Semantic Coding of Airplane Cockpit Screen Content for Very Low Bitrate Compression

Iulia Mitrica, Eric Mercier, Christophe Ruellan, Attilio Fiandrotti, *Member, IEEE*, Marco Cagnazzo, *Senior Member, IEEE*, and Béatrice Pesquet-Popescu, *Fellow, IEEE*

***Abstract*—This work addresses the problem of encoding the video generated by the screen of an airplane cockpit. As other computer screens, cockpit screens consists in computer-generated graphics often atop natural background. Existing screen content coding schemes fail notably in preserving the readability of textual information at the low bitrates required in avionic applications. We propose a screen coding scheme where textual information is encoded according to the relative semantics rather than in the pixel domain. The encoder localizes textual information, the semantics of each character are extracted with a convolutional neural network and are predictively encoded. Text is then removed via inpainting, the residual background video is compressed with a standard codec and transmitted to the receiver together with the text semantics. At the decoder side, text is synthesized from the encoded semantics and superimposed over the residual video recovering the original frame. Our proposed scheme offers two key advantages over a semantics-unaware scheme that encodes text in the pixel domain. First, the text readability at the decoder is not compromised by compression artifacts, whereas the relative bitrate is negligible. Second, removal of high-frequency transform coefficients associated to the inpainted text drastically reduces the bitrate of the residual video. Experiments with real cockpit video sequences show BD-rate gains up to 82% and 69 % over a reference H.265/HEVC encoder and its SCC extension. Moreover, our scheme achieves quasi-errorless character recognition already at very low bitrates, whereas even HEVC-SCC need at least 3 or 4 times more bit-rate to achieve a comparable error rate.**

***Index Terms*—HEVC, screen content coding, cockpit content coding, low bitrate, character recognition, semantic video coding, convolutional neural networks, compound video, compound images**

1 Introduction

The cockpit of modern airplanes consists in one or more screens displaying information reported by the plane instruments (e.g., the plane location as reported by the GPS, the fuel level as read by the sensors in the tanks, etc). The typical content of such screens consists in computer-generated graphics (e.g., text, lines, etc.) often superimposed over natural images (e.g., maps, outdoor pointing cameras, etc.). Such examples can be seen in Fig. 1 and will described in details in Section 3.

As the status of the plane instruments is rarely directly accessible due to security or legacy reasons, the cockpit screen is often the only way to access key plane information. For this reason, a video of each cockpit screen is often recorded on-board to be fetched either on a periodic basis or in the event of an accident. Constrains on the long-term recoding storage available on-board require the cockpit video be coded at low bitrates, whereas safety reasons require the textual information to remain intelligible after decoding.

Airplane cockpit video coding can be seen as particularly challenging application of screen content coding. Text and other computer generated graphics found in computer screens yield high-frequency components in the transformed domain usually not found in natural images. Over the past years, a number of schemes for screen coding and in general for coding images with mixed computer-generated and natural contents has been proposed [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. Most of such approaches subdivide the image in computer-generated and natural blocks and encode each block with specific tools (e.g., the former are encoded with lossless schemes). However, the rate-distortion performance of such approaches depends on the block size, on the quality of the block classification scheme and the methods used for coding the objects with different geometry. The recently standardized Screen Content Coding (SCC) extension [**?**] of the H.265/HEVC [**?**] standard includes tools for screen compression. One common point that is shared by all these methods is that the computer-generated areas are encoded as blocks of luminance values. Nevertheless, our experiments show that reconstruction artifacts are unavoidable at the very low bitrates entailed by our application, prompting the research for schemes where text is not encoded in the pixel domain. In the proposed method, the text and the graphical primitives are encoded as such rather than as blocks of pixels. In this sense, we refer to our approach as "semantic". Finally, the complexity constraints of avionic applications make the design of any scheme for airplane cockpit video scheme particularly challenging.

In this work, we propose to encode the computer-generated elements of a cockpit screen according to their semantics and the residual background of the video in the pixel domain. First, candidate characters in the screen are localized via a low-complexity yet effective scheme leveraging position-invariant features typical of cockpit screens. Then, each character is recognized via a Convolutional Neural Network (CNN) [**?**] and its semantics are encoded with a predictive scheme. While the complexity of a convolutional character search may hinder the practical deployability of a CNN in our complexity constrained environment, our low complexity text localization scheme workarounds such complexity. We also explore different fully connected and convolutional network architectures

I. Mitrica, is with Zodiac Data Systems and Télécom ParisTech; A. Fiandrotti, M. Cagnazzo and B. Pesquet-Popescu LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France; E. Mercier and C. Ruellan are with Zodiac Data Systems

to account for different tradeoffs between performance and memory and computational complexity. Second, lines are detected via Hough transform and the starting and ending points coordinates are predictively encoded. Characters and lines are then removed from the screen via pixel inpainting [?] and the residual video is compressed with H.265/HEVC-SCC. The semantics of characters and lines are made available to the decoder as side information to the compressed residual video. At the decoder side, the residual video is decompressed and text and lines are synthesized decoding the side information and superimposed, recovering the original frame. We evaluate our scheme on real cockpit screen video sequences measuring its performance under two aspects: compression efficiency and character readability. Concerning the video coding efficiency, in our test set, the proposed scheme achieves 69% BD-rate savings with respect to H.265/SCC and around 82% savings with respect to H.265/HEVC. That is, our scheme based on coding the semantics of text (and lines) enables 4 to 6 times lower bandwidth than a reference scheme where the such information is encoded in the pixel domain. A similar ratio is found when we compare the minimum rates demanded by the proposed and the reference schemes to assure character readability. Finally, while in this work we focus on intra-coded frames since intra-coded frames affect the most the overall R-D performance of a codec, however in principle nothing prevents extending our proposed scheme to inter-coded frames as well.

The rest of this article is organized as follows. The relevant literature is discussed in Sec. 2 and the necessary background is presented in Section 3. In Section 4 we describe our proposed system for low-bitrate airplane cockpit video coding. Then, in Section 5 we experiment with our proposed system over real cockpit video sequences assessing both the quality of the decoded video and the readability of the reconstructed text. Finally, Section 6 draws the conclusions of this work and outlines future research directions.

2 Related Works

A number of solutions for encoding mixed content images and video (sometimes also referred to as compound images and compound video) such as computer screens have been proposed over time [?]. In the following, we review the relevant literature highlighting the relative limitations that prompted the development of this work.

A first class of schemes focusing on still images compression revolves around the idea of subdividing the image in blocks or layers separating natural contents (e.g., images) from other contents (e.g., text) and encoding each type of content with ad-hoc techniques. Transmission of printed content via facsimile devices is among the earliest applications of compound still image compression. In [?], [?] the authors present a solution for multi-layer coding of compound raster content using a scheme based on block thresholding within a rate-distortion framework. In [?], the authors propose a solution for encoding the content using a block-based segmentation method for differentiate between the objects of a compound image. In [?], the authors investigate the tradeoffs of some practical methods to implement the above ideas. The ITU-T even proposed a standard for printed content compression based multiple-layers image segmentation [?] [?]. For all of the above methods, the rate-distortion gains depend on the coding scheme, on the quality of the image segmentation maps or on the methods used for coding objects with different geometry.

More recently, the problem of compressing moving pictures has emerged due to the rise of videoconferencing and screen sharing applications. In [?], [?], each intra coded block is classified either as pictorial or textual. Then, textual blocks are coded in the pixel domain so to preserve their quality while enhancing the coding efficiency. In [?], the authors leverage temporal masking to remove high frequency components from perceptually less meaningful regions of the video and allocating more bitrate to perceptually meaningful regions. In [?] the authors propose a lossless compression solution for screen sharing for mobile devices on wirelesses networks. The input image is split into blocks and depending on the characteristics of each block, the compression method utilizes predictive coding, edge coding and run coding. In [?] a compression solution is proposed, based on a fast block-based classification algorithm. In the same spirit, the images are divided into blocks, further classified into smooth blocks, text blocks, hybrid blocks and picture blocks. Based on the statistical properties of each, four different coding algorithms are employed to obtain compression gains.

The recently standardized Screen Content Coding (SCC) extension of the H.265/HEVC (High Efficiency Video Coding) standard [?] from the JVT of the ISO/ITU introduced coding tools designed to deal with computer screen characteristics such as limited palette, recurring patterns and sharp edges. Namely, Intra Block Copy (IBC) [?] is an intra-prediction tool leveraging recurrent patterns. It creates a prediction of current prediction unit by finding similar reconstructed block within the causal area of the same picture. Similar in spirit to inter picture prediction, it can be considered as motion compensation within the same frame. Palette mode (PLT) [?] is useful to represent blocks containing a small number of distinct color values. Namely, it explores the discrete tone characteristic of the screen content by signalizing the pixel values directly rather than using prediction or transform-based methods. A number of improvements to the SCC extension has even been proposed. For example, in [?] an improved String Matching (SM) scheme going beyond IBC and PLT block matching able to encode a wider range of patterns with different sizes and shapes is proposed. In [?], a residual differential pulse code modulation (RDPCM) coding technique using a weighted linear combination of neighboring residual samples is proposed. In [?], intra and inter coding transform skipping for improving compression efficiency are proposed; while effective for pure screen content, such approach is less effective for compound images case which is our main case of interest. Despite the success of the SCC HEVC extension and its derivatives, our experiments revealed that at very low bitrates imposed by our application artifacts around text are unavoidable, motivating the semantic coding scheme proposed in the present work.

For the sake of completeness, we briefly discuss also frame

buffer compression schemes. Such schemes, unlike traditional hybrid codecs such as H.265/HEVC, are designed to reduce the bandwidth of raw video between hardware devices (e.g., video decoder and framebuffer or framebuffer and display) while keeping the encoding-decoding complexity bounded [**?**] [**?**]. In particular, the Video Electronics Standards Association (VESA) has recently standardized a protocol for quasi-lossless framebuffer [**?**]. Such schemes are appealing for cockpit video compression systems, but target a different part of the video system (*i.e.* the link between the decoder and the screen) than ours (link between encoder and decoder). By the way, any framebuffer compression scheme could be plug in after our codec, but dealing with this topic is out of the scope of this paper

With respect to the above literature, our proposed scheme aims at improving both over earlier compound methods and more recent standard technologies. While we share with compound methods the idea of separately encoding computer graphics, we encode the computer graphics in their semantic domain rather in the pixel domain to avoid reconstruction artifacts irrespectively of the residual video coding rate. Concerning the residual video, our proposed scheme fully enjoys the benefits of standardized coding technologies while removal of graphical elements from the residual video significantly improves the relative efficiency.

3 Background

This section provides the background on airplane cockpit video and overviews (convolutional) artificial neural networks for text detection and reading.

3.1 Airplane Cockpit Video

The cockpit of a modern airliner is composed of one or more computer screen providing specific information to each of the pilots in the cockpit. Towards the efficient compression of cockpit video contents, which is the goal of the present work, we categorize cockpit screens into two main classes.

The first class is shown in Fig. 1(a), 1(b), 1(c) or 1(f) (*class A* sequences in the rest of this work): they are characterized by computer-generated graphics (text, lines, etc.) superimposed over a natural background image (typically, a video captured by an exterior-mounted camera). For example, the background of Fig. 1(a) is a color image captured in the visible light spectrum, whereas Fig. 1(b) has grayscale background captured in the infrared band or in low-light conditions (Fig. 1(c)).

Such screens typically assist the pilot in some specific tasks (e.g., let the second pilot inspecting the ground for obstacles on the take off/landing stripe). Due to the unpredictable background appearance, overlaid text usually has contour to improve its readability (e.g., black text with white contour).

The second class of cockpit screens we consider is shown in Fig. 1(d) and Fig. 1(e) (*class B* sequences). Such class of screens is characterized by complex computer generated graphics over a monochrome background. Sometimes referred to as *glass cockpits*, they replace a number of discrete board instruments such as compass, active and passive radars, fuel gauges, etc. Key plane and flight information being involved, complex time-variant color codes are used to focus the pilots' attention over the relevant data, whereas a black background is typically used to facilitate visual focus.

Extracting and separately coding the semantics of on-screen computer generated graphics, which is the pivotal idea of our proposed video compression method, is complicated by a number of factors.

First, contents difference across screens (e.g., screens with natural or monochrome background) require deploying the coding tools tailored to the specific screen content. However, in modern airliners the screen content may change when pilots swap roles (e.g., the second pilot takes the plane control from his seat), thus the requirement to handle content whose aspect may vary on a frame-by-frame basis complicates the system design.

Second, the lack of a standard for information reporting implies that the screen appearance may vary a lot within the same airliner. While character detection is somewhat simplified by key information having fixed on-screen position and relying on fixed-pitch fonts to facilitate data spotting, font typeface, color and size vary a lot between screens, demanding a character reading technology robust to such variations.

Finally, avionic operations impose constraints on power consumption and heat dissipation that result in a cap to system complexity that further complicate the task.

3.2 Character Recognition via CNNs

Convolutional Neural Networks (CNNs) have emerged as the cornerstone to solve a number of computer vision problems such as character recognition [**?**]. CNNs are feed-forward, multiple-layer, artificial neural networks that may be described as a feature extractor stage followed by an inference stage [**?**]. The feature extraction stage includes a number of convolutional layers, each layer encompassing multiple learnable filters. Each filter is convolved with the layer input(s) with a sliding window scheme and activates upon detection of one specific feature. The output of each filter is processed by some non-linear activation functions such as ReLUs [**?**]. Robustness to changes in object position is achieved discarding absolute spatial information via subsampling (pooling layers). Each convolutional layer learns to detect features of decreasing resolution yet increasing semantics. The output of the feature extraction stage is processed by one or more fully connected layers performing, for example, image classification. Finally, the last layer of the network provides the desired network output such as the image class probability distribution. CNNs are typically trained end-to-end via error gradient backpropagation with a fully supervised approach, relieving the system designer from the burden of designing ad-hoc feature extraction algorithms.

Concerning our problem of extracting the semantics of the text from an airplane cockpit screen, CNNs are particularly appealing due to their performance in character reading [**?**]. CNNs can be used to localize single characters in an image with a sliding window scheme that makes efficient reuse of convolutions (*convolutional character search*). More recently, CNNs have been shown suitable to spot text at the level

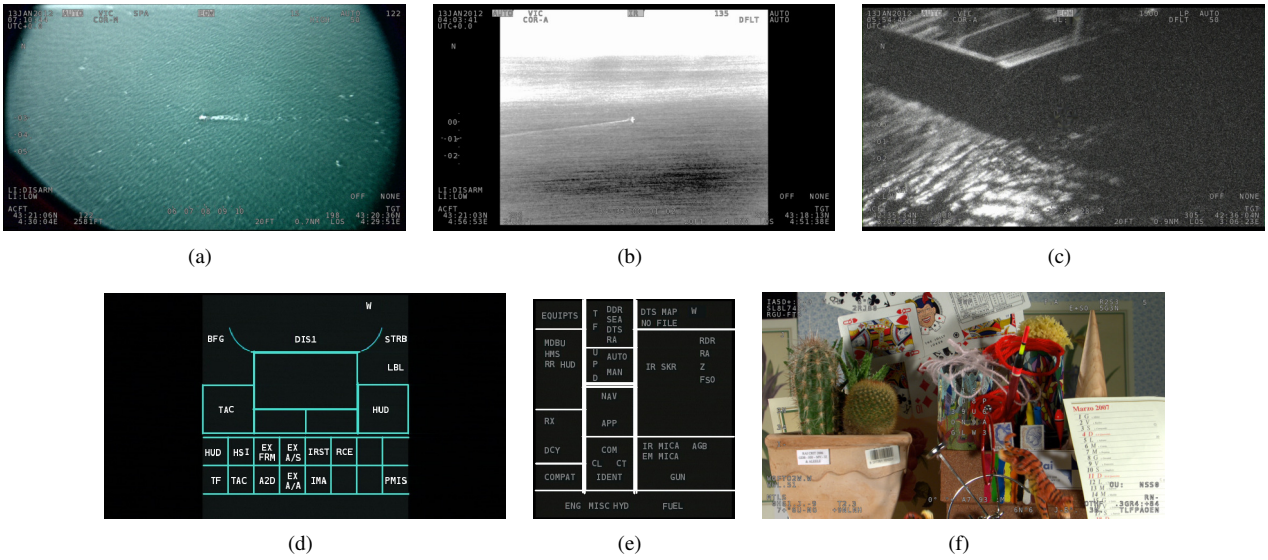


Fig. 1. Example of different types of airplane cockpit screens. First row - Left (Fig. 1(a)): class A - natural light with color background; Center (Fig. 1(b)): class A - infrared vision with grayscale background; Right (Fig. 1(c)): class A - low-light vision with grayscale background. Second row - Left and Center(Fig. 1(d) and Fig. 1(e)): class B - computer-generated background; Right (Fig. 1(f)): class A - synthetic content.

of entire words [**?**] and to read text at arbitrary positions in still images [**?**]. Concerning moving pictures, in [**?**] the authors survey several methods suitable for text detection, tracking and recognition in video. The complexity of such text localization approaches increases however with the image resolution, which may be prohibitive in our complexity-constrained avionic application. To this end, while we rely on a neural network for character recognition, we devise a simple scheme for text localization that leverages position invariant features in cockpit screens and keeps the overall complexity of extracting the text semantics bounded, as described in the next section.

4 Proposed Method

This section describes our scheme for airplane cockpit screen semantic coding, as illustrated in Figure 2. Text is first localized exploiting some position-invariant screen features (Sec. 4.1), then characters are recognized using a CNN (Sec. 4.2) and predictively encoded (Sec. 4.3). Lines are similarly detected and encoded (Sec. 4.4). Finally, text and lines are removed from the frame and the residual video is encoded via H.264/HEVC-SCC (Sec. 4.5). At the decoder, the residual video is decoded and text and lines are synthesized avoiding reconstruction artifacts.

4.1 Character Localization

Characters are detected and localized in the screen with a low-complexity yet effective approach as follows. The block diagram of the procedure is illustrated in Fig. 3.

Identifying character pixels is complicated by the fact that their aspect depends on the background complexity (e.g., natural-background class A screens in Fig. 1(a) show text with contour, whereas text in monochrome-background class B screens in Fig. 1(d) has no contour).

As a first preliminarily step, we perform a *coarse classification* of the image (Fig. 3) using color histograms to find if the input image is synthetic (only computer graphics) or natural (i.e, a compound of computer generated graphics overlaid on a natural image). To start with, we draw some tiles from the input frame and compute the corresponding histogram. Then, each tile is either labeled as synthetic (usually because of monochrome background) or natural (corresponding to natural background) as follows. In order to decide the label for each tile, a condition is imposed. The histograms are sorted in a decreasing order of frequency. Further, starting with the most frequent bin we count the number of bins that correspond to, at least 90% of the total number of pixels. If the number of bins is bigger than 20% of total number of bins, then the tile is labeled as natural. Otherwise, if the histogram is spiky, the tile is labeled as synthetic. Fig. 4(a) shows two examples of histograms corresponding to these typical cases of the content. Finally, if at least one of these histograms is flat enough, the input image is classified as natural, otherwise it is classified as synthetic.

As a second step, the *coarse classification* output is used to decide which thresholding algorithm shall be used. The *threshold* block (Fig. 3) takes as input the image and produces as output a *threshold map* TM exemplified in Fig. 4(b), which is a binary map indicating the possible positions of characters. If the coarse classifier decides that the input image is synthetic, like the one in Fig. 1(d), the Otsu [**?**] thresholding scheme is used. If the image is natural (like Fig. 1(b)), we exploit the fact that characters have an outline to ease the reading by the pilot, *e.g.* the characters are light-gray with a black outline. Then, our thresholding algorithm works as follows: first, we find all the pixels whose color is close to the character's color, *i.e.* whose hue, saturation, and value differ from the character's ones less than some suitable deviations (whose values are set empirically). The result is a bitmap M_1. Its value in a given position is high if and only if the image

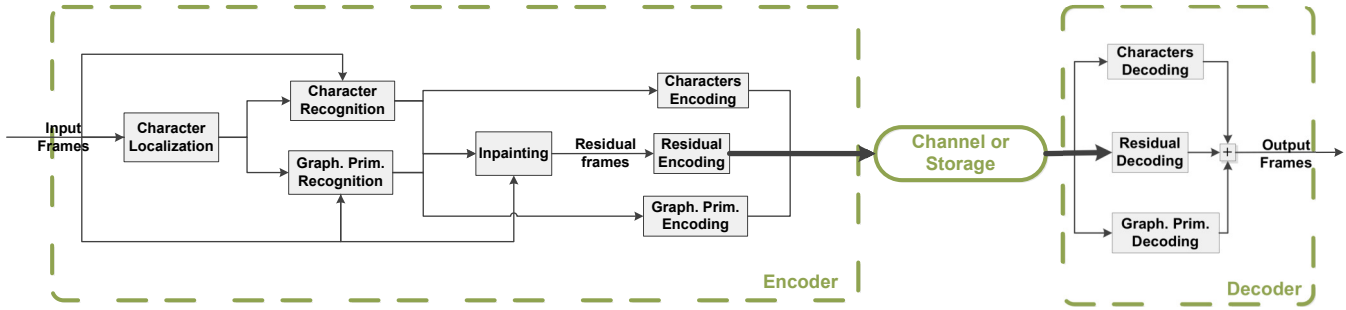


Fig. 2. Architecture of the proposed scheme for encoding airplane cockpit video at low bitrates: the semantics of computer generated graphics (text, lines) are relayed to the decoder separately from the compressed residual video.

color in that position is close to the character's color (light gray). Likewise we produce a second bitmap M_2 that is high in all and only positions where the input image's color is close to the outline color (black). These maps have usually many false positives, but most of them can be eliminated knowing that characters are localized in areas where both maps have high bits. Therefore, we first apply a morphological dilate operator with a suitable structuring element (circle of radius 4), obtaining respectively $\overline{M}_1$ and $\overline{M}_2$. Finally, the *threshold map* TM for the natural image is the bit-wise AND of these two bitmaps. This corresponds to pixels in whose neighborhood (defined by the structuring element) there are both character and outline colors.

As a third step, objects within the *threshold map* are identified via Connected Components Analysis (CCA). *CCA* (Fig. 3) clusters the white pixels in the binary map assigning the same label to pixels in the same *neighborhood*. Namely, we consider an 8-pixel neighborhood, to gather the pixels along any face and corner. Each maximal labeled region defines a connected component. The output of such process is the *segment map* SM exemplified in Fig. 4(c), where each cluster of computer graphics pixels is represented with its bounding box. Typically, the SM shows several false positive. In order to reduce them, we use *screen classification* and *reference maps*.

As a fourth step, the cockpit screen is classified among a set of possible screens options as follows. Let each possible class of cockpit screen have associated one *reference map* RM as in Fig. 4(d). The reference maps are bitmaps that are computed off-line. In a given position a RM is high if and only if a character may appear there in that screen type. This

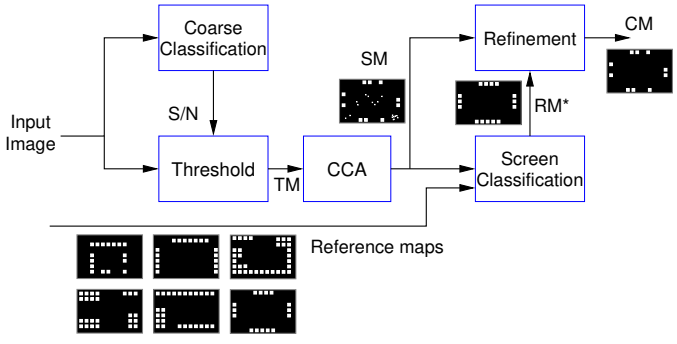


Fig. 3. Character localization procedure.

is shown in Fig. 1(a) via the relative bounding box. Since for a given cockpit type characters have time-invariant positions to facilitate the pilot in localizing key information, such features are highly distinctive of each screen type. However, we note explicitly that first, we do not know in advance which the screen type is, and second that the RM cannot directly be used for character localization since typically on a single image we will not find characters in *all* the positions where the RM is high. So first we have to perform screen classification: for each *reference map* representing a screen class, we perform a bit-wise AND operation between the *segment map* and the *reference map* and we count the number of high pixels in the resulting bitmap. The screen type whose *reference map* has the highest pixel score RM^* is selected as the one of the current image. In other words, we identify the screen type among a set of options simply by counting the high bits. Our experiments showed that, when the number of possible screen classes is up to 12, the *screen classification* is always correct. Finally, we perform a box-wise AND operation between the highest-scoring *reference map* (referred to as RM*) selected by the screen classifier and the *segment map*, in order to get a noiseless *characters map* CM. For example, the output of the box-wise AND between the *segment map* in Fig. 4(c) and the *reference map* in Fig. 4(d) is the *characters map* in Fig. 4(e). Notice how the *characters map* includes a bounding box of the actual character size for each character actually present in Fig. 1(a). The *characters map* is then given as input to the character recognition block. In this manner we define a character localizer, that fulfills the low-complexity constraints of our target application. It is able to generate the coordinates of the position of each bounding box, which is further classified as detailed in Sec. 4.2. In the end, the associated table of characters, suitable for coding, is obtained as it is elaborated in Sec. 4.3.

4.2 Character Recognition

This section discusses the three neural network architectures for on-screen character recognition in Figure 5. Each architecture offers a different trade off between performance and complexity, allowing to address the different types of constraints that avionic applications must undergo.

The first architecture is *LeNet300* [**?**], a simple fully connected network composed by two hidden layers with 300 and 100 units (neurons) respectively with sigmoid activation functions

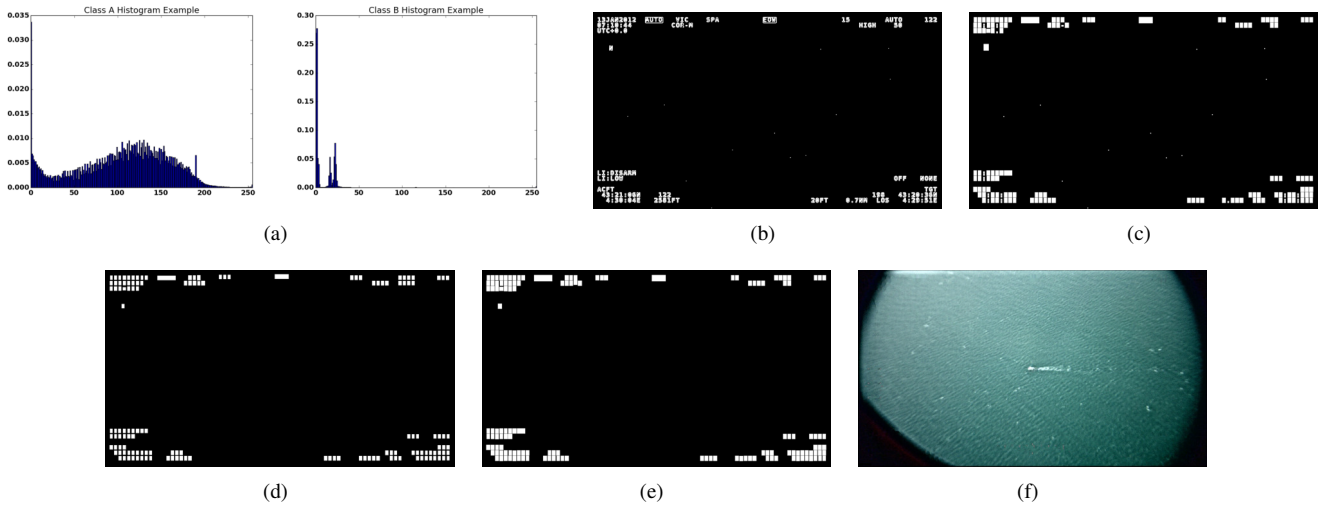


Fig. 4. Intermediate processing steps for the class A image in Fig. 1(a). First row - Left (Fig. 4(a)): Histograms examples; Center (Fig. 4(b)): Threshold map; Right (Fig. 4(c)): Segment map. Second row - Left (Fig. 4(d)): Reference map; Center (Fig. 4(e)): Predicted Characters map; Right (Fig. 4(f)): Residual after character inpainting.

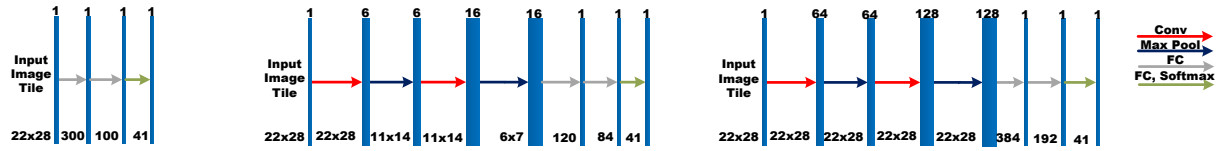


Fig. 5. The three neural networks architectures considered for on screen character recognition. Left: LeNet300; Center: LeNet5; Right: LeNet5+.

and one output layer.

The second architecture is the well-known *LeNet5* [**?**], a convolutional architecture with two convolutional layers and three fully connected layers. The two convolutional layers include 6 and 16 5×5 filters each with sigmoid activations and followed by max-pooling feature map subsampling. The first two fully connected layers include 120 and 84 units respectively with sigmoid activations.

The third architecture is an improved LeNet5 (*LeNet5+*, in the following) that leverages recent advances in deep neural architectures [**?**] such as ReLU activations [**?**] and an increased number of convolutional filters.

All three architectures include C=41 units in the output layer (i.e., they classify the character in an input image according to C=41 labels). A multinomial logistic regression (i.e., *SoftMax*) layer finally yields a *one-hot* output representing the character class probability distribution.

The networks are trained over character samples extracted from the annotated video sequences. With reference to the sequences in Fig. 1(a) and 1(b), we extract 22×28 character crops, matching the size of the on-screen characters. The characters we extract account for 10 digits, 31 letters, punctuation marks and symbols for a total of 41 character classes plus one background class. Characters class samples are extracted with the aid of the reference masks annotated with character labels. Such set of samples is the augmented by randomly shifting each character by either zero, one or two pixels in each directions, so that the network learns to be robust to small errors in character localization. In total, we generate about 8000 samples for each character class. The background class samples are cropped at random positions from the video sequences so that they do not overlap with the reference mask, i.e. they represent the background video. Showing the network a large number of background class samples is in fact key to train it to reject false positives, i.e. background elements in the thresholding mask that may had been mistaken for text. The extracted samples are divided into 80 % as training samples and 20 % as test samples for the purposes of our tests.

The networks are trained with a fully supervised approach as follows. Let x indicate a training sample, let y_i indicate the i-th network output (i.e., the predicted probability that x belongs to the i-th class) and t_i the corresponding target class score. The network is trained minimizing the cost function $J(w, y, t) = -\sum_{i=1}^{C} t_i \ \log(y_i) + \lambda R(w)$, where w represents the network parameters (weights and biases). Finally, $R(w)$ represents a regularization term that prevents the network from overfitting to the training images and is defined as the squared l_2 norm of the network weights, whereas λ is the relative regularization factor. Such cost function is minimized via Stochastic gradient descent (SGD) and the gradients of the error function with respect to the network parameters are computed via gradient backpropagation [**?**]. Practically, we update the network parameters with batches of 128 samples, i.e. we update the weights according to error gradients averaged over 128 training samples. Concerning the training procedure, the initial learning rate is set to 10^{-2} the training terminates when the error on the validation set stops decreasing for three consecutive epochs.

Table I reports some preliminary experiments on the performance-complexity trade offs of the three architectures.

Performance is reported in terms of character recognition accuracy, i.e. ratio of correctly classified characters. Memory complexity is reported as number of learnable parameters and computational complexity as number of Multiply-Accumulate (MAC) operations [**?**]. The three architectures show all accuracy in excess of 99.7%, i.e. they enable quasi-flawless recognition of computer-generated characters. LeNet5+ shows best performance, LeNet300 shows the lowest computational complexity, whereas LeNet5 shows the lowest memory footprint.

TABLE I
ACCURACY-COMPLEXITY TRADEOFF OF THE THREE ARCHITECTURES CONSIDERED FOR ON SCREEN CHARACTER RECOGNITION.

	LeNet300	LeNet5	LeNet5+
# Params	215.8k	**60k**	30M
MACs	**219k**	679k	32.7M
Accuracy [%]	99.73	99.84	**99.99**

4.3 Character Coding

For each frame, we need to encode the characters recognized by the neural network. More precisely, the character recognition outputs a list of N_C recognized characters; each of the N_C entries in the list is a tuple, composed by the bounding box position (horizontal and vertical coordinate of the top-left pixel), the recognized character (letters, digits and a few typographical signs such as dot, comma etc.) and possibly other characteristics as font and color. In the current version of our scheme, only one font and two colors are possible.

If we encode the character list using a simple constant-length code, it would require $N_C\left(\lceil\log_2 W\rceil + \lceil\log_2 H\rceil + \lceil\log_2 S\rceil + \lceil\log_2 F\rceil\right)$ bits per frame, where W and H are the frame's width and the height (in number of pixels), S is the number of possible symbols and F is the number of font's colors and shapes. For example, in class A sequences, we have $W = 1920$, $H = 1080$, $S = 41$ (26 uppercase letters, 10 digits and a 5 punctuation marks) and $F = 2$ (one font shape in two possible colors), which makes 29 bits per character. Typical values of N_C are around 170 for Class-A and 70 for class-B sequences: this would result in about respectively 5 kbits and 2 kbits per frame, which is non-negligible for very low bitrate use-cases. Therefore we need to resort to some lossless coding tools.

A first simple solution is to use some universal, dictionary-based lossless coding algorithm, such as LZW [**?**] and its variation. However these algorithms are typically not efficient when the number of symbols to be encoded is as small as in our case for a single image. Therefore, if we want to provide an effective coding of the text in "Intra" fashion (i.e. without exploiting temporal redundancy and thus allowing independent decoding of images), we must resort to some ad-hoc encoding scheme which exploits the regularity of the character list.

First, in cockpit video sequences, characters are typically aligned in rows and are equally spaced. Therefore, it could be convenient to differentially encode their horizontal and vertical coordinates, since the probability distributions of the coordinate differences are very spiky. For example, we have observed that vertical coordinates of successive characters are often identical or they differ by $\pm$ one pixel (accounting for some possible noise in the character localization). Therefore we encode the vertical coordinate v_n of the n-th character as follows. If n=1 (the first character of the image), v_n is encoded using $\lceil\log_2 H\rceil$ bits (since $v_1 \leq H$). For $n > 1$, we first compute $d_n^v = v_n - v_{n-1}$. If $d_n^v \in \{-1, 0, 1\}$, we first signal that v_n is encoded differentially, using a flag "0"; then the value of d_n^v is encoded using a simple variable-length code (VLC), see Tab. II. Otherwise we encode the flag "1" (which means that v_n is encoded, rather than d_n^v), followed by the binary representation of v_n on $\lceil\log_2 H\rceil$ bits.

Similarly, the horizontal coordinate of consecutive characters often differ by constant number of pixels corresponding to the character width, let it be c_w. Therefore, the horizontal coordinate h_n of the n-th character is encoded as follows. If n=1, we encode the binary representation of h_n on $\lceil\log_2 W\rceil$ bits. If $n > 1$ we compute $d_n^h = h_n - h_{n-1}$. If $d_n^h = \in \{c_w - 1, c_w, c_w + 1\}$ (which happens most of the times), we encode a flag "0" to signal the differential encoding, followed by d_n^h encoded with VLC; if $d_n^h = \notin \{c_w - 1, c_w, c_w + 1\}$ we encode a flag "1" to signal the encoding of h_n, followed by the $\lceil\log_2 W\rceil$ bits of the binary representation of h_n.

The encoding strategy of the character coordinates is resumed in Tab. II.

TABLE II
CODING OF THE CHARACTERS VERTICAL AND HORIZONTAL COORDINATES FOR $n > 1$. THE FIRST BIT INDICATES WHETHER THE COORDINATE DIFFERENCE OR THE COORDINATE VALUE IS ENCODED. IN THE FIRST CASE, THE COORDINATE DIFFERENCE IS ENCODED USING A VLC, WHILE IN THE SECOND THE COORDINATE VALUE IS ENCODED USING A FIXED LENGTH CODE.

d_n^v	bitstream	d_n^h	bitstream
0	`0`	c_w	`0`
$+1$	`100`	$c_w{+}1$	`100`
-1	`101`	$c_w{-}1$	`101`
other	`11`, v_n on $\lceil\log_2 H\rceil$ bits	other	`11`, h_n on $\lceil\log_2 W\rceil$ bits

This simple strategy allows to reduce the coding cost for class-A sequences to less than 10 bits per character in average, *i.e.* a rate reduction of about 66 %. This can be achieved since, for most of the characters, we will only use 2 bits for the coordinates, 6 for the symbol and 1 for the font.

We explicitly observe that in the proposed method, the characters are encoded in an "INTRA" fashion, that is, without considering temporal redundancy. Taking into account text from previous images could certainly provide further compression; however this issue is left for future works.

4.4 Graphical primitives recognition and coding

As was mention before, cockpit screens can contain graphical elements such as lines, circles *etc.* Even though the proposed method could be extended to general graphical primitives, in this work, we only consider sequences containing horizontal and vertical lines, in particular what we called Class B frames (see Fig. 1(d)). In such a context of computer generated images, horizontal and vertical lines can

be recognized quite easily and effectively, using for example such algorithms as the Hough transform [**?**]. In particular we use the line detector described by Matas *et al.* [**?**], which also provides the starting and ending point of each detected line.

The line detection algorithm outputs a list of four coordinates, representing the starting and ending points (the extreme points) of each of the N_L detected lines. Similarly to character encoding case, a plain, fixed-length coding of the lines would require $2N_L\left(\lceil \log_2 W \rceil + \lceil \log_2 H \rceil\right)$, which for a typical class B sequence amount to 1000 bits, i.e. roughly 40 bits per line. Additionally, other bits could be required to encode the line color and thickness. In the current version of our scheme, we allow only for two colors and one default thickness value, amounting for one additional bit per line.

As in the previous case, we could resort to existing or ad-hoc lossless coding algorithms to reduce this cost. We developed a prediction based coding algorithms which exploits the fact that, in our sequences, many lines share the same horizontal (or vertical) starting and ending point. First, the list of lines extreme points is lexicographically sorted. The first line (*i.e.* the first set of four coordinates) is encoded in fixed-length fashion. For each of the following lines, each coordinate is compared to the correspondent coordinate of the previous line. If they are equal, only a one-bit flag is written in the encoded stream; otherwise, a one-bit flag followed by $\lceil \log_2 H \rceil$ or $\lceil \log_2 W \rceil$ bits are written. A variant of this algorithm allow for a $\pm$one-pixel tolerance on points coordinates, meaning that two-bits flags are needed but more points can be encoded in differential mode. This variant allows for a $14\,\%$ rate reduction with respect to the fixed lenght coding, and is retained for our codec.

We observe that for more complex sequences than class-B ones, we would need more sophisticated algorithms for graphical primitives recognition and encoding. They could be based on classical computer vision techniques (*e.g.* the Hough transform for circles) or on DL classifiers. However, these issues, as well as the problem of exploiting the temporal redundancy for graphical primitives, will be explored in future works.

4.5 Residual Coding

Eventually, computer-generated graphics and text are removed from the video, filling the resulting gaps via inpainting [**?**] and encoding the residual video. The visual quality of the inpainted areas bears little importance, since such areas are small and, at the decoder side, they will be overlaid by the synthesized characters. Conversely, it is important that ringing artifacts are avoided because they can jeopardize the compression efficiency of the residual video. On the other hand, the computational complexity of the inpainting method process should meet the requirements of a real time video encoding. Thus, we resort to the Navier-Stockes [**?**] method The residual frame is characterized by smooth color distribution in the inpainted areas as shown in Fig. 4(f), which is more suitable for compression lacking the high frequencies in the original frame. Finally, the residual video is coded using the SCC extension of the HEVC/265 codec. We chose to use the extension to code the residual because, in this version, we have graphical elements that are not identified, like arcs of circle (*i.e.* in figure 1(d)), and those must be found in the residual frame, at a given quality, at the decoder side.

5 Experimental Results

We validated the proposed compression scheme by comparing it with three references: the H.265/HEVC codec (*HEVC* for short), its SCC extension (*SCC*) and the screen content coding technology DjVu [**?**] (*DjVu*). As HEVC codec, we used the version HM-16.14 of the standard reference software; for SCC we used the extension SCM-8.3; for DjVu we used DjVu Solo 3.1 (compiled for Windows).

Six airplane cockpit video sequences were used in our experiments, *i.e.* those shown in Fig. 1. We recall that the first three sequences (class-A) are characterized by text and symbols superimposed on natural background (video acquired with a camera installed outside of the airplane, either in the visible light spectrum as in Fig. 1(a) or in the infrared spectrum as in Fig. 1(b), or in low-light conditions as in Fig. 1(c)) at full HD resolution (1920×1080), 24 fps. Another novel synthetic sequence was generated by superimposing computer-generated text over a different background video, as in Fig.1(f). The background video is part of the H.265/HEVC test sequence and is known as "Cactus" and has both a very complex level of detail that stresses the video encoder and a lot of background clutter in the form of text that stresses the character detector. Thus, we obtained a synthetic sequence at full HD resolution at 24 fps. Conversely, another type of sequences (class-B, Fig. 1(d) and 1(e)) contains complex computer-generated patterns at 720×576 resolution, 24 fps. For each of the three considered schemes, we explore a wide range of video qualities, even though most focus is given to the low-bitrate part of it. We consider QP from 20 to 45 with steps of 5 and from 45 to 51 with steps of 1. The proposed and the reference codecs all work in an intra-only configuration; we leave for our future research evaluating the impact of temporal prediction.

After encoding and decoding the six test sequences, [1] we evaluate the coding rate, the objective video quality (or distortion) and the impact of compression on the character readability. As for the rate-distortion performance, the results are easily obtained in terms of rate-PSNR curves and of Bjontegaard's metric: this is discussed in Section 5.1. As for the character readability, things are more delicate. For the proposed scheme, since the text is encoded as such (and not as blocks of pixels), we can assume that this information is available to the decoder at any rate beyond a given minimum. On the contrary, in the case of the HEVC and SCC reference schemes, characters are encoded with the rest of the video, thus reconstruction artifacts at the decoder are to be expected and will affect readability. This issue is discussed in Section 5.2.

5.1 Rate-distortion performance

[1] A part of the decoded video sequences are made available to the reviewers of this manuscript as supplementary material.

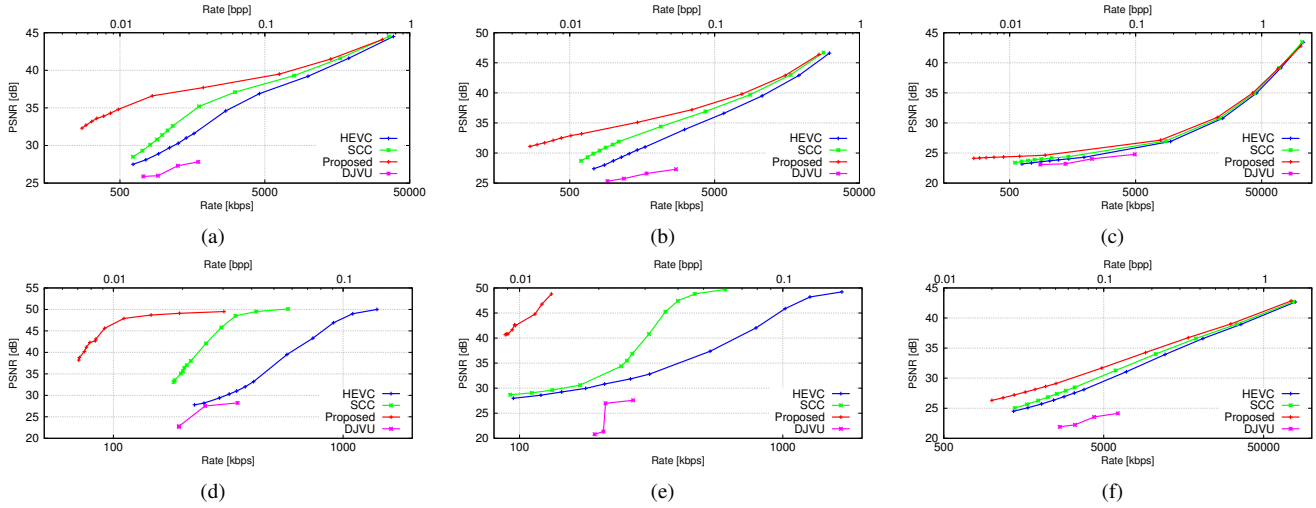


Fig. 6. PSNR vs. video bitrate. First row - Left: class-A color (Fig. 1(a)); Center: class-A gray-0 (Fig. 1(b)); Right: class-A gray-1 (Fig. 1(c)). Second row - Left: class-B 0 (Fig. 1(d)); Center: class-B 1 (Fig. 1(e)); Right: class-A synthetic (Fig. 1(f)).

The rate-distortion curves of the four methods (the proposed one and the three references) are shown in Fig. 6 for the six video sequences. From these figures, a few interesting facts emerge.

First, the proposed scheme is able to attain a minimum rate much smaller than HEVC and SCC. Class-A sequences can be encoded with as few as 0.005 bpp at QP 51, while SCC and HEVC cannot go below 0.011 and 0.013 bpp respectively. For the given resolution and frame rate, they correspond to 258 kbps (proposed), 547 kbps (SCC) and 649 kbps (HEVC). Moreover, for the proposed method only 41 kbps (i.e., roughly 1700 bits per frame, 1/6 of the total rate) are used for encoding the text and the graphical primitives, while the rest of the rate is dedicated to the residual video. For smaller QPs, only the residual rate increases, meaning that, already at 600kbps it takes more than 93 % of the total rate. At higher rates, the coding cost of the text becomes nearly negligible. For the class-B sequences similar considerations hold.

Second, as was expected, DjVu performs below HEVC encoder for all test sequences. Thus, we do not consider further this scheme for characters readability evaluations.

Third, the proposed codec outperforms the references at all rates, and above all at low and very low rates. This is already visible in Fig. 6, but for a more precise comparison we also compute the Bjontegaard metrics (BD-Rate and BR-PSNR) [**?**] for the two class-A sequences, see Tab. III. At low bitrate, we gain no less than 50 % with respect to SCC and up to more than 60 % with respect to HEVC. Bjontegaard metrics cannot be computed for class-B sequences since the rate overlap is too small. However, we can see from the RD curve in Fig. 6(d), that in order to achieve a PSNR of about 40dB, SCC needs three times as rate as the proposed method and HEVC more than seven times. At higher PSNRs, gaps are somewhat smaller but still very remarkable.

These large gains are not very surprising given that removing the text and the graphical primitives from video leaves a very smooth residual to encode, while the semantically relevant information is encoded with the *ad hoc* algorithms described in Sect. 4.3 and 4.4. To gain further insight about the reason of our gains, we show in Fig. 7 the partitions of one frame from a class-A sequence together with the histogram of

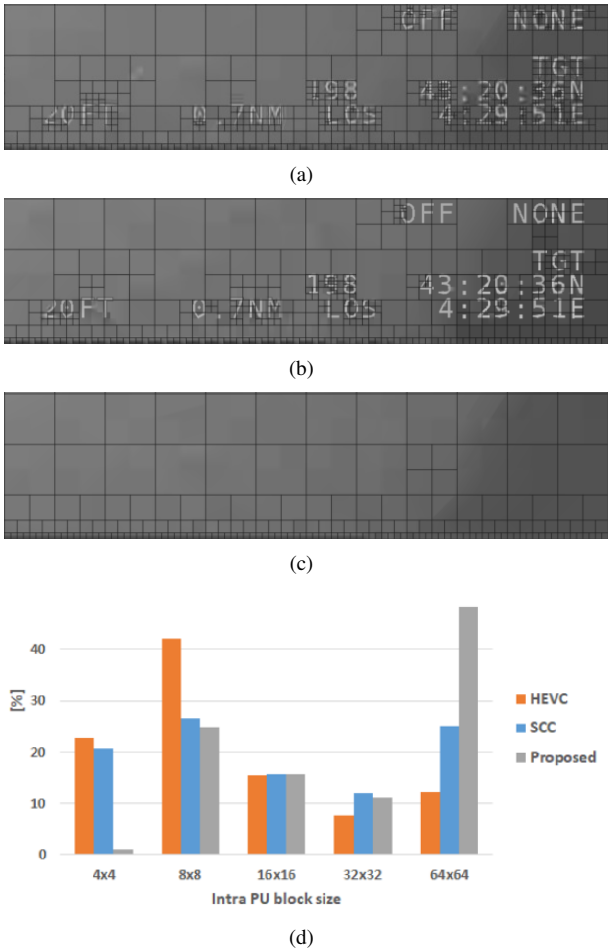


Fig. 7. PUs for the first frame of the class-A color sequence for QP 50. Top: HEVC; Middle: SCC; Bottom: proposed scheme. The corresponding PUs size distribution is reported in the last figure.

TABLE III
BJONTEGAARD METRICS FOR CLASS-A SEQUENCE. THE BD-RATE IS THE AVERAGE RATE CHANGE FOR THE SAME QUALITY. THE BD-PSNR IS THE AVERAGE PSNR CHANGE FOR THE SAME CODING RATE.

	High Quality $QP = [40, 35, 30, 25]$		Low Quality $QP = [50, 45, 40, 35]$	
	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
Prop.-SCC, color	1.04 dB	-37.66 %	4.59 dB	-69.04 %
Prop.-HEVC, color	1.86 dB	-53.25 %	7.12 dB	-81.98 %
Prop.-SCC, gray-0	0.84 dB	-20.43 %	2.18 dB	-52.69 %
Prop.-HEVC, gray-0	2.06 dB	-39.58 %	4.30 dB	-72.36 %
Prop.-SCC, gray-1	0.32 dB	-5.55 %	0.43 dB	-39.98 %
Prop.-HEVC, gray-1	0.63 dB	-10.2 %	0.75 dB	-46.83 %
Prop.-SCC, synth.	0.74 dB	-16.34 %	1.5 dB	-30.46 %
Prop.-HEVC, synth.	1.3 dB	-26.4 %	2.42 dB	-42.96 %

the Intra PU block size. The frame is encoded at QP=50 and the three partitions selected by the three codecs are shown. Removing the text not only allows a better compaction of the energy in the low-frequency transform coefficients, but also allows to use more often the largest PU size, resulting into a further increase of compression efficiency.

We also report in Fig. 8 and 9 some details of two decoded images from the class-A sequences, in order to visually appreciate the gains provided by our method at very low bitrate. The frames were encoded at QP 50 for the three schemes. For the image in Fig. 8, the resulting coding rates are 0.015 bpp for HEVC (left), 0.012 bpp for SCC (center) and 0.005 bpp for ours (right), and the associated PSNRs are respectively 27.4 dB, 28.7 dB and 31.1 dB. Similar values are obtained for the image in Fig. 9. Besides the raw rate-PSNR values, the figures show clearly how the reference method may introduce some severe compression artifacts on the text, in particular when the background is non uniform. SCC mitigates somewhat the artifacts, yet some characters are very difficult to read or, even worse, simply disappear.

As a further benefit from our scheme, the rate saved by removing high frequency content is used to better represent the background. Fig. 10 exemplifies decoded images for the class-A color sequence. In this case, the three methods use approximately the same rate, that is 0.018 bpp. With the HEVC and SCC schemes (top and center, respectively), sea texture and important details in the background such as the islands at the horizon line are either lost or hardly perceptible, whereas our proposed scheme preserves such details despite the very low coding rate.

5.2 *Character readability*

As highlighted in the previous section, the reference schemes suffer from coding artifact affecting the readability of the text at low bitrate, while this problem is circumvented by the propose scheme by the text recognition carried out at the encoder on the original video. Since the recognized text is encoded as such (and not in blocks of pixels), we can consider that it is always perfectly available at the decoder in our scheme, provided that the available bitrate is at least equal to that needed for text coding. However, this threshold is very low as highlighted in previous sections, being in the range of $5 \div 7 \cdot 10^{-3}$ bpp. For the reference HEVC and SCC schemes, the minimum rate for perfect character readability should be quite higher, as one can deduce from Fig 8 and 9. However, an accurate evaluation of this threshold would require an extensive campaign of subjective tests, which would be out of the scope of this work. However, an estimation of the compression impact on character readability can be obtained much more easily by partly exploiting the ideas of automatic character recognition shown in Sect. 4.2.

More precisely, and uniquely in order to estimate the compression degradation of the reference HEVC and SCC schemes, we can evaluate the character readability at the decoder side using any of the CNN architecture described in Sec. 4.2. In other words, we rely on the character recognition accuracy of our CNN as a proxy for otherwise resource-consuming subjective text readability evaluations.

We highlight that in this experiment the CNN is not used to evaluate the proposed method (which, as mentioned at the beginning does not suffer from artifacts on text data) but only the HEVC and SCC methods.

Of course, we need to retrain the CNN over character samples that resemble the conditions at the receiver, that is, samples obtained by compressed video sequences. While one different training per QP could be considered, we found that 4 training sets (with uncompressed samples and with samples compressed at QPs 25, 40 and 51) are sufficient to evaluate the text degradation. For example, we observed no recognition accuracy increment if the compressed frames with QP 30 are analyzed with a network trained with QP 30 with respect to a training with QP 25.

Once the four networks were trained as described, we used them to evaluate the accuracy of character recognition on the videos decoded with the HEVC and SCC reference schemes. More precisely, for the videos encoded with QP$<$ 25 we use the CNN that learned from uncompressed samples; for QP $\in \{25, 30, 35\}$, the one trained with samples compressed at QP 25; for QP $\in \{40, 41, \ldots, 46\}$, the one trained with samples compressed at QP 40; and finally, for the largest QP $\in \{47, 48, \ldots, 51\}$, the network trained on the samples compressed at QP 51.

In all the cases, we use the same training procedure described in Sec. 4.2, whereas the training set increases to 10000 samples per character class.

The results of our experiments are reported in terms of character recognition accuracy as a function of the bit-rate. They are shown in Fig. 11(a), 11(b), 11(c), 11(f) for class-A color, class-A gray-0, class-A gray-1 and Class-A synthetic; and in Fig. 11(d) and 11(e) for class-B 0 and 1 sequences respectively. Just for reference, we also report the accuracy of our scheme, which does not depend on the coding rate once a minimum rate constraint is satisfied.

Using the CNN accuracy rate as an estimator for the human readability of compressed characters, we observe that the minimum coding rate needed to achieve the same performance of the proposed scheme is quite larger: for HEVC we need 10, 6, 15, 9 and 8 times as much rate respectively for class-A color, class-A gray-0, class-A gray-1, class-A synthetic, and class-B 0 and 1 sequences. SCC performs slightly better but

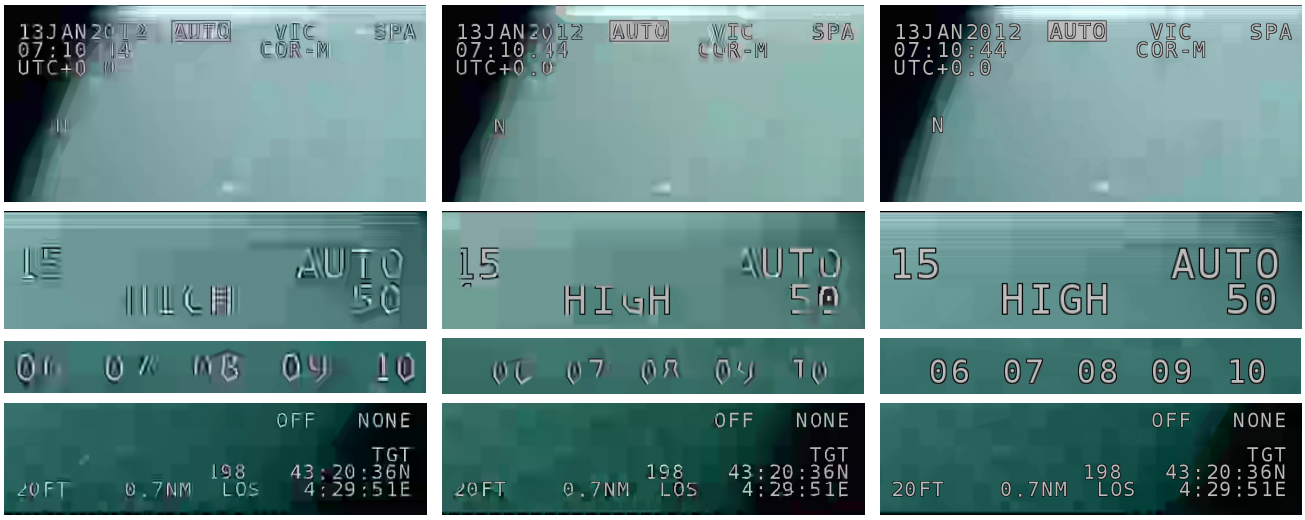


Fig. 8. Reconstruction artifacts for the class-A color decoded video at QP=50. Left: HEVC; Center: SCC; Right: our proposed scheme.

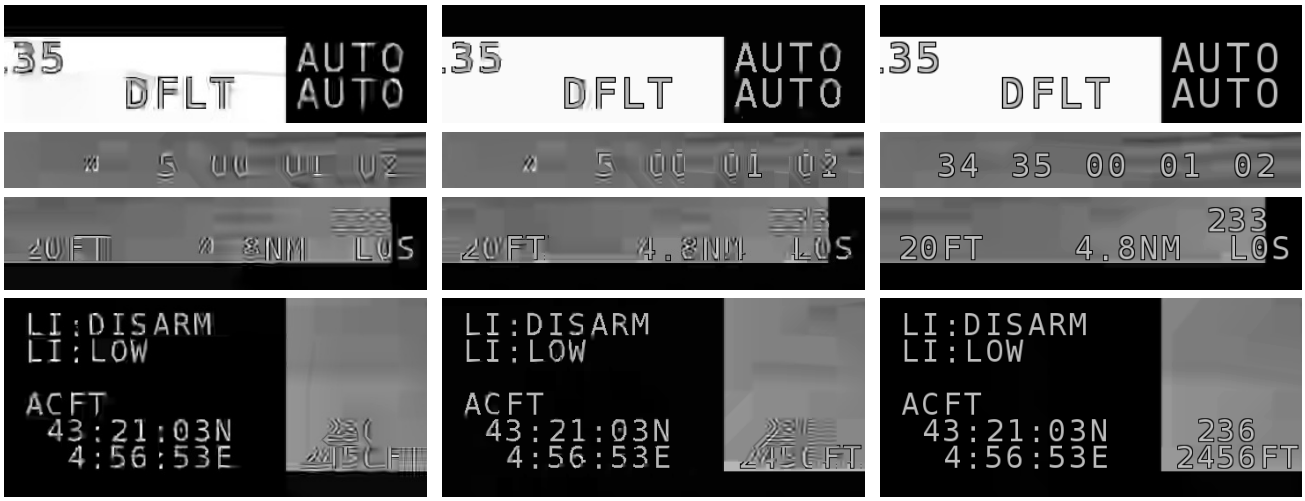


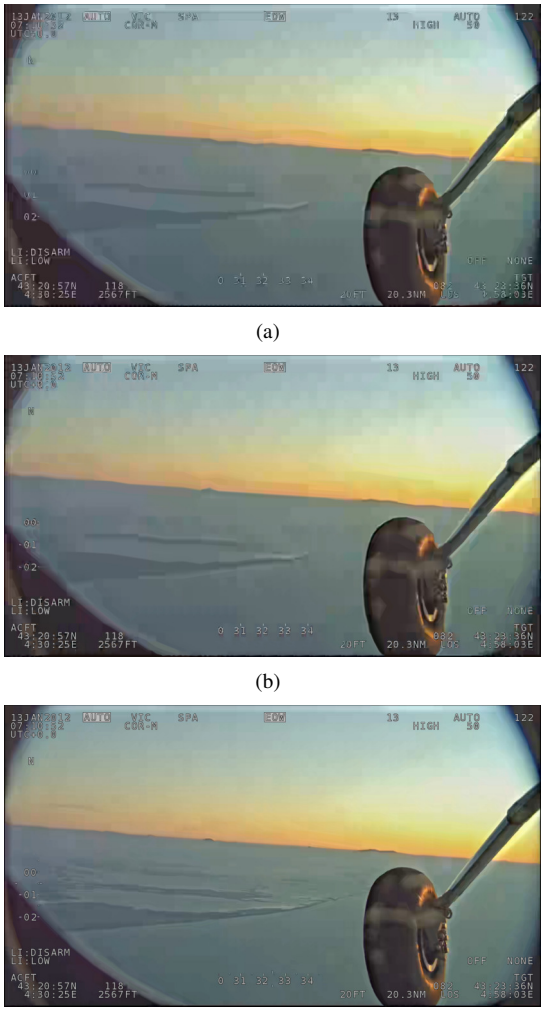
Fig. 9. Reconstruction artifacts for the class-A grayscale-0 decoded video at QP=50. Left: HEVC; Center: SCC; Right: our proposed scheme.

still requires respectively 5, 4, 7, 8 and 4 times more rate than the proposed scheme.

From a different perspective, the accuracy of the two references drops at the low bitrates which are our applicative scenario of interest. For example, at QP=50 the HEVC scheme accuracy is around 92.1% for the class-A color sequence, around 97.1% for the class-A grayscale-0 sequence, around 97.6% for the class-A grayscale-1 sequence, around 83.6% for the class-A synthetic sequence, and, respectively, around 91.3% for the class-B 0 sequence, around 6.3% for the class-B 1 sequence. The SCC scheme attains slightly better results only: 96.5% for the class-A color sequence, 97.2% for the class-A grayscale-0 sequence, 97.6% for the class-A grayscale-1 sequence, 91.3% for the class-A synthetic sequence, 93.5% for the class-B 0 sequence and 6% for the class-B 1 sequence. As shown in the left and central column of Fig. 8 and 9, the decoded characters are affected by heavy reconstruction artifacts that severely compromise the character readability.

5.3 Power Consumption

Finally, we estimate the the power consumption of our proposed video coding system. To allow for passive heating, the power consumption of our complete system shall fall in the 20 to 35 W range. To meet such constraint, computationally complex operations will be offloaded to a specialized hardware such as FPGAs and ad-hoc ICs. Concerning character recognition, in [**?**] an optimized LeNet5 implementation with a 3W power consumption envelope is proposed, leveraging Xilinx deep learning library [**?**]. Such implementation classifies a single character in about 0.151 ms, i.e. requires around 30 ms to classify the 200 characters found in our test sequences. Concerning residual video coding, the H.265/HEVC encoder library in [**?**] enables 4K video encoding at 60 fps video coding for a power consumption of 3W using an ad-hoc IC. Concerning character and graphics localization and inpaint-



(a) (b) (c)

Fig. 10. Reconstruction artifacts in background video, class-A color sequence encoded at 0.0018 bpp. Top: HEVC; Middle: SCC; Bottom: our proposed scheme.

ing, such operations could also be efficiently offloaded to a FPGA relying on standard computer vision libraries for a few additional Watts [**?**]. Concerning characters and graphical primitives coding, the algorithm complexity is very moderate and is better implemented on the system CPU. Concluding, offloading computationally intensive character recognition and residual video coding to specialized hardware would account for a fraction of the available power budget, leaving enough margin for an ARM-based CPU, memory, storage and other system components.

6 Conclusions And Further Work

In this work we proposed a scheme for compressing airplane cockpit video at low bitrates without affecting the readability of computer generated graphics (text, lines). At the encoder, we first encode the graphics exploiting their semantics, then the residual video obtained by inpainting the graphics is compressed with a standard codec. At the decoder, the graphics are reconstructed and overlaid on the decompressed residual video, recovering the original video. Our experiments with real cockpit video sequences show two key advantages with respect to other screen content coding techniques. First, removal of high-frequency components due to the computer graphics slashes the encoding bitrate by 10% to 80% with respect to standard HEVC/H.265 and by 5% to 70% with respect to its SCC extension. Second, characters remain perfectly readable at the receiver even at very low bitrates, competing schemes demanding far higher bitrates to achieve comparable error rates. While in this work we leveraged intra-frame redundancies to achieve efficient graphics compression, we leave for our future work to exploiting temporal redundancies. Moreover we only considered simple graphical primitives such as horizontal and vertical lines. In future works we plan to use CNN-based classifier to recognize more complex graphical object that can appear in airplane cockpit screens, in order to encode them losslessly with suitable methods.

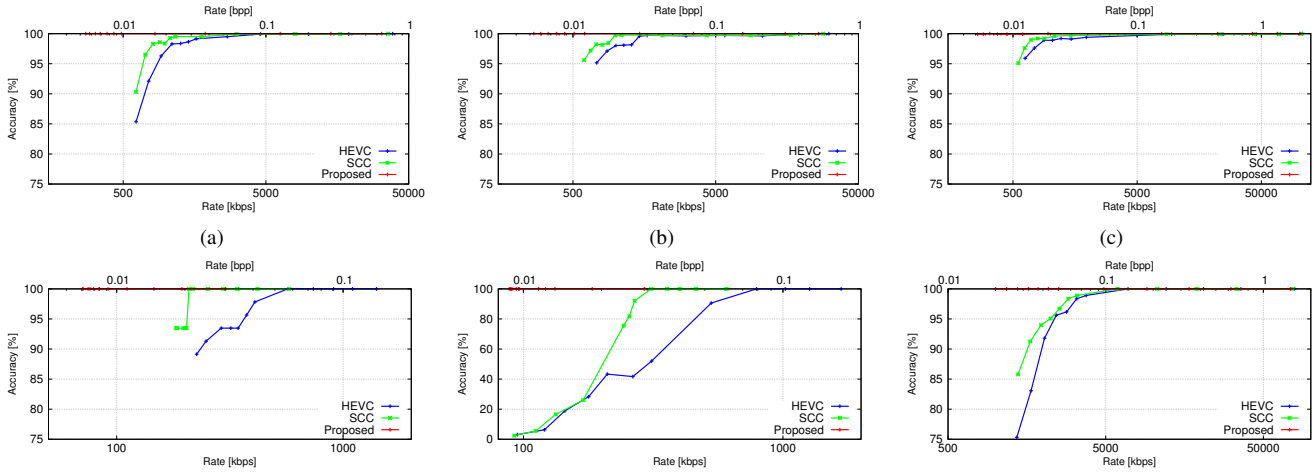


Fig. 11. Accuracy in character recognition as a function of video bitrate. First row - Left: class-A color (Fig. 1(a)); Center: class-A gray-0 (Fig. 1(b)); Right: class-A gray-1 (Fig. 1(c)). Second row - Left: class-B 0 (Fig. 1(d)); Center: class-B 1 (Fig. 1(e)); Right: class-A synthetic (Fig. 1(f)).