences and at least 1.5 times for difficult sequences with multiple occlusions and rapid character flickering. The improved character recognition even in the case of occlusion is of paramount importance towards enabling flawless reconstruction of the encoded video and hence enhancing aviation operations and safety, as we discuss later on.

Second, we reduce the text encoding rate by devising an inter-frame character coding scheme tailored to the characteristics of text in cockpit screens. Leveraging the temporal correlation among adjacent frames reduces the bitrate required to encode characters to a point where it becomes negligible with respect to the residual video rate, even at high QPs. Our experiments show that the overall encoding rate for our inter-frame text coding paired with H.264/AVC for the residual video is competitive with H.265/HEVC-SCC and in some cases even with the recent H.266/VVC codec, yet at a fraction of the encoder complexity for a wide range of content type, resolutions and screen aspect ratios.

The rest of this article is organized as follows. After reviewing the necessary background in Section 2, we introduce the proposed methods in Section 3. The experimental conditions, the results, and the related discussion are illustrated in Section 4. Section 4.5 discusses the hardware implementation of the proposed pipeline. Finally, Section 5 draws conclusions and proposes some future developments.

2 Background

In this section, we describe the relevant background to screen content coding (SCC), from early approaches to standardized solutions and semantic approaches.

2.1 Compound video coding

Compound video contents refers to the case where computer-generated text, graphics or other objects are rendered over a natural background. Compressing such videos has been recognized as a special case of video compression demanding *ad hoc* tools since the 90s [4, 5, 21]. However, only in the 2010s screen content coding tools have been deeply investigated and included in standard video codecs [13]. The large majority of papers dealing with screen content coding or compound image and videos share a common approach: the blocks of the current image are encoded with a suitable compression tool, which is decided by explicitly or implicitly classifying the block as "natural content" or "screen content". Most of the

earlier techniques relied on explicit block classification, while in recent standards, such as HEVC/SCC and VVC, the coding tool selection is decided through a rate-distortion optimization procedure, and SCC methods are selected if they are the best performing on the current block.

Wang et al. [25], considered time redundancy for text, but the latter is still compressed with a pixel-based approach. When a block is recognized as text, it is compressed in the spatial domain and can be directly copied from the same position in the previous frame (skip). The pixels are quantized to several major colors and the indices of these major colors are entropy-coded to achieve a high-efficiency compression. In [9], the authors propose block-based compression techniques using block classification into text and natural ones, being used to adjust the quality over the coding rate. In [6], a lossless video scheme based on progressive B-spline curve fitting over time is proposed, showing reduced entropy for both natural and synthetic sequences.

2.1.1 Screen content coding in standards

More recently, the JVET of the ISO/ITU recognized the importance of screen coding an amendment to the H.265/HEVC video coding standard [27]. The Screen Content Coding (SCC) extension [28] defines ad-hoc tools for screen coding such as Intra Block Copy which acts like a motion compensation within current frame, and as Palette Mode, which signals the pixel values directly, exploiting the reduced number of colors typical of computer screens. Adaptive color transform, and adaptive motion vector resolution are also introduced. The above-described new encoding tools result in different RD characteristics than standard HEVC, so ad-hoc RD models [7] and ad-hoc rate control algorithms have been proposed [24].

The new H.266/VVC standard [2] retains the SCC tools in all the profiles. For example, the intra-prediction tool [29] has acquired novel modes, as well the standard now supports coding block partitioning. Unfortunately, improved encoding efficiency comes at the cost of increased complexity over HEVC, which in turn is already more complex than H.264/AVC [8]. While such extra complexity may be tolerable in some cases, in our cockpit video coding context it is just not acceptable due to strict bounds on power consumption and heat dissipation that are enforced for typical plane avionics. Such considerations on complexity are among the main drivers of the semantic approach to cockpit video compression that we rely upon in this work.

2.1.2 Semantic video compression

Semantic compression is based on the idea that the compression method should take into account the intended meaning of the most important feature in an image (in our case, the text), rather than straightforwardly optimize a rate-distortion trade-off. The first step consists in detecting the text in the (compressed) video. For example, [23] tackles the preliminary problem of detecting text in videos, which is made complex by compression artifacts and low resolution. The authors propose a hybrid method based on morphological filtering and 2-D discrete wavelet transforms. Also, the work in [18] deals with detecting text in H.264/AVC compressed video, this time using the integer discrete cosine transform (DCT) coefficients of intra-frames. Increasing the complexity of the proposed methods, other articles put forward deep learning methods to detect objects in images or videos. Unfortunately, in our avionics context, complexity plays a key role, ruling out existing approaches and promoting the

development of an ad-hoc text-and-object detection pipeline, which we implemented with a hybrid software-hardware approach as described in the last section of this work.

3 Proposed architecture and contributions

In this Section, we describe the main functional blocks of the proposed semantic codec architecture, while we refer the reader to our previous work [16] for further details.

3.1 Semantic codec architecture

The encoder (left part of the Fig. 2) works as follows. First, TGOs are detected and recognized exploiting the fact that must appear in predefined locations to allow the pilot to spot them with ease. Character recognition is addressed via a LeNet5-based ConvNet that in our previous experiments did strike a favorable tradeoff between character recognition accuracy and complexity. Namely, in [16] we investigated a fully connected architecture known as LeNet300 and the convolutional LeNet5. LeNet300 is a simple fully connected network composed of two hidden layers with 300 and 100 units (neurons) respectively with sigmoid activation functions and one output layer. Our expeirmnt found that LeNet5 showed better accuracy than LeNet300 for a smaller number of parameters and bearable computational complexity. Second, TGOs are encoded in the semantic domain with the lossless methods described below rather than in the pixel domain. Third, the Navier-Stokes algorithm inpaints the TGOs yielding a smooth computer screen without high frequencies: the goal of inpainting is only removing high frequencies, as the inpainted TGOs will be rendered over by the decoder as described below. Fourth, residual images are encoded with an off-the-shelf video encoder such as H.264/AVC as its low complexity allowed us to implement it in FPGA over a tight power budget. For each frame, the encoder produces a smooth residual image without TGOs and a side stream encoding the TGOs in their semantic domain.

The decoder architecture (right part of the Fig. 2) is as follows. First, residual images are decoded producing a smooth background without text or graphics. Second, the side stream is decoded and TGOs are synthesized in the pixel domain and superimposed in the original position over the residual image. The result is an image where TGOs are not affected by compression artifacts as in traditional video codecs.

With respect to our previous research, this work introduces i) robust training of the ConvNet in charge of character recognition via synthetically augmented samples, ii) error

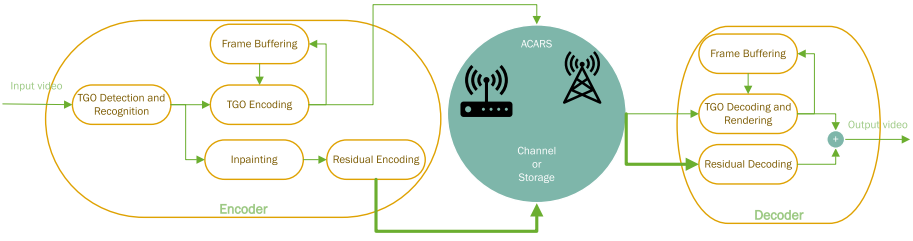


Fig. 2 Architecture of our proposed scheme for cockpit video compression. On the left, the encoder yields a residual video without TGOs and a side stream with the TGOs encoded according to their semantics. At the receiver side, the decoder superimposes newly synthesized TGOs over the residual video, recovering the original frame

correction based on a posteriori probabilities iii) full temporal predictive encoding of TGOs to slash the character coding rate iv) temporal predictive encoding of the residual video.

3.2 Robust character recognition

As shown in Fig. 1(b), characters in cockpit screens may be occluded by graphical objects in the foreground and background clutter. While the LeNet5 convolutional neural network (ConvNet) [12] performed very well when characters are not occluded, its performance degrades with occlusions. Our approach towards robust character recognition rests on two complementary legs that are described in the following.

3.2.1 Training on synthetic occluded samples

In our previous endeavors, we trained the LeNet5 from thousands of samples extracted from airplane cockpit videos. Unfortunately, few of these samples included occlusions from foreground objects, which made the network somewhat fragile to occlusions. We also experimented training our LeNet5 on publicly available datasets of occluded characters: however, the experiments showed poor performance that we attributed mainly to the lack of datasets with the peculiar type of occlusions found in cockpit screens. Because of the relative lack of occluded samples, we devised an approach to generate synthetic samples with occlusions. Our LeNet5 implementation takes in input samples sized 22×28 due to complexity and performance considerations, however, the described approach applies to any sample size. First, we randomly draw one character or symbol among those that shall be recognized. The character is then rendered within a 10x13 frame with an alpha channel over 256 levels of pixel intensity with color, font face, and size depending on the specific situation. Next, the frame is superimposed over a 22×28 empty canvas at a randomly drawn position, to enable robustness against loose character detection. Third and last, we draw random graphics over the canvas to gain the network robustness to occlusions. In the example, a line is drawn at a random angle and random position with probability $p_{occ} = 0.5$. This method enables generating a large dataset with controlled occlusions. In our experiments, we generate a dataset of 5000 samples that are divided into 80% training and 20% validation. In the experimental section, we show the benefits in terms of robustness of a LeNet5 trained with the above approach.

3.2.2 Exploiting temporal redundancy

In this section, we show how we propose to take advantage from temporal redundancy of characters and graphical objects in cockpit screens. We start from the observation that, considering a specific character (e.g., the first digit of the spatial coordinates) its "value" at time t is statistically dependent from its value at time $t - 1$. As for the notation, we refer to the value of a character in frame t with Y_t. It is possible to introduce mathematical models of the dependence between the values of the same character in consecutive frames though the conditional probability $P_{Y_{t-1}|Y_t}(y_{t-1}|y_t)$. The models used in this work are detailed in Section 3.2.3, and notwithstanding their simplicity, their impact on recognition performance is considerable, as shown in the experimental section. However, we underline that our goal is not to look for the best models. If a better model than the one used here would be made available, we would immediately be able to plug it into our proposed method, because, as shown later on, we only need an estimate of $P(y_{t-1}|y_t)$ (we drop the subscript for simplicity of notation).

Now we show how we leverage these statistical dependencies to improve recognition accuracy with negligible complexity increase. Before starting the mathematical development, we must mention that our proposed method is similar in spirit to some works in the literature such as [14, 19], which propose to improve the output of a classifier (the classes *a posteriori* probabilities) given some updated *a priori* probabilities of the data.

We consider a given character in the cockpit screen. To fix ideas, it could be the first digit of the latitude (see bottom-right part of Fig. 1(a)). Let X_t be the random vector representing the pixel patch of the character in the current frame t. The "value" of the character is just its class label, and it is represented with Y_t. In the current example, the realization of Y_t is "4". Thus, the baseline classifier is the one that computes the probability of a class label given the pixel patch: in our system, it is a ConvNet providing the conditional probabilities $P(y_t|x_t)$, and inferring from them the MAP estimator $\bar{y}_t$ of the character value:

$$\bar{y}_t = \arg\max_{y_t} P(y_t|x_t). \tag{1}$$

The baseline classifier in (1) does not take into account the temporal dependence among the instances of the same character, and it is the one we used in our previous works [15, 16].

Now we show our proposed method. We start from the idea of using the MAP estimator that also considers the past value y_{t-1} :

$$\widehat{y}_n = \arg\max_{y_t} P(y_t|x_t, y_{t-1}). \tag{2}$$

This new estimator is more convenient than the one in (1) because it takes into account the temporal dependence through the character label in the previous image, y_{t-1}. On the other hand, it requires in principle a new and more complicated ConvNet, and it could not take any advantage of an existing baseline classifier. One of the main contributions of the following is that, under a very reasonable hypothesis, we show how to compute $\widehat{y}_t$ without retraining the baseline classifier that provides $P(y_t|x_t)$: we only need to inject in the classification process the conditional probabilities $P_{Y_t|Y_{t-1}}(y_t|y_{t-1})$.

Our hypothesis is a form of *conditional independence*: we assume that, given the value Y_t of the character at time t, the patch X_t is independent from the previous value Y_{t-1}. In other words, Y_{t-1}, Y_t and X_t form a Markov chain in the order (sometimes this is expressed as $Y_{t-1} \rightarrow Y_t \rightarrow X_t$). We are assuming that the past value Y_{t-1} only influences the current patch through Y_t, but once given the latter, the former has no statistical influence on X_t. In our example about the latitude digit, our assumption means that, once we know the value of the current character (ie., "4"), the distribution of the current pixel patch is independent of the value of the character in the previous frame. With this hypothesis, we develop the conditional probability in (2) as follows:

$$\begin{aligned} P(y_t|x_t, y_{t-1}) &= P(x_t|y_t, y_{t-1}) \frac{P(y_t, y_{t-1})}{P(x_t, y_{t-1})} \\ &= P(x_t|y_t) \frac{P(y_t, y_{t-1})}{P(x_t, y_{t-1})} \qquad (3) \\ &= P(x_t|y_t) \frac{P(y_t|y_{t-1})P(y_{t-1})}{P(x_t, y_{t-1})} \\ &= \frac{P(x_t|y_t)}{P(x_t|y_{t-1})} P(y_t|y_{t-1}) \qquad (4) \end{aligned}$$

where in (3) we used the conditional independence hypothesis. Now, using the Bayes' rule, (4) becomes:

$$\begin{aligned} P(y_t|x_t, y_{t-1}) &= \frac{P(y_t|x_t)}{P(y_{t-1}|x_t)} \frac{P(y_{t-1})}{P(y_t)} P(y_t|y_{t-1}) \\ &= \frac{P(y_t|x_t)P(y_{t-1}|y_t)}{P(y_{t-1}|x_t)} \end{aligned} \tag{5}$$

The MAP estimator can then be written as

$$\widehat{y}_t = \arg\max_{y_t} P(y_t|x_t)P(y_{t-1}|y_t) \tag{6}$$

where the denominator of (5) is neglected because it does not depend on y_t. This result provides a way to account for temporal dependence in the character classifier: the MAP classifier estimating the current value given the pixel patch X_t and the previous label can be computed just by *weighting* the class probabilities $P(y_t|x_t)$ given by the baseline classifier by the conditional probabilities given by the model, and then picking the maximum of the weighted probabilities.

As for practical aspects of the proposed method, we also observe that, in order to compute $P(y_{t-1}|y_t)$ we are supposed to use the true value y_{t-1} of the character at time $t-1$. In practice, we do not have access to this information, but only to the previous estimate. We then propose to use $\widehat{y}_{t-1}$ in place of y_{t-1} for computing the weights. Therefore, our estimator is iteratively computed as:

$$\widehat{y}_t = \arg\max_{y_t} P(y_t|x_t)P(\widehat{y}_{t-1}|y_t), \tag{7}$$

rather than with (6). The use of an iterative algorithm to estimate the character labels has of course some inconvenience. First, the estimator in (7) is equivalent to the one in (6) if and only if the previous label was correctly estimated (ie., $\widehat{y}_t = y_t$. Second, an incorrect label can trigger error propagation. Third, we need an initialization value for the characters of the first frame.

Notwithstanding these problems, it is not difficult to imagine some solution to these problems. As observed in [16], the baseline classifier $\bar{y}_t = \arg\max_{y_t} P(y_t|x_t)$ has recognition rates close to 100% when the characters are not occluded. Thus we use $\bar{y}_0$ to initialize the recognition: $\widehat{y}_0 = \bar{y}_0$.

In order to limit error propagation, one could periodically check if $p(\bar{y}_t|x_t)$ is larger than a suitable threshold. In this case, one can safely assume that there is no occlusion and thus use $\widetilde{y}_t = \bar{y}_t$. This would stop any possible error propagation because the estimated label is not affected by past estimations.

In all our experiments we used $\widehat{y}_0 = \bar{y}_0$ to initialize the character recognition process and we observed that error propagation is extremely rare. As a consequence, the proposed method largely improves character recognition even in the case where we do not resort to specific protection against error propagation. The search for an optimized strategy against recognition error propagation is left for future works.

3.2.3 Models for character conditional probabilities

As discussed above, in order to take into account temporal dependence in character recognition, we need a model for the conditional probability of the current character value given its value in the previous instant. In principle, this probability distribution varies from cockpit to cockpit, however, our experiments show that some simple model works reasonably well in

practical situations. We use two different parametric models to address the cases where y_t is a letter or a digit.

In the case of a letter, our model is

$$P(y_{t-1}|y_t) = \begin{cases} p & \text{if } y_t = y_{t-1} \\ \frac{1-p}{M-1} & \text{otherwise} \end{cases} \tag{8}$$

that is, we assume that there is a certain probability p a letter does not change between consecutive frames. If the letter changes, our model assumes that it can switch with uniform probability to any of the remaining $M-1$ characters. This model has only one parameter p, that can be reliably estimated by observing how frequently a letter changes in some training data. This model could be improved, e.g., by updating on-line the letter switch frequency based on observed data. However, our experiments already show very good performance with the simple model in (8).

In the case of a digit, we use the following parametric model:

$$P(y_{t-1}|y_t) = \begin{cases} p & \text{if } y_t = y_{t-1} \\ q & \text{if } y_t = (y_{t-1} \pm 1) \mod 10 \\ \frac{1-p-2q}{7} & \text{otherwise} \end{cases} \tag{9}$$

where our parameters are the probability p that the digit does not change and the probability q to switch to the next or previous digit. If neither case happens, we assume a uniform probability to switch to any of the remaining $M-3 = 7$ digits. As in the previous case, we estimate p and q off-line from some training data and, even though it is not difficult to design some on-line method that would adapt the estimations of p and q to the observed data, our simple model of (9) already has very good performance, as shown in the experimental Section.

3.3 Efficient character coding using temporal redundancy

We propose a method to encode characters in a frame exploiting their specific temporal redundancy with respect to the characters displayed in the previous frame(s). We start with an example: Fig. 3 shows the frame to be encoded at time t (right) and the reference frame $t-1$ used for prediction (left). Let each character be represented with a *tuple* of size N_T composed by the position (horizontal and vertical coordinate) and the value (i.e., the recognized letter or digit): in this case $N_T = 3$, but more character features (font, size, color) could be added if needed.

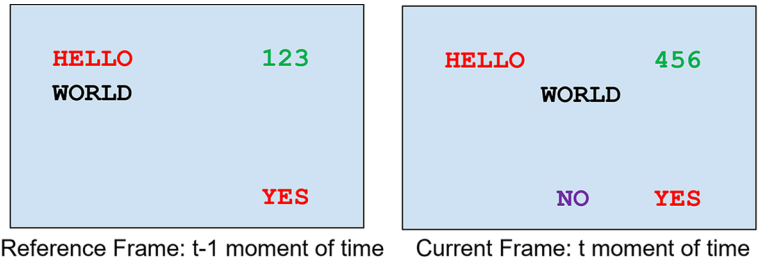


Fig. 3 Toy example of temporally consecutive cockpit screen frames: frame at time t is predicted from frame at time $t-1$

Now, let us assume that N_C characters are detected in a frame and are represented in raster scan order into a $N_C \times N_T$ table, as shown in Table 1 (the number of recognized characters can be different from one image to the other). In cockpit screens a strong correlation between the content of adjacent frames exists. For example, a (string of) character(s) often retains both position and value across multiple frames (red text in Fig. 3). In other cases, a character may change value and retain position, as the text in green (data coming from instruments, such as altitude, coordinates, *etc.*, which for ease of reading appear always in the same position); Otherwise, text may change the position and hold value (black string), for example when it is associated to some object tracked by the on-board sensors. Finally, text may suddenly pop-up in a position where no text was found previously (purple). Therefore, we define a set of *flags* corresponding to these cases and that are used to encode the current table given some reference. We use two indexes (`IRef` and `ICur`) that run through the tables, trying to find the most effective predictor for the current character in the reference table. Without loss of generality, we assume that characters in both tables are stored in raster scan order. We refer to the number of tuples (each representing a character) in the two tables as N_{CC} for the current table and N_{CR} for the reference table.

The proposed algorithm (see Fig. 4) is initialized with `ICur`=1 and `IRef`=1. Then, step one of our algorithm consist in retrieving the tuples referred by `ICur` and `IRef`, unless `ICur` $> N_{CC}$ (in this case the algorithm terminates), or `IRef` $> N_{CR}$ (in this case the reference tuple is a special, "void" one, but the algorithm does not stop). Step 2 consists in comparing the two tuples, respectively referred to as CC and RC (for current and reference character). The following cases are possible:

1. The two tuples have the same value and position

Table 1 Reference table, current table and corresponding encoding primitives for the example in Fig. 3

Index	Reference table		Current table		Encoding primitive
	Value	Position	Value	Position	
1	H	30, 20	H	30, 20	`EncodeF("U")`
2	E	30, 28	E	30, 28	`EncodeF("U")`
3	L	30, 36	L	30, 36	`EncodeF("U")`
4	L	30, 44	L	30, 44	`EncodeF("U")`
5	O	30, 52	O	30, 52	`EncodeF("U")`
6	1	30, 100	4	30, 100	`EncodeF("C") EncodeV("4")`
7	2	30,108	5	30, 108	`EncodeF("C") EncodeV("5")`
8	3	30, 116	6	30, 116	`EncodeF("C") EncodeV("6")`
9	W	40, 20	W	40, 40	`EncodeF("M") EncodeD("(0,20)")`
10	O	40, 28	O	40, 48	`EncodeF("M") EncodeD("(0,20)")`
11	R	40, 36	R	40, 56	`EncodeF("M") EncodeD("(0,20)")`
12	L	40, 44	L	40, 64	`EncodeF("M") EncodeD("(0,20)")`
13	D	40, 52	D	40, 72	`EncodeF("M") EncodeD("(0,20)")`
14	Y	60, 100	N	60, 48	`EncodeF("N") EncodeT("(N,60,48)")`
15	E	60, 108	O	60, 56	`EncodeF("N") EncodeT("(O,60,56)")`
16	S	60, 116	Y	60, 100	`EncodeF("R") EncodeR("14") EncodeF("U")`
17			E	60, 108	`EncodeF("U")`
18			S	60, 116	`EncodeF("U") EncodeF("END")`

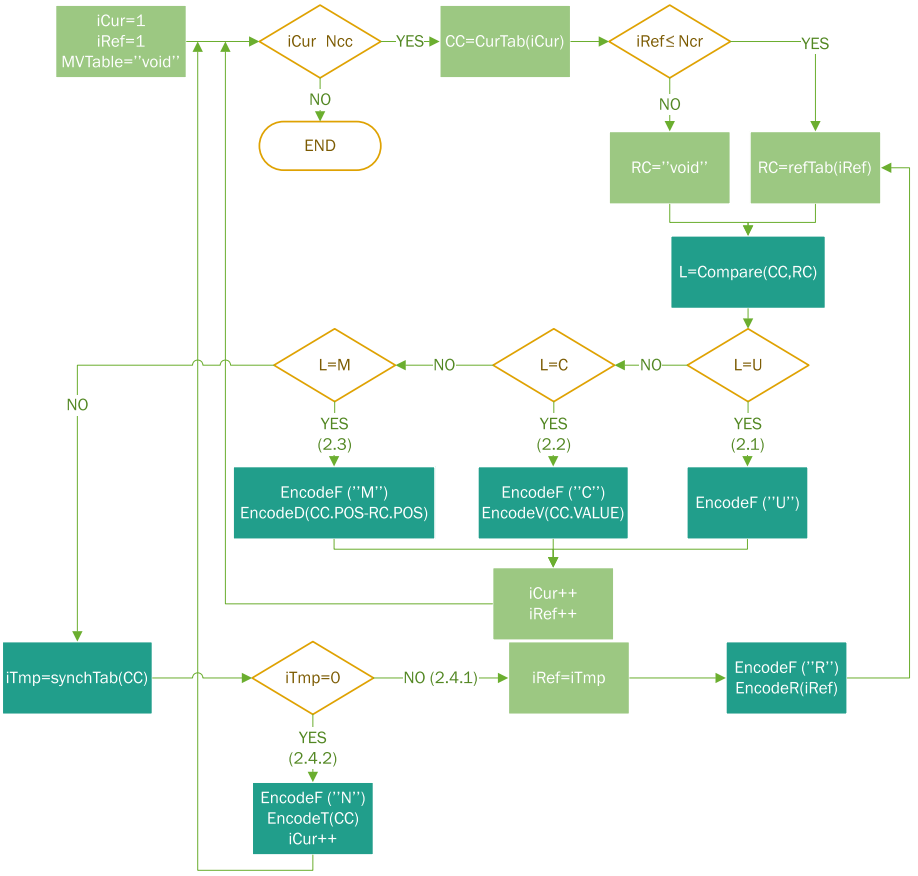


Fig. 4 Iterative algorithm to encode a current table of characters given a reference table using a set of flags. The figure illustrates in dark green the procedures with the primitives, in light green the assignations and in diamond the conditions. An example of this algorithm is detailed in Table 1 with respect to the Fig. 3

2. CC and RC have the same position and not the same value
3. CC and RC have the same value and not the same position, but their distance is within a given threshold
4. All the other cases, including void reference

In the first case (2.1), we just encode a flag "U" (for *unchanged*), with a suitable flag dictionary. The decoder will just copy the tuple RC to the current table at index `ICur`.

In the second case (2.2), we encode a flag "C" (for *changed*) and the new value of the character. The decoder will copy the tuple position from the reference table at index `IRef` to the current table at index `ICur`, and will read from the bitstream the value to use.

In the third case (2.3), we encode a flag "M" (for *moving*) and the displacement between the two tuples, using a suitable coding method that is described later on. The decoder will copy the tuple value from the reference table at index `IRef` to the current table at index `ICur`, and will adjust the position using the displacement read from the bitstream.

Finally, in all these three cases, both `ICur` and `IRef` are increased by 1, and the algorithm goes back to step 1.

For the fourth case, we must discriminate between two sub-cases: (2.4.1) the tuple in the current table corresponds to an unchanged, changed or moved tuple in the reference, but not at the index `IRef`; or, (2.4.2) the tuple in the current table is a new character appearing at time t. In order to synchronize tables, we use the `synchTab` procedure, which runs through the reference table using a temporary index `ITmp`, and for each tuple pointed by it, we check if it is an unchanged, changed, or moving tuple. If one of those three cases happens, it means that the two indexes must be realigned, so we set `IRef=ITmp`, encode a flag "R" and the new value of `IRef`, and go back to step 1. Finally, if no matching tuple is found in the reference table, `ITmp`=0 is returned and the character is considered as a new one: we encode a flag "N" and all the information of the tuple (value and position).

For our example, we show in Table 1 the primitives used in to encode the current character table. Each primitive must produce the bit-stream representing a symbol from a given input alphabet. `EncodeF()` must encode a flag among `U`, `C`, `D`, `N`, `R`, and `END`. In our implementation, we used a simple variable-length code for the flags (`C` as `100`, `D` as `101`, `N` as `110`, `R` as `1110`, `END` as `1111`), albeit more sophisticated methods such as context-based, adaptive arithmetic encoders could be used. `EncodeV()` must encode a character value, *i.e.*, a letter or a digit: we use a fixed-length code, but here as well more sophisticated tools could be considered. `EncodeT()` must encode a whole tuple: again, we consider simple fixed-length code, since this primitive is supposed to be used very rarely. Finally, as far as `EncodeD()` is concerned, its argument can be seen as a motion vector. Thus any motion vector coding method can be used here, from a simple fixed-length code to Exp-Golomb codes or to more sophisticated arithmetic codes. However, we observe that in cockpit screen videos, most if not all the moving characters of an image share the same motion. Therefore, we conceive a more effective dictionary-based method. We start with a void dictionary. As soon as a vector must be encoded, we look in the dictionary if the vector is already present. If yes, we encode the index of the vector in the dictionary with the Exp-Golomb code; if not, we append the vector to the dictionary and encode as an index the updated size of the dictionary. After that all the characters in the table have been encoded, we also have to write into the stream the motion dictionary, where each motion vector is represented with a fixed-length code.

The decoder simply decodes the flags and updates the table accordingly. The decoder keeps the indexes `ICur` and `IRef` in the same way as the encoder, and it has to read the motion vector table from the encoded stream.

In the following, the character encoding algorithm described in this section is referred to as the *predictive character encoding method*.

4 Experiments

In this section we validate the proposed method over a comprehensive set of experiments. We first describe the experimental setup, then we evaluate in the order character recognition accuracy and character compression efficiency and eventually we provide video compression results over multiple configurations.

4.1 Experimental setup

We experiment over the 10 cockpit video sequences in Fig. 5 (only 8 are shown). Seq. 1 and 2 have been captured with cameras installed outside the airplane in the visible and infrared spectrum, respectively. Seq. 3 and 4 have been generated superimposing text to MPEG test

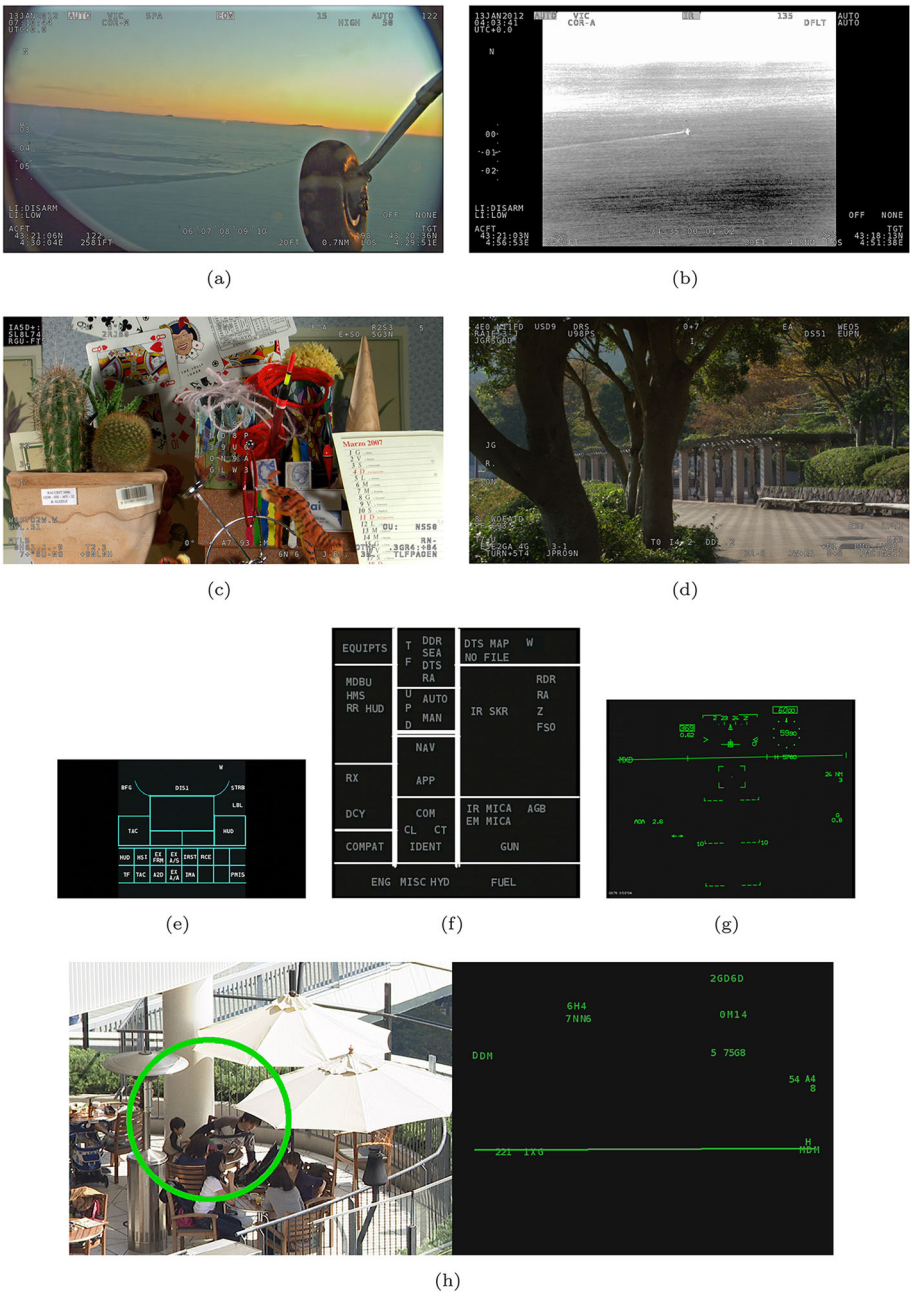


Fig. 5 The ten airplane cockpit screens video sequences used in our experiments (sequences 9 and 10 are identical to 8 yet with more occlusions). See Table 2 for the relative characteristics)

sequences *Cactus* and *Park*. These two sequences have a highly cluttered background and are meant to stress the encoder rather than representing the actual case of cockpit video. We refer to these four natural-background sequences as "Cockpit HD". Sequences 5, 6 and 7 are

actual cockpit screens, with complex, computer-generated text and graphics overlaid on black background ("Cockpit SD"). Finally sequences 8, 9 and 10 emulate the new glass cockpit control, which mixes natural video (*BQTerrace* from the MPEG test set) with computer generated complex graphics for head-up displays ("Cockpit HUD"). Sequences 9 and 10 are identical to 8, however they present more and more complex occlusions over the characters. The TGOs in the synthetic sequences 3, 4, 8-10, have been generated as follows. Characters can randomly appear in any position of the video. At each new frame, characters can change with a given probability reflecting the character temporal evolution in real footage. Concerning the graphic objects, we generate lines and circles for sequences 8-10. In the left part of the screen, a circle that slowly translates upward. In the right part we have: a slightly changing line which occludes some characters (seq. 8); a faster moving line which occlude characters (seq. 9); or two lines which move and occlude characters (seq. 10). Table 2 summarizes the characteristics of our test sequences.

4.2 Characters recognition accuracy

In this section we evaluate the results of our robust character recognition algorithm on sequences 7-10. Sequences 1-6 are not shown because characters are not occluded and the ConvNet recognizes all characters without a flaw.

The first row of Table 3 is relative to the baseline case where the ConvNet is trained without occlusions and class scores are not weighted according to the conditional probabilities. In other words, this is the methods used in our previous works [16] and [15]. The average character recognition error rate is 3.15 %, topping nearly 5% for sequence 10, where 2 moving lines occlude on screen characters.

The second row of the table refers to the case where the ConvNet is trained over occluded samples as in Section 3.2, however the class scores are not weighted according to the conditional probabilities as in as in (6). Training the network over occluded samples reduces the average rate of errors to 2.42 %, however most heavily occluded characters may still be misrecognized.

The third row of the table is for our proposed scheme where the ConvNet is trained with occlusions and class scores are weighted according to conditional probabilities, *i.e.* when temporal redundancy is taken into account. On sequence 7, the error rate drops by almost 20 times (17.9x) from 2.34 % to 0.13 %. In sequences 8 and 9, where only one line may occlude

Table 2 Characteristics of the cockpit video test sequences used in our semantic encoding experiments

# Seq.	Resolution	Class	Source
1 Fig. 5(a)	1920x1080	Cockpit HD	Actual footage
2 Fig. 5(b)	1920x1080	Cockpit HD	Actual footage
3 Fig. 5(c)	1920x1080	Cockpit HD	Synthetic
4 Fig. 5(d)	1920x1080	Cockpit HD	Synthetic
5 Fig. 5(e)	720x576	Cockpit SD	Actual footage
6 Fig. 5(f)	720x576	Cockpit SD	Actual footage
7 Fig. 5(g)	720x576	Cockpit SD	Actual footage
8 Fig. 5(h)	1440x576	Cockpit HUD	Synthetic, weak occlusions
9 Fig. 5(h)	1440x576	Cockpit HUD	Synthetic, mild occlusions
10 Fig. 5(h)	1440x576	Cockpit HUD	Synthetic, strong occlusions

Table 3 Character recognition error rate for sequences with occlusions (baseline is [16])

	Seq. 7	Seq. 8	Seq. 9	Seq. 10
Baseline	2.34 %	3.08 %	2.31 %	4.87 %
Train with occlusions only	1.73 %	2.31 %	1.67 %	3.97 %
Proposed	0.13 %	1.03 %	0.77 %	3.21 %
Error-rate variation w.r.t. baseline	−94.4 %	−66.6 %	−66.7 %	−34.1 %

characters, errors are 3 times less frequent, and in sequence 10, where two lines may occlude characters, one third of the errors is corrected. Most of the residual errors occur when a static line occlude a character (thus, the network is never able to properly see it); another potential source of error is when the occlusion happens at the same time as a character change. Finally, we remark that our robust character method entails only a negligible complexity increase (weighting of the probabilities).

To gain some insight about the error correction capability of the proposed method, we show an example in Fig. 6. In the top of this figure, we show a small block of pixels (patch) taken from sequence 9, at time t going from 0 to 10. The patch x_t shows the digit "6" at time 0 to 5, but at t=4 the digit is occluded by a moving line.

Then, in Fig. 6(b), we show the temporal evolution of the estimation of $P(y_t|x_t)$, ie., the probability of each class label y_t given the pixel patch x_t, which is provided by the baseline ConvNet. Each column of Fig. 6(b) corresponds to a time instant and the probability of each label is color mapped as shown in the right part of the figure. The estimated label $\overline{y}_t$ produced by the ConvNet is the arg max$_{y_t}$ of this distribution. We observe that, at time t=4 the ConvNet is induced to error by the occlusion, since $P(Y_4{=}8|x_4) > P(Y_4{=}6|x_4)$. This is also shown in Fig. 6(c), where we plot $P(y_4|x_4)$.

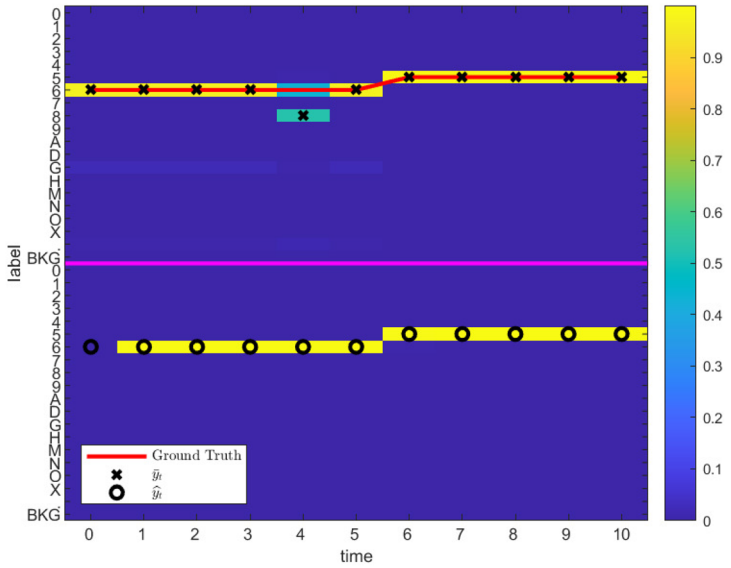
In opposition, when the weights are used, the estimated class is correct even at t=4: this is because the weight $P(Y_3{=}6|Y_4{=}6)$ is sufficiently larger than $P(Y_3{=}6|Y_4{=}8)$, and thus the weighted distribution (shown in the bottom part of Fig. 6(b) and with an orange line in Fig. 6(c) for t=4) has a peak for y_4=6. Moreover, the weighs do not prevent our classifier to find the correct label at time t=6, when the digit changes from 6 to 5.

4.3 Character coding efficiency with temporal redundancy

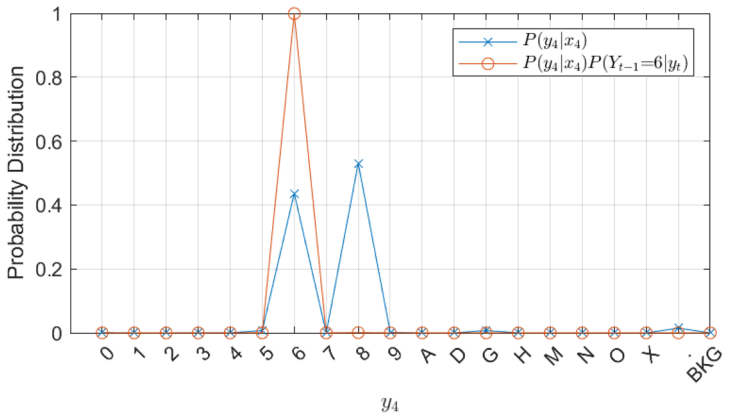
We now evaluate the efficiency of the predictive character encoding method proposed in Section 3.3. We consider as input to the algorithm the characters recognized in our ten test sequences and we encode value and position of all characters. Four lossless techniques have been considered for comparison: Huffman [10], Burrows-Wheeler transform (BWT, also block-sorting compression [3]), PPM (Dmitry Shkarin's prediction by partial matching method [22]), LZMA algorithm (Lempel-Ziv-Markov chain algorithm [17]). We also compare our proposed method with the baseline method we previously described in [16], where no temporal redundancy is taken into account. Table 4 shows the encoding rate in bits per character (i.e., the average number of bits needed to encode a tuple in the character table). If we encoded characters using a simple constant-length code, for HD sequences that would require 29 bits per character: the horizontal and vertical coordinates need each 11 bits, 6 for the label and one for the color. For HD sequences (1-4), our scheme requires around 30% less rate than the baseline; for HD and SD sequences, the bit-rate saving is as high as 80%.

0 1 2 3 4 5 6 7 8 9 10

(a) Temporal evolution from seq. 9 digit 6 switching to 5 between frames 5 and 6. At time t=4 the digit is occluded by a moving line.



(b) Temporal evolution of the probabilities of the estimated class labels y_t given the pixel patch x_t (top), and of the weighted probabilities $P(y_t|x_t)P(y_{t-1}|y_t)$ (bottom). The estimated labels are the $\arg\max_{y_t}$ of those functions, as in Eqs. (1) and (6).



(c) Class score distribution (blue line) and weighted scores distribution (orange line) at time t=4 (ground truth label is 6). These distributions correspond to the column t=4 in Fig. 6(b).

Fig. 6 In this example, the digit "6" being occluded by a line at time $t = 4$ and before transitioning to digit "5" at time $t = 6$

Table 4 Characters coding rates for different characters coding modes

Seq.	Huffman	BWT	PPM	LZMA	Baseline	Prop.	Prop @24fps
1	27.64	25.33	27.08	18.94	10.24	5.80	23.66 kbps
2	27.67	25.40	27.15	18.98	10.35	6.02	24.56 kbps
3	27.88	26.69	27.24	19.05	10.37	8.12	33.13 kbps
4	27.90	26.71	27.42	19.23	10.48	8.34	34.03 kbps
5	8.55	12.39	16.09	7.08	12.35	0.73	1.23 kbps
6	8.57	12.41	16.11	7.10	12.47	0.82	1.38 kbps
7	9.61	14.31	16.23	8.71	12.24	1.74	2.92 kbps
8	9.65	10.3	9.32	9.25	11.73	1.80	3.02 kbps
9	9.72	10.41	9.04	8.73	11.56	2.01	3.38 kbps
10	10.62	10.97	9.83	9.28	11.64	2.78	4.67 kbps
Av.	16.79	17.49	18.63	12.64	11.34	3.82	13.2 kbps

Measurement unit is the number of bits per character (the baseline is [16])

4.4 Video coding experiments

Finally, we compare the video compression efficiency of our scheme with state-of-the-art references. In our scheme, TGOs are encoded separately from the inpainted residual video, so the residual video coincides with the characterless screen background. We selected the H.264/AVC (JM 19.0) codec for residual video coding because, to the present date, is the only that can be implemented in FPGA within our given 2W power budget as detailed in Section 4.5 (*Prop-AVC* in the following). For the sole purpose of comparison, we consider an hypothetical implementation of our scheme where the residual video is encoded using the H.265/HEVC codec (HM-16.14). Unfortunately, this *Prop-HEVC* scheme does not fit our power budget at the moment, so it should not be considered as a practical alternative to *Prop-AVC*.

Next, we consider two reference schemes where TGOs are not removed form the screen and each frame is encoded as an image. The first reference is the the H.265/HEVC codec (HM-16.14 with Screen Content Coding extension SCM-8.3) (*HEVC-SCC* in the following). The second reference is the H.266/VVC codec (VTM13.2), which natively inherits HEVC-SCC screen content coding tools from (*VVC* in the following). Their complexity is out of our power budget, plus at the moment no complete hardware implementation of the SCC/VVC screen content coding tools exists, so they shall be considered only as references to benchmark Prop-AVC.

All experiments consider values for QP in the range from 20 to 45 with steps of 5, plus on the very low bit-rate range QP from 45 to 51 with steps of 1. The results are reported in terms of rate-PSNR curves and of Bjontegaard metric [1].

Our video coding experiments are organized below into two sets. In *Prop-Intra*, each residual frame is independently encoded as in our previous research [16], as a reference to benchmark our method. In *Prop-Inter* temporal prediction is enabled and these results represent the scheme proposed in this work.

As a remark, we would like to stress that these objective quality experiments are mainly meant to evaluate the encoding efficiency of our method. In a practical application of our method, the actual quality of the residual video would be of limited interest as long as the characters remain readable.

Sample videos are available for visual inspection[1]: due to the large size of the YUV files (about 25 MB each), we have uploaded the first frame of sequence 9 because it includes both natural and black background. Namely, we report both for the Inter schemes VVC, SCC, Prop-HEVC and Prop-AVC. For each coding schemes, we report all the QPs we experimented with from 20 to 50, resolution is 1440x576 px and format is YUV 4:2:0;bit depth is 10 bpp only for VVC, 8 bit otherwise.

4.4.1 Experiments with intra-frame prediction

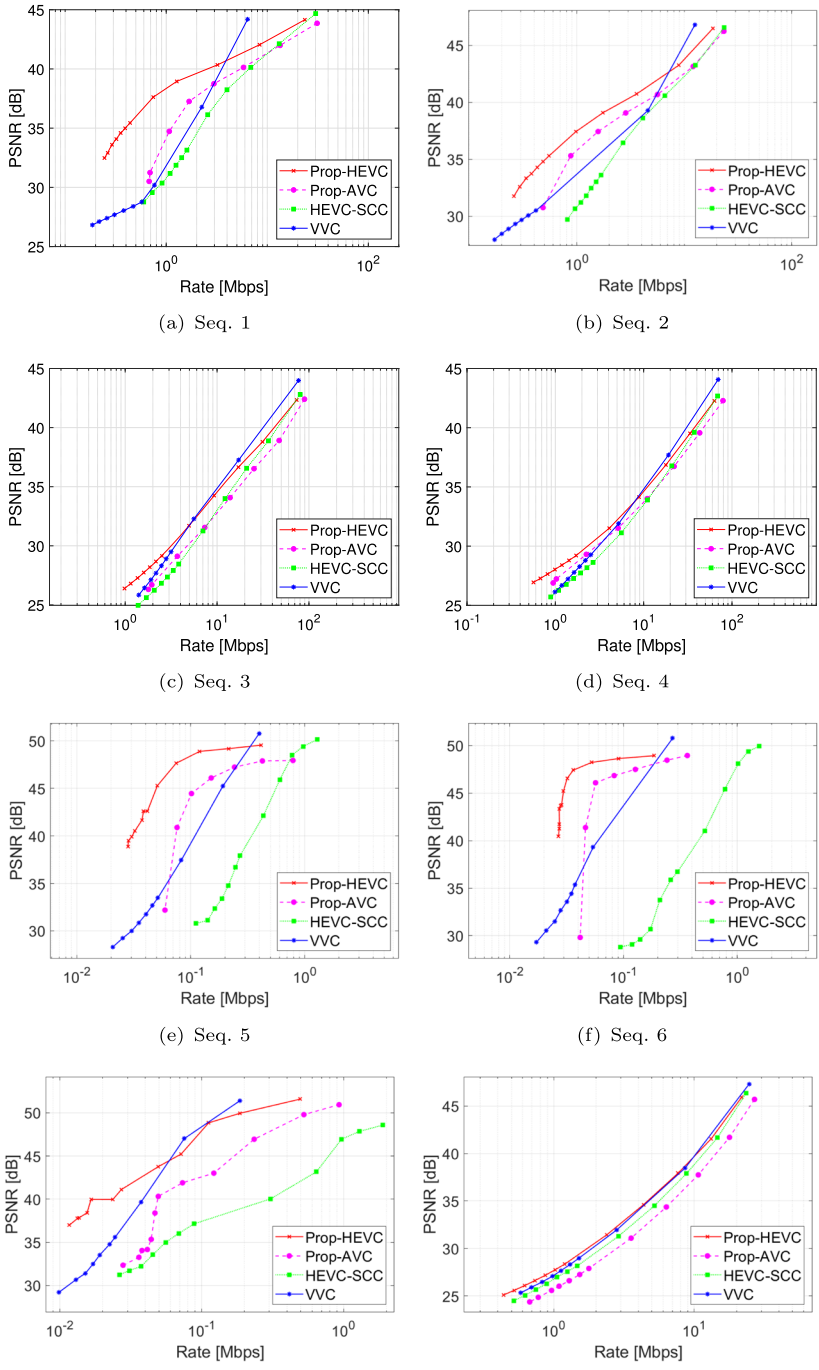
Figure 7 shows the rate-distortion curves for sequences 1-8 (results for sequences 9 and 10 are similar to 8 and omitted for the sake of conciseness). Prop-AVC consistently outperforms HEVC-SCC at low bitrates in HD and SD sequences (sequences 1-7). VVC does outperforms Prop-AVC, however the performance gap is thin despite VVC higher complexity. Prop-HEVC outperforms both Prop-AVC and VVC for all sequences and at almost any bitrates but the lowest QPs, where VVC still achieves better results. For sequences 5-7 (black background), our scheme shows a sudden rate drop above QP 40 since inpainting the text leaves to the encoder an almost black screen to compress, which is encoded with a few large PUs. Prop-AVC achieves performance closer or better to VVC yet at a fraction of the complexity, which is a major edge in our application scenario.

Table 5 reports Bjontegaard Delta PSNR (BD-PSNR) and Delta rate (BD-Rate) of the proposed method against HEVC-SCC. We consider two quality ranges: medium-to-high (QP=[40, 35, 30, 25]) and low-to-medium (QP=[50, 45, 40, 35]). Gains are high in particular at low rates: up to -90 % in BD-Rate for SD sequences with black background. Note that for sequences (5-7) BD-PSNR cannot be computed because we have not enough coding rate superposition. Somewhat smaller gains are observed for HD and HUD sequences because a larger share of the rate is used to encode the background.
Similar results are shown in Table 6, where we give the BD-PSNR and the BD-Rate of Prop-HEVC with respect to VVC *at very low bit-rate*. Prop-AVC remains competitive despite being much simpler: up to -60 % rate drop in the most favorable case.

4.4.2 Experiments with inter-frame prediction

We repeat the previous experiments enabling inter-frame prediction for both text and residual video encoding. Towards keeping the encoder complexity limited, we use an LD-P configuration with an Intra period of 4 frames. The encoder is allowed to keep just one frame in the decoded picture buffer (DPB) due to memory constraints of the hardware implementation described in the following. Our experiment showed that increasing the DPB size does not significantly improve the efficiency despite increased complexity. Figure 8 shows the corresponding rate-distortion curves. We recall that only Prop-AVC can be implemented in hardware, while the other schemes are meant for reference only. Overall, the trends resemble those of the previous case, except that now HEVC-SCC outperforms Prop-AVC also in a couple of HD sequences due to HEVC improved temporal prediction. However, Prop-HEVC still outperforms in a consistent way both HEVC-SCC and VVC as in the All-Intra experiments. This also can be seen from Bjontegaard metrics shown in Table 7 with respect to the HEVC-SCC reference. The gains are similar to the All-Intra case, yet now we achieve even larger gains for HUD sequences. One reason could be that the improved text compression

[1] https://drive.google.com/drive/folders/1X98bbDYd0hYp3DNrc2179Qxi3VAjcI_1



(a) Seq. 1 (b) Seq. 2

(c) Seq. 3 (d) Seq. 4

(e) Seq. 5 (f) Seq. 6

(g) Seq. 7 (h) Seq. 8

Fig. 7 PSNR vs. video bitrate for Prop-Intra coding configuration. Prop-HEVC and Prop-AVC have consistent gains wrt HEVC-SCC and VVC

Table 5 Bjontegaard gains for Prop-HEVC with respect to HEVC-SCC, both in All-Intra configurations

Quality range	Medium-to-high		Low-to-medium	
Sequence	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
Seq. 1	1.30 dB	−46.5 %	6.48 dB	−69.88 %
Seq. 2	1.06 dB	−30.28 %	3.82 dB	−57.33 %
Seq. 3	0.67 dB	−15.13 %	1.56 dB	−31.45 %
Seq. 4	0.60 dB	−13.79 %	1.23 dB	−32.99 %
Seq. 5	1.37 dB	−67.17 %	–	−79.25%
Seq. 6	1.45 dB	−68.2 %	–	−91.04%
Seq. 7	1.81 dB	−50.94 %	–	−79.45%
Seq. 8	0.63 dB	−6.2 %	0.86 dB	−17.06 %
Seq. 9	0.50 dB	−7.99 %	0.88 dB	−17.39 %
Seq. 10	0.38 dB	−6.20 %	0.79 dB	−15.73 %

has more impact in the Inter than in the Intra case, where the share of rate allocated to the text is smaller.

Looking at the comparison with VVC at very low bit-rate (Table 8), even in an Inter configuration, the proposed, HEVC-based scheme outperforms VVC.

We observe that while in [15] the semantic encoder suffers from the relatively high coding cost of intra-character encoding when the residual is inter-coded, our predictive character coding technique brings again large gains. Finally, Fig. 9 we shows the same picture crop for the four methods. Both Prop-AVC and Prop-HEVC maintain text readability at a coding rate much smaller than HEVC-SCC and VVC. Moreover, the two latter introduce characters artefacts that hinder the character readability. Conversely, with Prop-AVc and Prop-HEVC, the characters are perfectly readable.

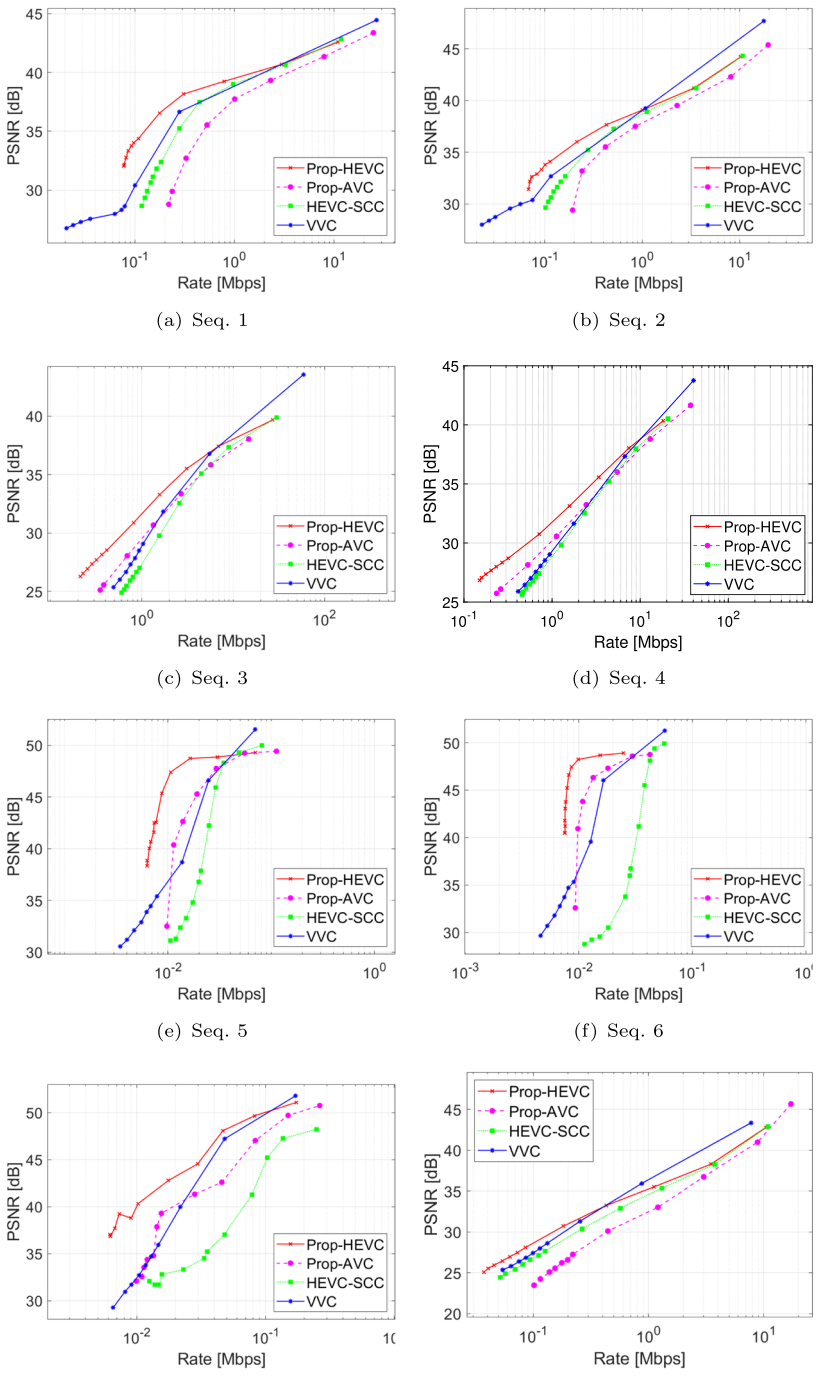
4.4.3 Bit-rate distribution analysis

As an insight on the reported gains, we report the distribution of the coding rate between residual video and TGOs. We consider sequences 1 and 5 respectively as representative of the cases where TGOs are overlaid on a natural background or a black screen. For each QP in {20, 30, 40, 50}, we report the total coding rate and the percentage thereof devoted to TGO

Table 6 Bjontegaard gains for Prop-HEVC with respect to VVC, both in All-Intra configurations

Sequence	BD-PSNR	BD-rate
Seq. 1	5.93 dB	−61.78 %
Seq. 2	3.07 dB	−46.60 %
Seq. 3	0.81 dB	−32.28 %
Seq. 4	1.36 dB	−37.79 %
Seq. 5	9.87 dB	−56.33 %
Seq. 6	9.94 dB	−57.43 %
Seq. 7	6.81 dB	−20.56 %
Seq. 8	0.53 dB	−15.94 %
Seq. 9	0.49 dB	−14.26 %
Seq. 10	0.56 dB	−15.80 %

The QP range is [48, 49, 50, 51]



(a) Seq. 1 (b) Seq. 2

(c) Seq. 3 (d) Seq. 4

(e) Seq. 5 (f) Seq. 6

(g) Seq. 7 (h) Seq. 8

Fig. 8 PSNR vs. residual video bitrate for Prop-Inter coding configuration. The proposed semantic coding scheme outperforms even recent off-the-shelf standardized codecs

Table 7 Bjontegaard gains for Prop-HEVC with respect to HEVC-SCC, both in LD-P configurations

Quality range	Medium-to-High		Low-to-Medium	
Sequence	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
Seq. 1	0.77 dB	−39.47 %	4.03 dB	−53.74 %
Seq. 2	0.52 dB	−21.75 %	2.04 dB	−43.52 %
Seq. 3	2.0 dB	−43.9 %	4.27 dB	−62.98 %
Seq. 4	1.58 dB	−36.23 %	3.07 dB	−61.28 %
Seq. 5	–	−83.56 %	–	−68.02%
Seq. 6	–	−85.64 %	–	−78.68%
Seq. 7	–	−56.33 %	–	−73.60%
Seq. 8	0.74 dB	−25.28 %	1.66 dB	−38.28 %
Seq. 9	0.75 dB	−24.94 %	1.81 dB	−40.49 %
Seq. 10	0.76 dB	−25.28 %	1.85 dB	−41.03 %

Table 8 Bjontegaard gains for Prop-HEVC with respect to VVC, both in LD-P configurations

Sequence	BD-PSNR	BD-rate
Seq. 1	3.89 dB	−53.85 %
Seq. 2	2.05 dB	−43.06 %
Seq. 3	1.45 dB	−46.12 %
Seq. 4	1.89 dB	−51.06 %
Seq. 5	5.82 dB	−52.37 %
Seq. 6	5.08 dB	−43.19 %
Seq. 7	1.12 dB	−41.55 %
Seq. 8	1.13 dB	−30.88 %
Seq. 9	1.22 dB	−30.48 %
Seq. 10	1.46 dB	−31.86 %

The QP range is [48, 49, 50, 51]

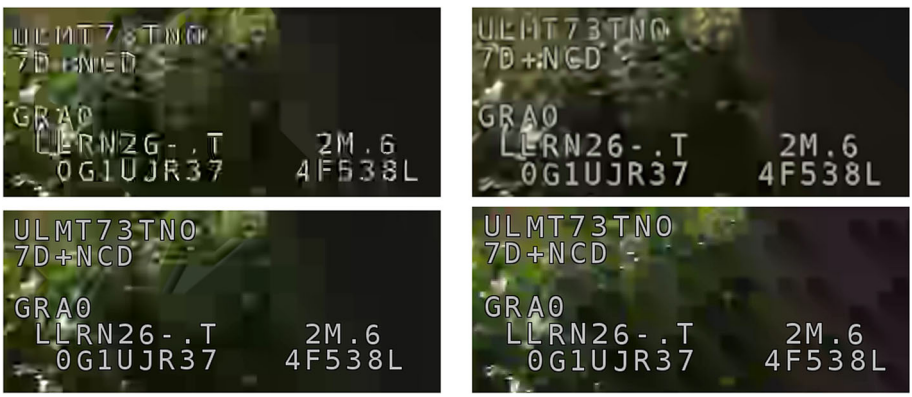


Fig. 9 Reconstruction artefacts at QP=50 for Prop-Inter. Top-left HEVC-SCC (Rate=0.47 Mbps, PSNR=25.86 dB). Top-right VVC (Rate=0.41 Mbps, PSNR=25.90 dB). Bottom-left Prop-HEVC (Rate=0.16 Mbps, PSNR=27.07 dB). Bottom-right Prop-AVC (Rate=0.26 Mbps, PSNR=26.10 dB)

Table 9 Total rate and share of TGOs coding rate for Prop-Intra, i.e. inter-frame prediction is disabled (baseline is [16])

	Seq.1 (natural background)			Seq. 5 (black background)		
QP	Total rate Proposed	TGOs rate Proposed	TGOs rate Baseline	Total rate Proposed	TGOs rate Proposed	TGOs rate Baseline
50	306 kbps	7 %	13 %	28 kbps	4 %	43 %
40	370 kbps	6 %	11 %	51 kbps	2 %	30 %
30	455 kbps	5 %	9 %	74 kbps	1.7 %	22 %
20	1287 kbps	2 %	3 %	119 kbps	1 %	15 %

coding. Moreover, we report the character coding rate for a setup equivalent to our previous work [16], i.e. when character and picture inter-frame coding are both disabled.
Table 9 shows the case where the residual video is encoded without temporal prediction (Prop-Intra). At the low bit-rates relevant for our work, the character rate grows up to a significant share if temporal correlation among TGOs is not exploited. This is especially true if the residual is just black background as in Seq. 5 and so the overall coding rate is low. With the method presented in our previous work, up to 43 % of the bit budget is allocated to the TGOs. In Seq. 1 the residual is more complex and so the residual rate increases, yet the text needs up to 13 % of the total rate. When the predictive text coding method is used ("Proposed") this share drops to more reasonable values.
The relevance of TGOs temporal prediction is even more evident in Table 10 when the residual video is inter-coded (Prop-Inter). The residual is better compressed and as a consequence, the text takes a larger share of the total bitrate, up to 80 % for Seq. 5 and 37 % for Seq. 1 if TGOs are not predicted. When TGOs are predicted, the TGOs rate share falls to about 20 %.

4.5 Hardware implementation

The pipeline described in this work has been implemented on the VS1410 Airborne Mission Data Recorder [20] platform illustrated in Fig. 10. Our pipeline is completely implemented into this system as an hybrid solution where a MIPS-based System on Chip (SoC) implements in software all the stages but character recognition and residual video compression. ConvNet-based character recognition and AVC-based background encoding are in fact offloaded to an Intel Aria V FPGA - $5AGXB1$ [11] with 300k gates for performance and efficiency reasons. The ConvNet has a recognition rate of 4320 characters per second, which proved more than

Table 10 Total rate and share of TGOs coding rate for Prop-Inter, i.e. inter-frame prediction is enabled (baseline is [15])

	Seq.1 (natural background)			Seq. 5 (black background)		
QP	Total rate TGOs	TGOs rate Proposed	TGOs rate Baseline	Total rate Proposed	TGOs rate Proposed	TGOs rate Baseline
50	94 kbps	25 %	37 %	6 kbps	21 %	80 %
40	193 kbps	12 %	20 %	8.5 kbps	15 %	74 %
30	324 kbps	7 %	12 %	10 kbps	12 %	69 %
20	801 kbps	3 %	5 %	16 kbps	8 %	58 %

Fig. 10 The VS1410 On-board mission recorder implementing the pipeline described in this work over a total of 2W of power

enough for practical use cases. The background video is H.264/AVC encoded in real-time using an H.264/AVC soft core over the same FPGA. Character recognition and AVC video coding exhaust almost completely the gates available on the FPGA. The pipeline includes 16 MB or RAM,hence the DPB is limited to 1 frame. The pipeline power consumption is below the 2W budget we were assigned, where 0.8 W are for the SoC, about 0.5 W for the FPGA and the rest for other ancillary devices such as I/O, etc.

5 Conclusions

In this work we exploit the temporal redundancy in airplane cockpit videos for lossless recording text and graphics at very low bitrates. First, we proposed an *ad-hoc* method for character encoding that leverages the temporal correlation, to the point where we enable nearly real-time streaming of key plane data with less than 3 kbps according to the video sequence. Then, we boost the performance of a ConvNet-based character detector leveraging the probability of a character appearing in a frame conditioned to the co-located character in the previous frame. The relatively low-complexity H.264/AVC codec retrofitted with our scheme becomes competitive with H.266/VVC for the goals of this work, despite the encoder much lower complexity that makes it a better candidate for integration in avionic electronics. The entire pipeline described in this article as been implemented in FPGA as a ruggedized on-board cockpit video recorder that can serve a number of purposes from pilot training to rescue mission.

Funding Open access funding provided by Università degli Studi di Torino within the CRUI-CARE Agreement. This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

Data availability statement The datasets and video sequences generated during and/or analysed during the current study are available online at the following address https://drive.google.com/drive/folders/1X98bbDYd0hYp3DNrc2179Qxi3VAjcI_1.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Bjontegaard G (2001) Calculation of average PSNR differences between RD-curves. In: VCEG Meeting, Austin, USA
2. Bross B, Wang YK, Ye Y et al (2021) Overview of the versatile video coding (VVC) standard and its applications. IEEE Trans Circuits Syst Video Technol
3. Burrows M, Wheeler D (1994) A block-sorting lossless data compression algorithm. In: Digital SRC research report, Citeseer
4. Cagnazzo M, Parrilli S, Poggi G et al (2007) Costs and advantages of object-based image coding with shape-adaptive wavelet transform. EURASIP J Image Video Process 2007 78323:13. https://doi.org/10.1155/2007/78323
5. De Queiroz RL, Fan Z, Tran TD (2000) Optimizing block-thresholding segmentation for multilayer compression of compound images. IEEE Trans Image Process 9(9):1461–1471
6. Ebadi M, Ebrahimi A (2021) Video data compression by progressive iterative approximation. Int J Interact Multimed Artif Intell
7. Esmaeeli H, Rezaei M (2022) A content-based intra rate-distortion model for HEVC-SCC. Multimed Tools Appl 81(12):16515–16536
8. García-Lucas D, Cebrián-Márquez G, Cuenca P (2020) Rate-distortion/complexity analysis of HEVC, VVC and AV1 video codecs. Multimed Tools Appl 79:29621–29638
9. Han B, Wu D, Zhang H (2010) Block-based method for real-time compound video compression. In: Mobile multimedia/image processing, security, and applications 2010, international society for optics and photonics, p 77080S
10. Huffman DA (1952) A method for the construction of minimum-redundancy codes. Proc IRE 40(9):1098–1101
11. Intel (2022) Arria v 5agxb1 fpga. https://ark.intel.com/content/www/us/en/ark/products/210398/arria-v-5agxb1-fpga.html
12. LeCun Y, Bottou L, Bengio Y et al (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
13. Liu S, Xu X, Lei S et al (2015) Overview of HEVC extensions on screen content coding. APSIPA Trans Signal Inform Process 4:e10. https://doi.org/10.1017/ATSIP.2015.11
14. McLachlan GJ, Krishnan T (2007) The EM algorithm and extensions, vol 382. John Wiley & Sons
15. Mitrica I, Fiandrotti A, Cagnazzo M et al (2019a) Cockpit video coding with temporal prediction. In: 2019 8th European workshop on visual information processing (EUVIP), pp 28–33. https://doi.org/10.1109/EUVIP47703.2019.8946234
16. Mitrica I, Mercier E, Ruellan C et al (2019) Very low bitrate semantic compression of airplane cockpit screen content. IEEE Trans Multimed 21(9):2157–2170. https://doi.org/10.1109/TMM.2019.2900168
17. Pavlov I (2021) 7z format. https://www.7-zip.org/7z.html
18. Qian X, Wang H, Hou X (2014) Video text detection and localization in intra-frames of h. 264/avc compressed video. Multimed Tools Appl 70:1487–1502
19. Saerens M, Latinne P, Decaestecker C (2002) Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. Neural Comput 14:21–41. https://doi.org/10.1162/089976602753284446
20. Safran (2022) VS1410, VS1510-rugged and compact enertec mission video & data recorders/servers. https://www.safran-group.com/products-services/vs1410-vs1510-rugged-and-compact-enertectm-mission-video-data-recordersservers-harsh-environments

21. Said A, Drukarev A (1999) Simplified segmentation for compound image compression. Paper presented at the IEEE internat conf on image processing, pp 229–233
22. Shkarin D (2002) Ppm: one step to practicality. In: Proceedings of the data compression conference, pp 202–211. https://doi.org/10.1109/DCC.2002.999958
23. Sravani M, Maheswararao A, Murthy MK (2021) Robust detection of video text using an efficient hybrid method via key frame extraction and text localization. Multimed Tools Appl 80(6):9671–9686
24. Tang T, Li L (2019) A low delay rate control method for screen content coding. Multimed Tools Appl 78:28231–28256
25. Wang S, Fu J, Lu Y et al (2012) Content-aware layered compound video compression. In: 2012 IEEE international symposium on circuits and systems (ISCAS), IEEE, pp 145–148
26. Wiegand T, Sullivan GJ, Bjontegaard G et al (2003) Overview of the h. 264/AVC video coding standard. IEEE Trans Circ Syst Vid Tech 13(7):560–576
27. Wien M (2015) High efficiency video coding. Coding Tools Specif 24
28. Xu J, Joshi R, Cohen RA (2015) Overview of the emerging HEVC screen content coding extension. IEEE Trans Circ Syst Vid Tech 26(1):50–62
29. Zouidi N, Kessentini A, Hamidouche W et al (2023) Complexity assessment of the intra prediction in versatile video coding. Multimed Tools Appl 1–20