Video compression of airplane cockpit screens content

Iulia Mitrica

▶ **To cite this version:**

Iulia Mitrica. Video compression of airplane cockpit screens content. Signal and Image processing. Institut Polytechnique de Paris, 2021. English. NNT : 2021IPPAT042 . tel-03712265

HAL Id: tel-03712265
https://theses.hal.science/tel-03712265

Submitted on 3 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT042

Thèse de doctorat

Video Compression of Airplane Cockpit Screens Content

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris et Safran Datasystems

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 17 Décembre 2021, par

IULIA MITRICA

Composition du Jury :

Marc Antonini Directeur de Recherche, CNRS and University of Nice-Sophia Antipolis	Président
Lu ZHANG Maître de Conférence, INSA Rennes	Rapporteur
Adrian Munteanu Professeur, Vrije Universiteit Brussel (VUB)	Rapporteur
Federica Battisti Chercheuse, University of Padova	Examinateur
Marco Cagnazzo Professeur, Telecom Paris	Directeur de thèse
Attilio Fiandrotti MdC chez Télécom et Chercheur chez l'Université de Turin	Co-directeur de thèse
Christophe Ruellan Ingénieur, Safran Datasystems	Co-directeur de thèse
Eric Mercier Ingénieur, AnotherBrain	Invité
Béatrice Pesquet-Popescu Directrice de l'Innovation, Thales	Invité

Abstract

This thesis addresses the problem of encoding the video of airplane cockpits. The cockpit of modern airliners consists in one or more screens displaying the status of 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.,) often superimposed over natural images (e.g., navigation maps, outdoor cameras, etc.). Plane sensors are usually inaccessible due to security reasons, so recording the cockpit is often the only way to log vital plane data in the event of, e.g., an accident. Constraints on the recording storage available on-board require the cockpit video to be coded at low to very low bitrates, whereas safety reasons require the textual information to remain intelligible after decoding. In addition, constraints on the power envelope of avionic devices limit the cockpit recording subsystem complexity.

Over the years, a number of schemes for coding images or videos with mixed computer-generated and natural contents have been proposed. Text and other computer generated graphics yield high-frequency components in the transformed domain. Therefore, the loss due to compression may hinder the readability of the video and thus its usefulness. For example, the recently standardized Screen Content Coding (SCC) extension [15] of the H.265/HEVC [16] standard includes tools designed explicitly for screen contents compression. Our experiments show however that artifacts persist at the low bitrates targeted by our application, prompting for schemes where the video is not encoded in the pixel domain.

This thesis proposes methods for low-complexity screen coding where text and graphical primitives are encoded in terms of their semantics rather than as blocks of pixels. At the encoder side, characters are detected and read using a convolutional neural network. Detected characters are then removed from screen via pixel inpainting, yielding a smoother residual video with fewer high frequencies. The residual video is encoded with a standard video codec and is transmitted to the receiver side together with text and graphics semantics as side information. At the decoder side, text and graphics are synthesized using the decoded semantics and superimposed over the residual video, eventually recovering the original frame. Our experiments show that an AVC/H.264 encoder retrofitted with our method has better rate-distortion performance than H.265/HEVC and approaches that of its SCC extension.

If the complexity constraints allow inter-frame prediction, we also exploit the fact that co-located characters in neighbor frames are strongly correlated. Namely, the misclassified symbols are recovered using a proposed method based on low-complexity model of transitional probabilities for characters and graphics. Concerning character recognition, the error rate drops up to 18 times in the easiest cases and at least 1.5 times in the most difficult sequences despite complex occlusions. By exploiting temporal redundancy, our scheme further improves in rate-distortion terms and enables quasi-errorless character decoding. Experiments with real cockpit video footage show large rate-distortion gains for

the proposed method with respect to video compression standards.

Abstrait

Cette thèse aborde le problème de l'encodage de la vidéo des cockpits d'avion. Le cockpit des avions de ligne modernes consiste en un ou plusieurs écrans affichant l'état des instruments de l'avion (par exemple, la position de l'avion telle que rapportée par le GPS, le niveau de carburant tel que lu par les capteurs dans les réservoirs, etc.,) souvent superposés au naturel images (par exemple, cartes de navigation, caméras extérieures, etc.). Les capteurs d'avion sont généralement inaccessibles pour des raisons de sécurité, de sorte que l'enregistrement du cockpit est souvent le seul moyen de consigner les données vitales de l'avion en cas, par exemple, d'un accident. Les contraintes sur la mémoire d'enregistrement disponible à bord nécessitent que la vidéo du cockpit soit codée à des débits faibles à très faibles, alors que pour des raisons de sécurité, les informations textuelles doivent rester intelligibles après le décodage. De plus, les contraintes sur l'enveloppe de puissance des dispositifs avioniques limitent la complexité du sous-système d'enregistrement du poste de pilotage.

Au fil des ans, un certain nombre de schémas de codage d'images ou de vidéos avec des contenus mixtes générés par ordinateur et naturels ont été proposés. Le texte et d'autres graphiques générés par ordinateur produisent des composants haute fréquence dans le domaine transformé. Par conséquent, la perte due à la compression peut nuire à la lisibilité de la vidéo et donc à son utilité. Par exemple, l'extension récemment normalisée SCC (Screen Content Coding) [15] de la norme H.265/HEVC [16] comprend des outils conçus explicitement pour la compression du contenu de l'écran. Nos expériences montrent cependant que les artefacts persistent aux bas débits ciblés par notre application, incitant à des schémas où la vidéo n'est pas encodée dans le domaine des pixels.

Cette thèse propose des méthodes de codage d'écran de faible complexité où le texte et les primitives graphiques sont codés en fonction de leur sémantique plutôt que sous forme de blocs de pixels. Du côté du codeur, les caractères sont détectés et lus à l'aide d'un réseau neuronal convolutif. Les caractères détectés sont ensuite supprimés de l'écran via le pixel inpainting, ce qui donne une vidéo résiduelle plus fluide avec moins de hautes fréquences. La vidéo résiduelle est codée avec un codec vidéo standard et est transmise du côté récepteur avec une sémantique textuelle et graphique en tant qu'informations secondaires. Du côté du décodeur, le texte et les graphiques sont synthétisés à l'aide de la sémantique décodée et superposés à la vidéo résiduelle, récupérant finalement l'image d'origine. Nos expériences montrent qu'un encodeur AVC/H.264 équipé de notre méthode a de meilleures performances de distorsion-débit que H.265/HEVC et se rapproche de celle de son extension SCC.

Si les contraintes de complexité permettent la prédiction inter-trame, nous exploitons également le fait que les caractères co-localisés dans les trames voisines sont fortement corrélés. À savoir, les symboles mal classés sont récupérés à l'aide d'une méthode proposée basée sur un modèle de faible complexité des probabilités de transition pour les caractères

et les graphiques. Concernant la reconnaissance de caractères, le taux d'erreur chute jusqu'à 18 fois dans les cas les plus faciles et au moins 1,5 fois dans les séquences les plus difficiles malgré des occlusions complexes. En exploitant la redondance temporelle, notre schéma s'améliore encore en termes de distorsion de débit et permet un décodage de caractères quasi sans erreur. Des expériences avec de vraies séquences vidéo de cockpit montrent des gains de distorsion de débit importants pour la méthode proposée par rapport aux normes de compression vidéo.

Contents

Acronyms

3GPP	The 3rd Generation Partnership Project
ACT	Adaptive Color Transform
ALF	Adaptive Loop Filtering
AMVR	Adaptive Motion Vector Resolution
ATSC	Advanced Television Systems Committee
AVC	Advanced Video Coding
BWT	Burrrows-Wheeler Transform
CCP	Cross-Component Prediction
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CRT	Cathode-Ray Tube
CTU	Coding Tree Unit
CU	Coding Unit
DVB	Digital Video Broadcasting
EFB	Electronic Flight Bag
FPGA	Field-Programmable Gate Array
GPU	Graphics Processing Unit
GSM	Global System for Mobile Communications
H.264	Advanced Video Coding
H.265	High Efficiency Video Coding
HD	High Definition
HEVC	High Efficiency Video Coding
HUD	Head-Up Displays
YOLO	You Only Look Once
YCoCgR	Luma, Chrominance orange, Chrominance green, Red Conversion
JVT	Joint Video Team
LCD	Liquid-Crystal Display
LZW	Lempel-Ziv-Welch
MAC	Multiply-Accumulation Operations
MPEG	Moving Picture Expert Group
MTBF	Mean Time Between Failure
OCR	Optical Character Recognition
PLT	Palette Mode
PU	Prediction Unit
QP	Quantisation Parameter
ReLU	Rectified Linear Unit
RGB	Red, Green, Blue

RDPCM	Residual differential pulse code modulation
SM	String Mode
SMS	Short Message Service
SCC	Screen Content Coding
TS	Transform Skip
TU	Transform Unit
VESA	Video Electronics Standards Association

Chapter 1

Introduction

In this chapter, we will introduce the motivation of the thesis, we will describe the background and we will highlight the contributions brought by this thesis.

1.1 Motivation

The cockpit of modern airplanes consists of 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 is composed by computer-generated graphics (e.g., text, lines, etc.) often superimposed over natural images (e.g., maps, outdoor pointing cameras, etc.).

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. Furthermore, the complexity constraints of avionic applications make the design of any scheme for airplane cockpit video compression particularly challenging.

We propose a low complex semantic screen coding scheme for the avionics power constraint environment 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 being compressed with a standard codec and transmitted to the receiver together with the text semantics. At the decoder side, text is synthesized using the decoded semantics and superimposed over the decoded 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.

1.2 Backgrond

1.2.1 Airplane Cockpit Screens

Airplane cockpit screens [108] are complex displays systems that integrates the output and/or input of several types of instruments depending on the type of plane (e.g. passenger, cargo or military) displaying specific functionalities that will help the pilots in their activities. Several types of generic instruments can be found in a cockpit: flight instruments (altimeters, airspeed indicator, magnetic direction indicator), navigation instruments (GPS location system), communication systems (facilitates the communication with the air traffic control), systems management (supervisors of parameters of aircraft's engine, as temperature, pressure, fuel and oil) and mission management. The content of those kind of screens groups a high amount of functions and information. Several categories can be identified. Analogue cockpit is characterised by mechanical and analogue instruments. Glass cockpit represent a solid-state display screens in cockpit instrumentation. Interactive cockpit are interactive displays and computers. Those cockpit screens displays the flight information with a certain graphic's policy specific to each type of screen, to facilitates the readability of the illustrated symbols. In figure 1.1 we may see a real example of a cockpit of A380 airplane.

Other instrumentation technologies are integrated such as Head-Up Displays (HUD), Video Recording Systems, Helmet-Mounted Displays or Electronic Flight Bag (EFB). Head-Up Displays are transparent displays system that illustrate the data and helps pilots not to refocus to view outside after regarding the optical displays system and are similar in spirit with Helmet-Mounted Displays. The Electronic Flight Bag is the replacement of the traditional pilot's flight bag of documents that pilots carry to the cockpit in a digital format.

In a cockpit, the technology of the display is an important parameter that influence the content of the computer screens videos. It can be cathode-ray tube (CRT) or more recent liquid-crystal display (LCD). LCDs improve readability in direct sunlight and can display sophisticated graphics and real time recordings. They weigh 40% less, occupy 40% less space and consume 40% less power than CRTs [110].

Nowadays, airplanes cockpits dispose of onboard Video and Data Recorders. Those are systems that work in harsh environments and record data for mission analysis, for example. Most of the time, they do not have a cooling system because of the limited space, relaying on the design dimensions of the system box to dissipate the heat, the power consumption being limited between 20 to 40 W. They use video standards to encode the recorded content on 64 GB to 2TB storage capacity. Furtermore,those systems are not designed for screen content coding, they lacking efficiency when encoding hybrid image and textual content and suffer from coding artifacts that affect the semantic understanding of textual information at low rates. The technology on which they relay is FPGA, in order to minimise the obsolescence problem of avionics when a product is design for several decades [111].

The aim of this thesis is to propose a low complex semantic architecture for power constraint environment to code the content displayed by the screens of the computers presented in the cockpit of an airplane by exploring spatial and temporal redundancies of such videos. At the decoder side the video must be reconstructed without loss of key plane information even at very low coding rates. In the section 1.3 we acquire our contributions towards achieving this aim. Beside we will provide an outline of the thesis along with the

Figure 1.1: Cokpit screens of Airbus A380 airplane [109].

list of published work.

1.3 Contributions

In Chapter 1 we provide an introduction in the field of screen content coding. We started by motivating the thesis development and we described the airplane cockpit screens videos. Further, we introduce our contributions to the field of screen content coding. Finally, we provide a comprehensive study of the reference techniques with a predilection for mixed content image and video compression techniques on one hand, on the other on character recognition via convolutional neural networks highlighting our attempt of complexity.

In Chapter 3 we describe our proposed scheme for compressing airplane cockpit video at low bitrates without affecting the readability of computer generated graphics (text, lines). At the encoder, we first compress 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 H.265/HEVC and by 5% to 70% with respect to its SCC extension. Second, characters remain perfectly readable at the receiver even at very low bitrates, where competing schemes demanding far higher bitrates to achieve comparable error rates. The Chapter 3 is evaluated in the following published work:

- Mitrica, I., Mercier, E., Ruellan, C., Fiandrotti, A., Cagnazzo, M. and Pesquet-Popescu, B., 2019. Very low bitrate semantic compression of airplane cockpit screen content. IEEE Transactions on Multimedia, 21(9), pp.21572170.

In Chapter 4 we describe our enhancements to our proposed method to encode lossless cockpit screens videos. We experimentally evaluated our semantic scheme for airplane cockpit video compression with two different video codecs. Availing different coding configurations, several interesting results emerge. When the video is encoded in Intra-only mode, the AVC/H.264 codec retrofitted with our proposed scheme outperforms the more recent H.265/HEVC codec and performs close to its SCC extension. Thus, we can exploit the modular structure of the semantic coding scheme to improve the coding efficiency keeping the complexity suitable for avionic applications. Contrary, when temporal inter-frame prediction is used, the competitive advantage of our semantic video compression scheme decreases as the rate of the encoded characters is not negligible any more with respect to the rate of the residual video. The Chapter 4 is evaluated in the following published work:

- Mitrica, I., Fiandrotti, A., Cagnazzo, M., Mercier, E. and Ruellan, C., 2019, October. Cockpit Video Coding with Temporal Prediction. In 2019 8th European Workshop on Visual Information Processing (EUVIP) (pp. 2833). IEEE. **Best Paper Award**

In Chapter 5 we proposed to explore the temporal axis to recover the occluded characters through a low complex method and to compress the airplane cockpit video at low bitrates without affecting the readability of computer generated graphics (text, lines). Firstly, our experiments that allow the codecs and the coding of the characters tables to explore the redundancy between adjacent frames display gains that range from 15% for low quality to 70% for high quality regarding the reference coding schemes. Secondly, the accuracy of the neural network trained without occlusions and without outputs adjusted increases with 0.9% by training it with occlusions, result which shows that it is important to consider the occlusions during the training of a neural network for character detection. Furthermore, our experiments shows that is almost impossible to become quasi-errorless just by training the detector with a given set of occlusions. Thus, we propose a method to recover the errors by taking into account the temporal aspect, keeping the complexity of the proposed method low. By adjusting the outputs of the character classifier, we always increased classification accuracy significantly, becoming quasi-errorless by introducing some additional rules and keeping the complexity low. The Chapter 5 will be evaluated in the IEEE Transactions on Multimedia publication.

In Chapter 6 we draw the conclusions of this thesis and we discuss the possible future research perspectives.

Chapter 2

Background and State of the Art

In this chapter, we first provide the key concepts related to image and video compression (Sec. 2.1) and character recognition (Sec. 2.2) that are instrumental to the understanding of the rest of this work. Then, we review the state of the art in screen content coding (Sec. 2.3) and we highlight the limitations that prompted the development of this work. We will discuss the reference technologies regarding mixed images and video coding solutions, character recognition via CNN and some aspects regarding computational complexity.

2.1 Background on Image and Video Coding

2.1.1 Image Compression

Video compression builds and extends upon techniques for still image compression, so before delving into video compression, we will review the basics of image compression. One of the most popular standard for *image compression* is *JPEG* [137]. It allows lossy or lossless still images compression. The algorithm for *lossy coding* is a DCT-based encoding scheme, illustrated in figure 2.1. The input image is first partitioned into blocks of 8×8 pixels. Two-dimensional DCT (Discrete Cosine Transform) is applied to each block. The transform coefficients are quantized by a uniform quantizer. After quantization, the encoder computes the difference between the quantized coefficient of the current block and that of the previous block of the same component, encoding it lossless with Huffman tables [53, 56]. Thus, spatial redundancy reduction techniques are leveraged to improve the coding efficiency, image compression being based on exploiting *spatial redundancy* (also called *intra*-frame redundancy, i.e. the statistical dependence between nearby pixels within an image).

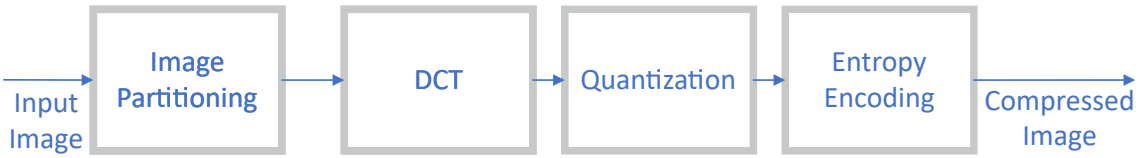


Figure 2.1: Functional principle of an image encoder (e.g. JPEG).

2.1.2 Video Compression

A group of successive images constitute a video. The compression techniques for video coding explore both spatial and temporal redundancy. *Temporal redundancy* represent the statistical dependence between pixels from successive frames in video sequence. It is also called *Inter*-frame redundancy [53, 56]. The redundancy between temporally adjacent frames is reduced by computing *temporal prediction* meaning that the corresponding pixels among frames have to be found by computing a error function for a matching criterion. This solution is mirrored by *motion compensation* as the process of compensating for the displacement of moving objects from one frame to another [53, 56]. The prediction error is defined as:

$$e(x, y, t) = I(x, y, t) - I(x - u, y - v, s) \tag{2.1}$$

where $I(x, y, t)$ are pixel values at spatial location (x, y) in frame t and $I(x - u, y - v, s)$ are corresponding pixel values at spatial location $(x - u, y - v)$ in a reference frame s. $I(x - u, y - v, s)$ is referred to as the motion-compensated prediction of $I(x, y, t)$, and $e(x, y, t)$ is the prediction residual for $I(x, y, t)$. The coordinates (u, v) represent the output of the motion estimator and define the relative motion of a block from one frame to another being referred to as the *motion vector* for a given block at (x, y). It is determined by considering a search window in a reference frame (e.g. of $\times[-p, p]$) at the location (x, y) and a matching criterion (e.g. sum of absolute differences (SAD) or sum of squared differences (SSD)), as illustrated in figure 2.2 [53, 56].

The prediction error is further transformed and quantized. The functions used as transform convert the images information from the spatial domain into the frequency domain. The quantization step consists of dividing the transform coefficients by a dynamic value, used to manage the amount of coding rate, and then discarding trivial coefficients by reducing them to zero. Also, in modern video compression the spatial and temporal prediction is also performed. The prediction parameters required at the decoder side to perform the same prediction are also encoded. This video coding solution is called hybrid video coding because it uses both prediction and transform and it is illustrated in Fig. 2.3. The input pictures are partitioned into blocks which can be predicted using either Intra or Inter modes. In Intra mode, pixels values in a block are predicted using spatially neighboring pixels. In Inter mode, a block is predicted by a reference block in a temporal reference picture. The motion compensated prediction is reffered to as Inter prediction.

In any lossy compression scheme, there is a trade-off between the distortion D and the rate R at the encoder output. We refer to it as the *rate-distortion function* $D(R)$. Let s be a set of samples and s_k the k-th sample of the set, such as a video frame or a block of a video frame, and let $p \in P_k$ be a vector of coding decisions or syntax element values out of a set P_k of coding options for a set of source samples s_k. The problem of finding the coding decisions p that minimize a distortion measure $D_k(p) = D(s_k, s'_k)$ between the original samples s_k and their reconstruction $s'_k = s'_k(p)$ subject to a rate constraint R_c can be formulated as:

$$\min_{p \in P_k} D_k(p) \quad \text{subject to} \quad R_k(p) \leq R_c \tag{2.2}$$

where $R_k(p)$ represents the number of bits that are required for signaling the coding decisions p in the bitstream and the rate for the actual coding of the samples. This

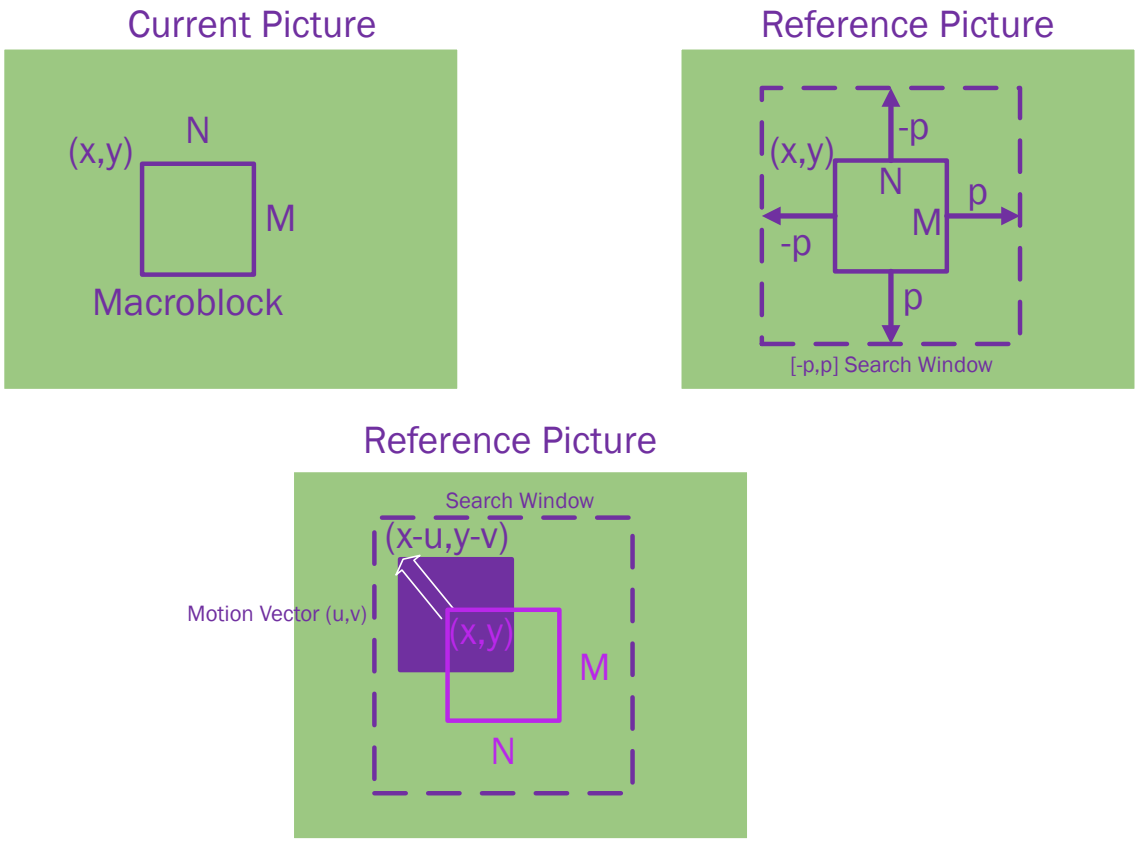


Figure 2.2: Motion estimation solution [53, 56].

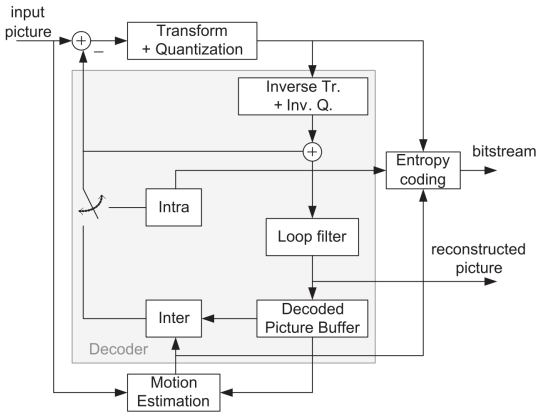


Figure 2.3: Hybrid video coding block diagram [139]

constrained minimization problem can be reformulated as an unconstrained minimization,

$$\min_{p \in P_k} D_k(p) + \lambda R_k(p) \tag{2.3}$$

where $\lambda \geq 0$ denotes so-called Lagrange multiplier [54].

Rate-distortion optimisation is very important, being implemented in the reference software of many coding solutions. It is non normative and often codecs employ "early termination" strategies, because a full rate-distortion optimisation is computationally complex.

2.1.3 The Need for Standards

The need for interoperability among pieces of video equipment manufactured by different producers, quickly set the need for standard-compliant video coding techniques. In Fig. 2.4 we have an ensemble of communication systems. The vertical line X represents the interface between system A and system B. An implementer can expose interface X as a standard [138] meaning that a standard is a *documented agreement reached by a group of individuals who recognise the advantage of all doing certain things in an agreed way.*

As for the standardisation bodies, the Moving Picture Experts Group (ISO/IEC MPEG) is an organization that develops standards for digital audio and video encoding within the International Organization for Standardization (ISO). The standardisation process includes several phases. In the first step the requirements for a specific application are identified. The next step includes the development of the algorithms by various laboratories, companies or contributors. The last phase generates a draft which will be further validated using simulations and tests using specialised metrics and will be published after successful validation.

2.1.4 Evaluation Criteria

2.1.4.1 Visual Quality Assessment

In the multimedia applications, it is necessary to evaluate the quality of the compressed image or video with respect to the initial version of the data. There are several types of visual quality assessment: subjective methods using groups of observers to establish the images quality; objective methods that explore the statistical properties of the signals. Without having the aim to be exhaustive, it is necessary to recall several of those objective methods.

Mean Square Error (MSE) is a difference between the original signal x and the reconstructed $x^{'}$, as follows:

$$MSE = \frac{1}{N} \times \sum_{i \in N} (x_i - x_i^{'})^2 \tag{2.4}$$

where N is the number of samples.

In this thesis we will use objective methods to evaluate the quality of the decoded videos, such as **Peak-Signal-to-Noise-Ratio (PSNR)**. This metric represents a gain expressed in dB with respect to the reconstructed signal. It indicates the degree of similarity between the original signal and the reconstructed signal. It is generally calculated on the entire image and do not account for the local artefacts.

$$PSNR = 10 \times log_{10}(\frac{255^2}{MSE}) \tag{2.5}$$

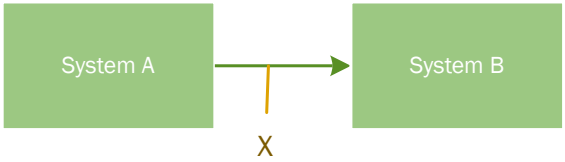


Figure 2.4: If an interface is exposed (i.e. X), it can be implemented as standard [138].

Another metric is **Structural Similarity Index (SSIM)**. The structural similarity index (SSIM) is a perception-based model that considers image degradation as a perceived change in structural information. SSIM has been initially designed to measure the quality of still images and then adapted to video quality assessment.

2.1.4.2 Rate-Distortion Performance

To assess the trade-off between the gain in bit rate and the quality of the reconstructed signal, a comparison between different coding approaches is done using a representation of their rate-distortion curves and the metric called Bjontegaard metric. There are two Bjontegaard metrics: 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 [44].

2.1.4.3 Computational Complexity

The computational complexity is a performance parameter that will be evaluated. The complexity is computed for a software algorithm and compared with another scheme. We use different methods to calculate the complexity such as algorithm complexity, the processing time or operations and memory occupancy.

2.1.5 The Advanced Video Coding (AVC/H.264) Standard

MPEG was established in 1988 and produced a number of digital standards. One of the well known video compression standard is AVC/H.264 for high-definition digital video developed together with ITU-T VCEG and Joint Video Team (JVT). Also known as MPEG-4 Part 10 or Advanced Video Coding (MPEG-4 AVC), AVC/H.264 is defined as a block-based compression standard because the input video pictures are split in macroblocks (MB), grouped in slices. There are I-slices, P-slices and B-slices. Spatial prediction is possible for I-slices meaning that a block is predicted from the pixels of neighboring blocks in the same slice in a causal area. Inter mode exploits the temporal correlation between the frames, being supported by P and B-slices. While for P-slices, only prediction from past frames is possible, for B-slices also prediction from future slices can be considered. The motion estimation is performed at a quarter-pixel precision, with variable-size, square or rectangular blocks. Two alternative entropy coding methods are implemented: Context-Adaptive Variable Length Coding (CAVLC) and Context-Adaptive Binary Arithmetic Coding (CABAC). AVC/H.264 is suitable for a wide range of applications from low-bitrate, low-delay mobile transmission to high definition TV. It delivers the same visual quality at 2 times less coding rate and being 4 times more complex than its predecessor MPEG-2.

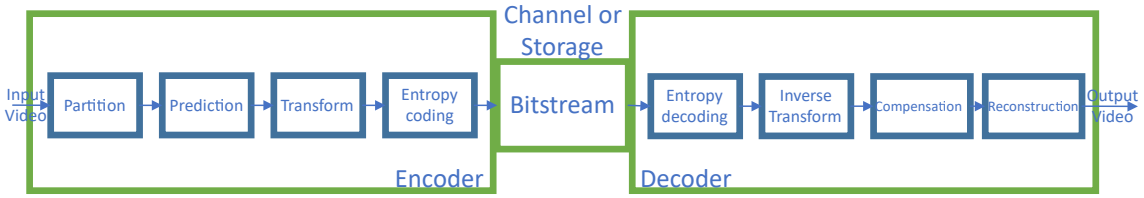


Figure 2.5: Functional principle of a video codec (e.g. H.265/HEVC).

2.1.6 The High Efficiency Video Coding (H.265/HEVC) Standard

High Efficiency Video Coding Standard was released in 2013. Compared with AVC/H.264, it delivers the same visual quality at a rate that is 50% lower. DVB (Digital Video Broadcasting) and ATSC (Advanced Television Systems Committee) standards organizations recommended H.265/HEVC for 4K broadcast services, while 3GPP propose H.265/HEVC for HD and 4K mobile streaming. The principal differences between H.265/HEVC and AVC/H.264 come from coding block (CB) size extended from 16 × 16 to sizes up to 64 × 64, improved variable-block-size partition, improved Intra prediction, improved motion vector prediction and motion region merging and from improved motion-compensation filtering, H.265/HEVC requiring more signal processing power than AVC/H.264 [55].

The functional principle of H.265/HEVC is similar with AVC/H.264. It is illustrated in figure 2.5. The input video is decomposed through partitioning, frame by frame, in coding units (CUs), following a coding tree partitioning structure (CTU), of maximum 64x64 pixels. Each CTU is splitted hierarchically using mode selection algorithms into CU, which can be from 8x8 to 64x64 pixels. During the encoding process, each CU can be coded either using Intra or Inter prediction. Unlikely AVC/H.264, H.265/HEVC has two basic units: prediction unit (PUs) and transform unit (TUs). Each PU is used to perform the prediction operations, all the pixels in one PU being predicted using the same rule. H.265/HEVC supports eight types of PU and one CU can be splitted into several PUs. Further, each transform unit (TU) is used to process the predictive residue information. Within an intra coded CU, a TU is always part of a PU, while for an inter-frame coded CU, a TU can cross different PUs [55].

After decomposition of input video, the CUs are coded with specified block partitioning shape as side information, being sent to the decoder side. As in all coding standards, the first frame is encoded as an Intra frame. The intra frame prediction uses directional prediction within the same frame. The inter-frame prediction performs motion-estimation in the encoder and motion compensation in the decoder. The motion vector information and coding mode information are contained in the bitstream as side information and conveyed at the decoder. The predictive residuals from either intra-prediction or inter-frame prediction will then be converted to the transform domain. The transformed coefficients are then scaled, quantized and filtered. Finally, the scaled and quantized coefficients are sent to the entropy coder to further reduce the size of the bitstream. In the decoder, exactly the reverses processing as in the encoder will be performed to reconstruct the video data [55].

2.1.7 Versatile Video Coding (VVC/H.266) Standard

Versatile Video Coding solution or VVC/H.266 represent the new evolution of the standardisation of JVT committee after H.265/HEVC, finalised in July 2020, that enables

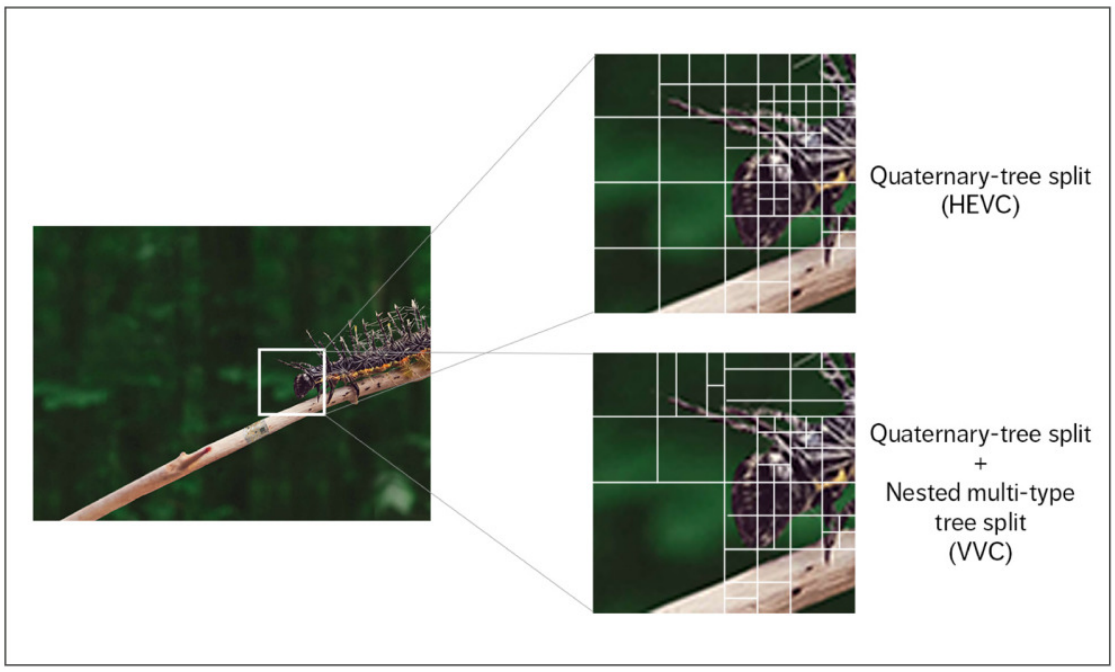


Figure 2.6: Partitioning comparisons between H.265/HEVC and VVC/H.266 [49].

gains of around 40% with respect to H.265/HEVC. It is composed by all the extensions of H.265/HEVC, including Screen Content Coding solution, defining its adaptability characteristic to several applications as one standard [49, 50, 51]. While its complexity makes it unfit for the embedded scenario we target in this work, we nevertheless quickly overview it as it represents state-of-the-art in standardized video coding techniques. VVC/H.266 builds upon techniques proposed in earlier MPEG standards adding several aspects as novelties. The block partitioning is done as a hierarchical tree partitioning, with a maximum value of 128x128 pixels, adding savings in coding rate at higher resolutions. The partitioning scheme is more flexible that for H.265/HEVC. It eliminates the PU and TU concepts of H.265/HEVC and introduce the coding quadtree with multitype tree. It signifies that as a CTU is first partitioned by a quadtree structure, then, the quadtree tree leaf nodes can be further partitioned by a multitype tree structure. Those can be combined using the concepts of slices and tiles. The novelty here is that they can be independently encodable/decodable. The block partitioning provides about 8% bitrate reduction with respect to H.265/HEVC [49, 50, 51]. For example, figure 2.6 illustrates a section of a partitioned picture. H.265/HEVC block partitioning (at top right) uses quaternary-tree split with coding blocks up to 64x64 pixels, while VVC/H.266 block partitioning (at bottom right) uses quaternary-tree split with coding blocks up to 128x128 pixels and the multitype tree [49, 50, 51].

Regarding the prediction scheme, it has also been improved. The Intra prediction directions were doubled with respect to H.265/HEVC. The techniques that improve Inter prediction significantly reduce the bitrate with respect to H.265/HEVC. Those includes the affine motion prediction, adaptive motion vector resolution, bi-directional optical flow, decoder side-motion vector refinement and geometric partitioning mode. For example, Inter prediction identifies cases in which the motion information does not have to be transmit-

ted as motion vectors, but can be inferred at the decoder side from similar moving parts of the frame. The bi-directional optical flow tool can infer motion vectors by determining the optical flow in reference frames and the decoded side-motion vector refinement can infer motion vectors by minimizing differences between reference pictures. Moreover, the precision of the motion vectors has also been increased to 1/16 pixel compared to the quarter pixel resolution of H.265/HEVC [49, 50, 51].

Another novel tool is Adaptive Loop Filtering (ALF) that scans the picture after it has been decoded and applies at CTU level one of several two-dimensional finite impulse response filters before the picture becomes output. It reduces the artifacts and contributes to a consistent coding rate saving over H.265/HEVC [49, 50, 51].

With the increasing of the number of solutions that the encoder has to evaluate, the complexity of the of the coding scheme increases significantly. This is one of the reasons that, for the moment, VVC/H.266 cannot be considered by us a feasible solution in our real coding experiments.

2.1.8 Computational Complexity of Video Coding

Video coding is a resource-intensive, power-hungry task due to the amount of techniques leveraged by modern video codecs to achieve high compression efficiency. Our goal of compressing the cockpit video aboard the airplane must however complain with the constraints on power consumption and heath dissipation typical of avionic devices. Due to such constraints, video encoders are usually implemented in FPGAs to maximize the efficiency per watt and rely on low-complexity implementations of the coding tools where available. The complexity of video encoders usually increases with their efficiency, i.e. each generation of video coding standards is usually more complex to implement than the previous. For example, [1] compares the complexity of H.265/HEVC with AVC/H.264 is. They considered for each component of the codecs, a profiling solution to compute the computational time required by each component part to perform. They concluded the following. An H.265/HEVC encoder that integrate all the functionalities of software model is several times more complex that AVC encoder, with the expected improvements in rate-distortion performances. Substantial more complex are expected to be the entropy coding and in-loop filtering. Further, the implementation cost of modules such as transforms, Intra picture prediction and motion compensation is somehow higher in H.265/HEVC than in AVC/H.264.

From the point of view of the complexity, the literature propose studies to modify the rate-distortion optimisation process to take into account the computational complexity [104, 105, 106, 107]. Complexity-Rate-Distortion analysis is based on the usage of the parametric optimization. The computational complexity of H.265/HEVC is highly dependent on inner data structures, coding units trees requiring the most of the encoding computational complexity. The domains of study aimed are wireless communications where video streaming is realised by limited available energy mobile devices.

For our purposes, we studied the complexity of some H.265/HEVC and AVC/H.264 encoders implemented in FPGA technology, as summarised in Tab. 2.1. The rows compare two implementations of H.265/HEVC and AVC/H.264 encoders in inter-mode, showing that AVC/H.264 power consumption is one third of its H.265/HEVC counterpart. Thus, while AVC/H.264 compression efficiency is lower than H.265/HEVC, its lower complexity makes it an good solution in our power-constrained scenario, as will be shown in the following chapters.

Table 2.1: FPGA complexity estimate for H.264/AVC and H.265/HEVC encoders (1920x1080, 30 fps).

Codec	Logic [kALM]	Power [W]
AVC-Inter [4]	30-60	< 1
HEVC-Inter [5]	90	< 3

2.2 Character Recognition and Encoding

In this section we first provide a primer about Convolutional Neural Networks (CNNs), a class of artificial neural networks widely used for image analysis, then we describe their application to the recognition of characters (letters, digits and other graphical primitives) in images. Next, we introduce lossless compression techniques commonly used to encode text, i.e. sequences of characters.

2.2.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have emerged as the cornerstone to solve a number of computer vision problems such as character recognition [17]. CNNs are feed-forward, multiple-layer, artificial neural networks that may be described as a feature extractor stage followed by an inference stage [31]. 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 [32]. 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 [17]. 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 of entire words [33] and to read text at arbitrary positions in still images [34]. Concerning moving pictures, in [35] 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 chapter.

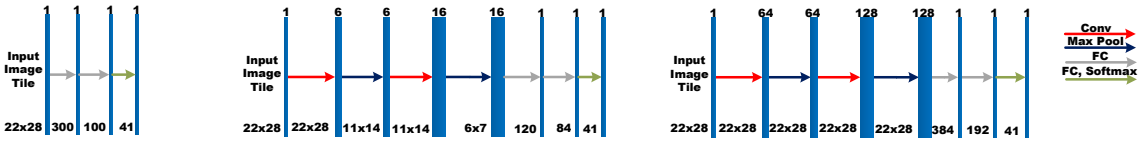


Figure 2.7: The three neural networks architectures considered for on screen character recognition. Left: LeNet300; Center: LeNet5; Right: LeNet5+.

2.2.2 Character reading with CNNs

Following, we will introduce the reference architectures that will be exploited in this thesis, illustrated in figure 2.7. Those are the first ever proposed neural networks for recognition task: LeNet300, LeNet5 and LeNet5+. Each solution offers a different trade off between performance and complexity.

The first architecture is *LeNet300* [17] a simple fully connected network build of two hidden layers with 300 and 100 units (neurons) and one output layer, with sigmoid activation function. The second architecture is the well-known *LeNet5* [17], a convolutional architecture with two convolutional layers and three fully connected layers. The first convolutional layer include 6 5×5 filters, while the second convolutional layer is composed by 16 5×5 filters. Each has sigmoid activations and are followed by max-pooling feature map subsampling. Following, the network includes two fully connected layers of 120 and 84 units respectively with sigmoid activations. The third architecture is an improved LeNet5 (*LeNet5+*, in the following) [37] that add as novelty ReLU activations [32] and an increased number of convolutional filters. All three architectures are composed by 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.

2.2.3 Improving the Performance of CNN-Based Character Reading

A common problem in many machine learning algorithms is that training samples can be obtained from different sources where classes appear with frequencies differing from the test-time, aiming correcting the classifier's output. The literature offers methods for adjusting classifier outputs to new and unknown apriori probabilities [57, 58]. The authors of [59] propose a procedure to accomplish the correction of the estimated a posteriori probabilities of the classifer's outputs, in accordance with the new a priori probabilities of the real-world data, in order to make more accurate predictions, without having to refit the neural network model. They present an iterative procedure of the expectation-maximization (EM) algorithm [60], which aims to maximize the likelihood of the new observed data.

The authors of [61] propose two integrated approaches of individual classifiers to improves classification accuracy. Their approach integrated a rule-based expert system classifier and a neural network classifier. By incorporating expert knowledge in a neural network classifier they achieved the higher classification accuracy. In [62] the authors describe a stochastic error-correcting parsing algorithm as a method to adjust the results of an optical character recognizer (OCR). In [63] the authors introduce a posteriori corrections as computational inexpensive techniques which can improve accuracy by correcting the differences between a priori training and real class distributions for the output of neural and any other classification models.

In [65] the authors propose a corrective tool for arbitrary machine learning models called Black Box Shift Estimation (BBSE). It estimate label shift between training and test set using a black box predictor. Further it detect, quantify the shift and correct their classifier.

In [66] the authors consider another approach of a decomposition strategy where each simple problem is binary. They proposed a simple solution for combining binary classifiers with inexact probabilities as outputs, consisting in computing a set of probability distributions by solving a global optimization problem whose constraints depend on the classifiers outputs.

The literature proposed another class of methods that are using error correcting output coding algorithm (ECOC) to correct the output of different task models. The author in [67, 68] applies error-correcting output coding to the task of document categorization and for learning text classifiers. In [69] the authors applies error-correcting output coding to pedestrian detection.

Compared to our second proposed method, complexity is not a design constraint that is taken into account by all relevant articles in the literature.

With respect to the above literature, we proposed to explore the temporal redundancy both for coding and for recognition, using proposed low complex methods to meet the requirements for avionic applications being able to be quasi-errorless on the character recognition experiments and improving both in compression and character readability.

2.2.4 Computational Complexity of Character Recognition Task

There are various methods for object detection based on CNNs and diverse application of them. We are interested in study the complexity of those methods, more precisely, in FPGA-based implementations, given their flexibility property.

The real time object detection and recognition in videos is achieved by the reference You Only Look Once (YOLO) architecture. The authors define the object detection problem as a regression of sparse bounding boxes and of associated class probabilities, predicting bounding boxes and class probabilities from full frames in videos in one network pass [124]. In the study [125], the authors proposed a resource-aware optimised framework for object detection and recognition in videos based on YOLO, on FPGA. In average, the system consume 20 W, representing almost half from the allowed power consumption in avionic applications. In another article [126], the authors compare their FPGA proposed implementation of YOLO with an equivalent implemented in CPU and GPU. The power consumption of the latter was 105 W and 250 W, respectively, just to give an order of magnitude. For the FPGA architecture the total on-chip power consumption was 5.96 W. The article is a good reference point for other much more complex FPGA implementations of YOLO. The article [127] present an FPGA implementation that consume 7.78 W. In the same spirit, the article[128] present several FPGA implementations of YOLO that are able to work real time.

Those are examples, that demonstrate that, nowadays, is difficult to consider a fully automated detection and recognition of objects in video for avionic applications, even if they are optimised. The constraints come from the FPGA hardware technology that supports the network or from the working frequency, for example.

During the last years, there have been several attempts to accelerate the hardware implementations of CNN for more efficient resource utilisations [129]. For example, the authors [130] propose an implementation of a simpler CNN for object recognition in videos

Table 2.2: Complexity of the three architectures considered for on screen character recognition.

	LeNet300	LeNet5	LeNet5+
# Params	215.8k	**60k**	30M
MACs	**219k**	679k	32.7M

that can handle images at 82 frames per second. Even if their implementation outperforms existing FPGA implementations they do not report the complexity dimension of power consumption, which maybe substantial. A simpler network is proposed in [131] to accomplish optical character recognition task for real time License Plate Recognition. The authors manage to achieve a quarter of memory utilisation in comparison with their reference architectures.

Given all above, for the aims of this thesis, table 2.2 reports the complexity of the three architectures: LeNet300, LeNet5 and LeNet5+. Those architectures are with several orders of magnitude much simpler than YOLO. Memory complexity is reported as number of learnable parameters and computational complexity as number of Multiply-Accumulate (MAC) operations [39]. The conclusion is the following: LeNet5+ is the most complex, LeNet300 shows the lowest computational complexity, whereas LeNet5 shows the lowest memory footprint.

2.2.5 Lossless Data Compression for Text Encoding

In lossless coding (or entropy coding), each symbol is assigned a code whose length depends on its occurrence probability. Highly probable symbols get short codes and less probable symbols are assigned longer codes. Dictionary compression is where groups of consecutive symbols, are replaced by a code by searching it in dictionary. Dictionary coding techniques rely on recurring patterns of the data, those repetitions being replaced by shorter references to a dictionary representing the original. Arithmetic coding replaces a sequence of input symbols with a single floating point number. Lempel-Ziv compresses symbol strings sequentially, mainly by replacing sub-strings with pointers to match sub-strings previously encountered [96, 97]. The Burrows-Wheeler Transform (BWT) takes as input a block of data, and permutes the symbols in that block to form an output block. To permute the block means that symbols following or preceding a particular sequence of symbols in the input will be adjacent in the BWT output. The overall idea behind lossless coding algorithms is to preprocess the information exploiting the natural redundancy of the language in making the transformation. As examples, in these articles [98, 99] the authors use the block-sorting mechanism to word-based model being implicit that block-sorting implementations are using as input message a sequence of characters. The authors [100] presented a method for compressing small text files using the Burrows-Wheeler transformation.

The literature proposed algorithms for text data compression and transmission, in order to improve the quality and efficiency of wireless data transmission. In [70, 71], the authors discussed a Huffman coding solution for wireless transmission of GPS text data. The authors [72] presented a neural network probability prediction model of the maximum entropy principle combined with the optimized Huffman encoding algorithm in order to optimize the efficiency of vehicle text data compression and transmission.

Optimized Lempel–Ziv–Welch (LZW) compression algorithms are proposed by the authors in [73, 74, 75, 76].

While we share with these methods a form of grouping of the characters in order to perform coding we explore further features of screen characters and graphics, namely spatial and temporal domains of the cockpit screen content.

2.3 Related Work and Prior Results

After reviewing techniques and standards for video compression or text encoding in the previous sections of this chapter, we now divert our attention to techniques targeting compounds of natural images and text, that we will refer to as mixed content.

2.3.1 Mixed Content Images and Compound Video Coding Solutions with or without Temporal Prediction

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 [19].

A class of schemes focusing on still image 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 [6, 7] 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 [8], the authors propose a solution for encoding the content using a block-based segmentation method to differentiate between the objects of a compound image. In [9], 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 [20] [21]. For example, DjVu [21] is able to separate an image into a background layer (i.e., texture or pictures) and foreground layer (text or line drawings). It is able to preserve readability of the text while compressing the backgrounds at lower resolution with a wavelet-based compression technique. 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 [10, 11], 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 [22], 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 [12] 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 [13] 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.

In [79], the time axis is considered and the text is still compressed with a pixel-based approach. This article shows that it is difficult and never done before to compress text as such in the temporal dimension. In the text layer, the text block is compressed in the spatial domain and skip block is directly copied from the same block in the previous frame. The pixels are quantized to several major colors and the indices of these major colors are entropy coded to achieve a high efficiency compression. These methods are similar with the following [80], where the temporal coding of text regions is still performed at pixel level.

In [81], the authors correctly make a difference between compound image compression and compound video compression. They proposed block-based compression techniques using block classification into text and natural picture, being used to adjust the the quality over the coding rate. Another contribution is an adapted rate control algorithm which takes into account the adaptive quantization. Compound video compression method similar in spirit to the previous papers is proposed in [82]. In [83] a layered structure for screen coding is proposed with different rendering frame rates. The interaction content with small region screen is compressed by a blockwise screen codec while the natural video screen content is compressed by standard video codec.

Another solution to compress screen images is proposed by the literature [95]. It is based on a probability distribution of pixels and the relationship between probable pixel colours and surrounding texture is learned by a predictor. The method is called soft context formation (SCF) and is used to achieve much lower bitrate for images with a small number of colors.

While we share the same basic principle of segmenting the content in order to preserve the quality of some parts of the videos, those methods do not consider complexity in their designs.

2.3.2 Intelligent Analysis Methods for Semantic Video Compression

In [89], the authors describe a framework for pedestrian detection that extract the foreground objects from the video sequence by modelling the background. The encoder will allow a higher quality for foreground sequence, while for the few background pictures it will compress them in lower quality. The article [90] introduced a knowledge-based coding method for encoding multisource surveillance video data, to exploit the redundancy induced by multiple video sources. The authors of [91] proposed an object-based video coding approach for preserving the quality of target objects in videos when the coding rate saving becomes important in the case of a disaster. The objects are extracted using saliency and temporal correlation and the distance between the objects and the rest of the scene is used to decide the quantization of the background encoded with HEVC.

Increasing the complexity of the proposed methods, other articles put forward deep learning methods to detect the preserved object in images or videos. In [92] the authors detect vehicles in videos acquired from an airborne camera to improve the coding efficiency of georegistered aerial videos. The articles [93, 94] describe semantic analysis methods for image compression using deep learning models.

2.3.3 MPEG-4 Standard for Subtitles Coding

From another perspective, the standardization committee proposed a extension to the MPEG-4 standard for subtitles coding. It has several important features such as temporal synchronization and exact text positioning [84]. Similarly, in [85] the authors proposed a

coding technique for the text content of the multimedia presentations. Several patents for subtitle encoding are proposed in the literature either pixel-based or text-based [86, 87, 88].

While for this kind of methods we share the exploitation of spatial redundancy, those methods fail at considering the temporal aspect in performing compression.

2.3.4 Frame Buffer Compression Schemes

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 [28] [29]. In particular, the Video Electronics Standards Association (VESA) has recently standardized a protocol for quasi-lossless framebuffer [30]. 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) being an important component in modern connectors like DisplayPort or HDMI cables. 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 work.

2.3.5 Screen Content Coding Extension of H.265/HEVC

The recently standardized Screen Content Coding (SCC) extension of the H.265/HEVC (High Efficiency Video Coding) standard [16] 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) [23] 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) [24] 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.

Adaptive Color Transform (ACT) [52, 56] helps to decorelate the color components by color space transformation from RGB domain to YCoCgR space. This method is applied to TU level in an adaptive manner.

Cross-Component Prediction (CCP) [52, 56] is used to explore the inter-component redundancy between the three components of the residual. This method is applied for intra and inter-predicted TUs.

Transform Skip (TS) [52, 56] is a coding solution which decides to skip spatial transform providing important gains. Its usage is signaled at TU level.

Adaptive Motion Vector Resolution (AMVR) [52, 56] is applied at slice level and it allows motion vectors to switch from quarter-pixel to integer-pixel resolutions, much of the screen content being characterised by integer-pixel motion.

Regarding the complexity, the memory access is considered the major problem. Namely, IBC require the same computational complexity as the Inter mode in H.265/HEVC, while PLT require less computational complexity. Optimisation techniques was considered for ACT and CCP since the color components can be explored in parallel. Regarding TS and AMVR the complexity is negligible in comparison with Inter mode in H.265/HEVC.

Optimisation techniques were considered, such as a pre-analysis of the slice for AMVC to avoid two parsing of the slice to do the encoding [52, 56].

A number of improvements to the SCC extension have been proposed. For example, in [25] 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 [26], a residual differential pulse code modulation (RDPCM) coding technique using a weighted linear combination of neighbouring residual samples is proposed. In [27], 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, extension of HEVC and its derivatives, our experiments revealed that at very low bitrates imposed by our application artifacts around text are unavoidable, motivating the semantic compression scheme proposed in the present work.

Chapter 3

Very Low Bitrate Semantic Compression of Airplane Cockpit Screen Content

This chapter 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. First, the text readability 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 HEVC/H.265 encoder and its SCC extension. Moreover, our scheme achieves quasi-errorless character recognition already at very low bitrates, whereas even SCC needs at least 3 or 4 times more bit-rate to achieve a comparable error rate.

3.1 Introduction

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 [6, 7, 8, 9, 10, 11, 12, 13, 14]. 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 [15] of the H.265/HEVC [16] 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 in terms of their semantics rather than as blocks of pixels and, 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 compression particularly challenging.

In this chapter, 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) [17] 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 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 [18] and the residual video is compressed with 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 SCC and around 82% savings with respect to HEVC/H.265. That is, our scheme based on the compression of the semantics of text (and lines) enables 4 to 6 times lower bandwidth than a reference scheme where 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 chapter 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 chapter is organized as follows. The necessary context of airplane cockpit screens video is presented in Section 3.2. In Section 3.3 we describe our proposed system for low-bitrate airplane cockpit video coding. Then, in Section 4.3 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 4.4 draws the conclusions of this chapter and outlines future research directions discussed in the following sections.

3.2 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. 3.1(a), 3.1(b), 3.1(c) or 3.1(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. 3.1(a) is a color image captured in the visible light spectrum, whereas Fig. 3.1(b) has grayscale background captured in the infrared band or in low-light conditions (Fig. 3.1(c)). The background in figure 3.1(f) is represented by MPEG sequence called Cactus to better evaluate the resilience of the character detector.

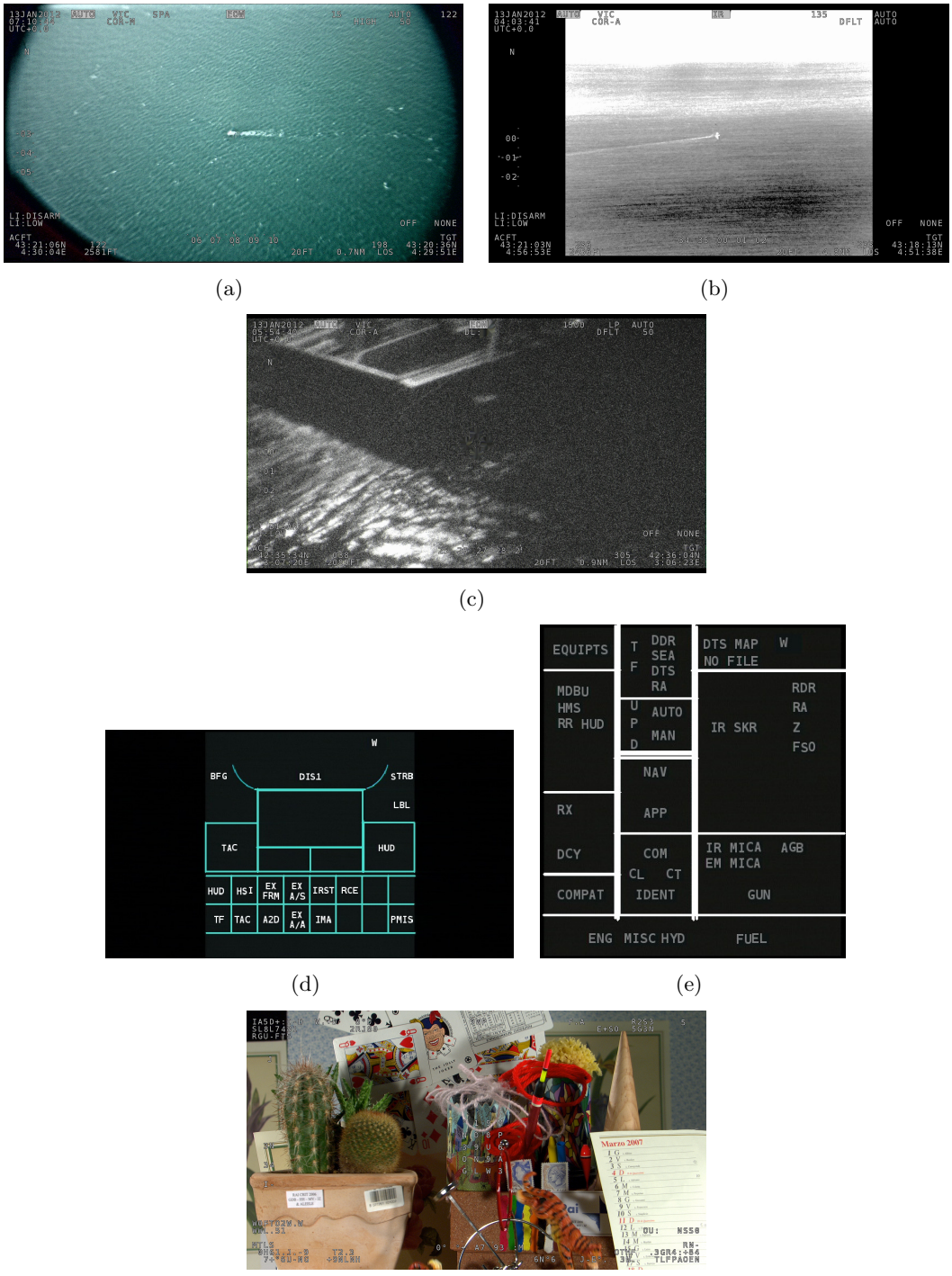
Such screens typically assist the pilot in some specific tasks (e.g., let the second pilot inspecting the ground for obstacles on the take off/landing stripe). Due to the unpredictable background appearance, overlaid text usually has contour to improve its readability (e.g., black text with white contour). The second class of cockpit screens we consider is shown in Fig. 3.1(d) and Fig. 3.1(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 compressing 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 differences 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.

3.3 Proposed Method

This section describes our scheme for airplane cockpit screens content semantic compression, as illustrated in Figure 3.2. Text is first localized exploiting some position-invariant screen features (Sec. 3.3.1), then characters are recognized using a CNN (Sec. 3.3.2) and predictively compressed (Sec. 3.3.3). Lines are similarly detected and encoded (Sec. 3.3.4). Finally, text and lines are removed from the frame and the residual video is encoded via SCC (Sec. 3.3.5). At the decoder, the residual video is decoded and text and lines are synthesized avoiding reconstruction artifacts.

3.3.1 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. 3.3.



(a) (b)

(c)

(d) (e)

(f)

Figure 3.1: Example of different types of airplane cockpit screens. First row - Left (Fig. 3.1(a)): class A - natural light with color background. First row - Right (Fig. 3.1(b)): class A - infrared vision with grayscale background. Second row (Fig. 3.1(c)): class A - low-light vision with grayscale background. Third row (Fig. 3.1(d) and Fig. 3.1(e)): class B - computer-generated background. Fourth row (Fig. 3.1(f)): class A - synthetic content.

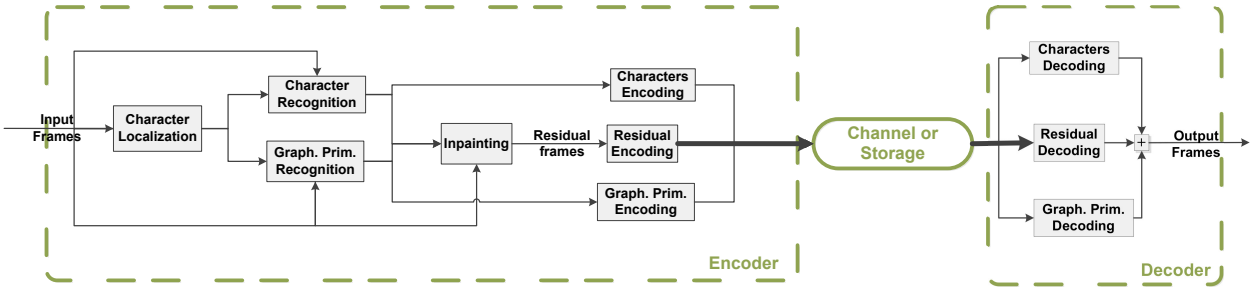


Figure 3.2: Architecture of the proposed scheme for encoding airplane cockpit video at low bitrates: the semantics of computer generated graphics (text, lines) are relayed to the decoder separately from the compressed residual video.

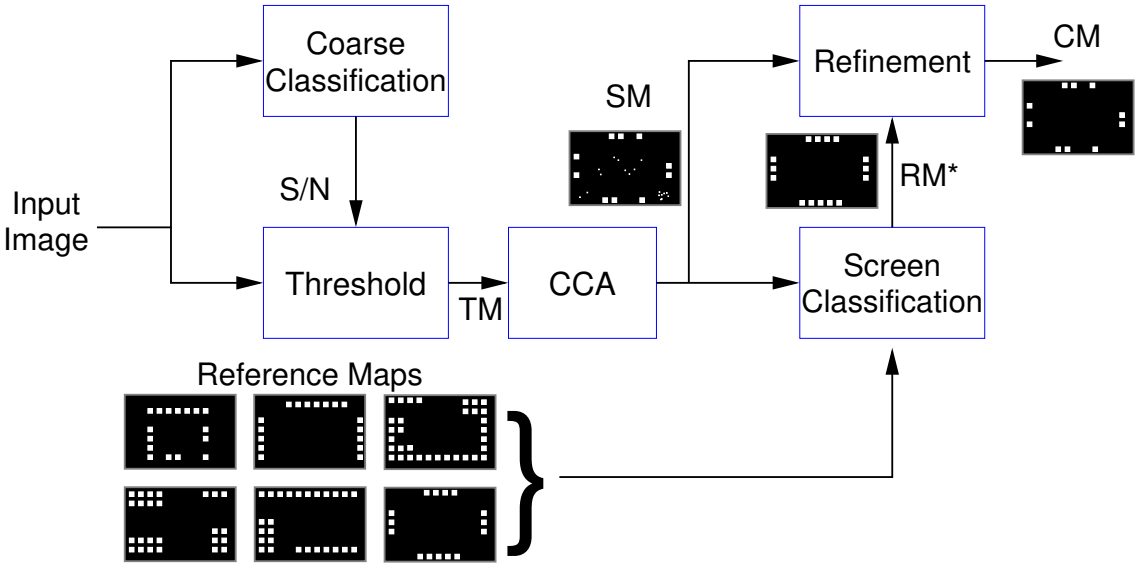


Figure 3.3: Character localization procedure.

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. 3.1(a) show text with contour, whereas text in monochrome-background class B screens in Fig. 3.1(d) has no contour).

As a first preliminarily step, we perform a *coarse classification* of the image (Fig. 3.3) 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 such 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. 3.4(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. 3.3) takes as input the image and produces as output a *threshold map* TM exemplified in Fig. 3.4(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. 3.1(d), the Otsu [36] thresholding scheme is used. If the image is natural (like Fig. 3.1(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 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 white pixels. 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. 3.3) 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. 3.4(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. 3.4(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 is shown in Fig. 3.1(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 white 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. 3.4(c) and the *reference map* in Fig. 3.4(d) is the *characters map* in Fig. 3.4(e). Notice how the *characters map* includes a bounding box of the actual character size for each character actually present in Fig. 3.1(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. 3.3.2. In the end, the associated table of characters, suitable for coding, is obtained as it is elaborated in Sec. 3.3.3.

3.3.2 Character Recognition

This section discusses the three neural network architectures for on-screen character recognition: LeNet300, LeNet5 and LeNet5+. 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 networks are trained over character samples extracted from the annotated video sequences. With reference to the sequences in Fig. 3.1(a) and 3.1(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 plus one background class for a total of 41 character classes. 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 [38]. 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 3.1 reports some preliminary experiments on the performance-complexity trade offs of the three architectures. Performance is reported in terms of character recognition

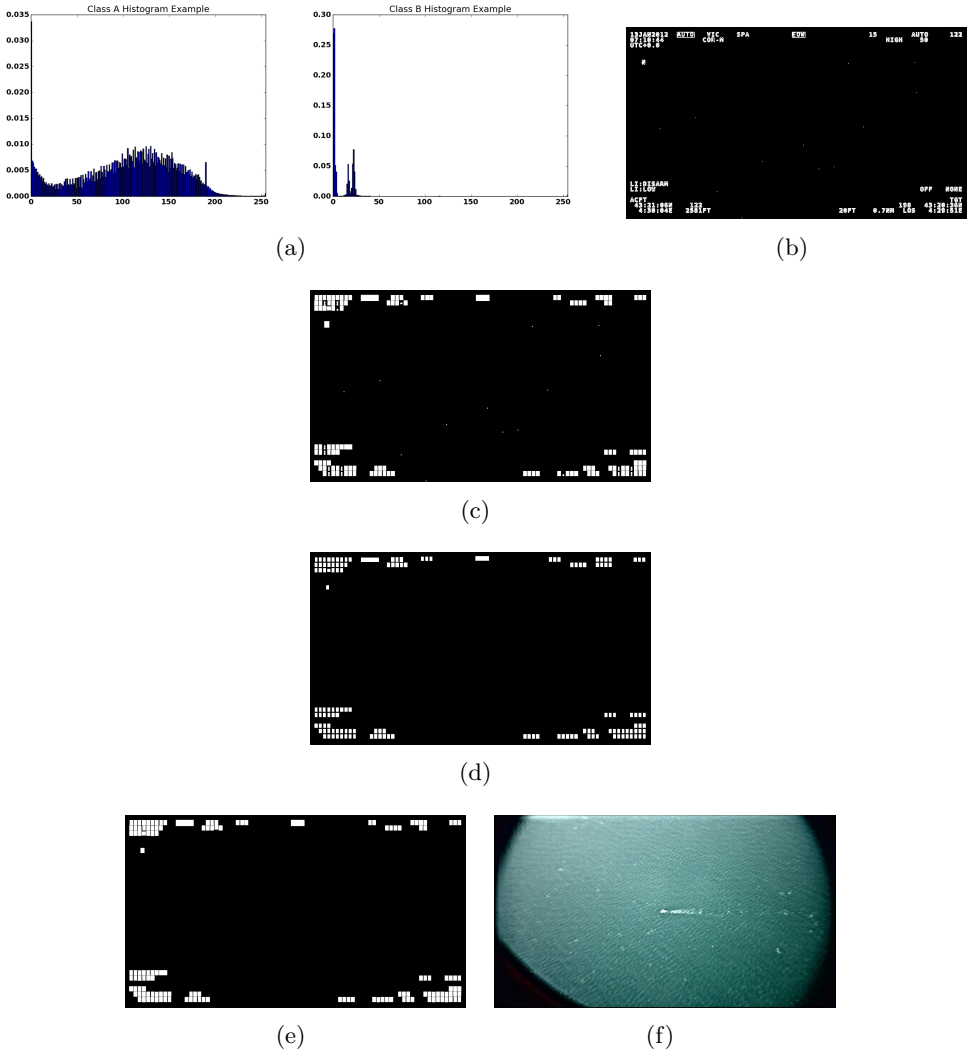


Figure 3.4: Intermediate processing steps for the class A image in Fig. 3.1(a). First row - Left (Fig. 3.4(a)): Histograms examples. First row - Right (Fig. 3.4(b)): Threshold map. Second row (Fig. 3.4(c)): Segment map. Third row (Fig. 3.4(d)): Reference map. Fourth row - Left (Fig. 3.4(e)): Predicted characters map. Fourth row - Right (Fig. 3.4(f)): Residual after character inpainting.

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 [39]. 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 3.1: 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**

3.3.3 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 [40] 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 ± 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

Table 3.2: 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

value of d_n^v is encoded using a simple variable-length code (VLC), see Tab. 3.2. 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 summarized in Tab. 3.2.

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.

3.3.4 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. 3.1(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 [41]. In particular we use the line detector described by Matas *et al.* [42], 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 $2N_L\,(\lceil \log_2 W \rceil + \lceil \log_2 H \rceil)$, 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 algorithm 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 corresponding 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 ±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 length 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.

3.3.5 Residual Coding

Eventually, computer-generated graphics and text are removed from the video, filling the resulting gaps via inpainting [18] 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 [43] method. The residual frame is characterized by smooth color distribution in the inpainted areas as shown in Fig. 3.4(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/H.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 (eg. in figure 3.1(d)), and those must be found in the residual frame, at a given quality, at the decoder side.

3.4 Experimental Results

We validated the proposed compression scheme by comparing it with three references: the HEVC/H.265 codec (*HEVC* for short), its SCC extension (*SCC*) and the screen content coding technology DjVu [21] (*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. 3.1. 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. 3.1(a) or in the infrared spectrum as in Fig. 3.1(b), or in low-light conditions as in Fig. 3.1(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.3.1(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. 3.1(d) and 3.1(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 3.4.1. 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 3.4.2.

3.4.1 Rate-distortion performance

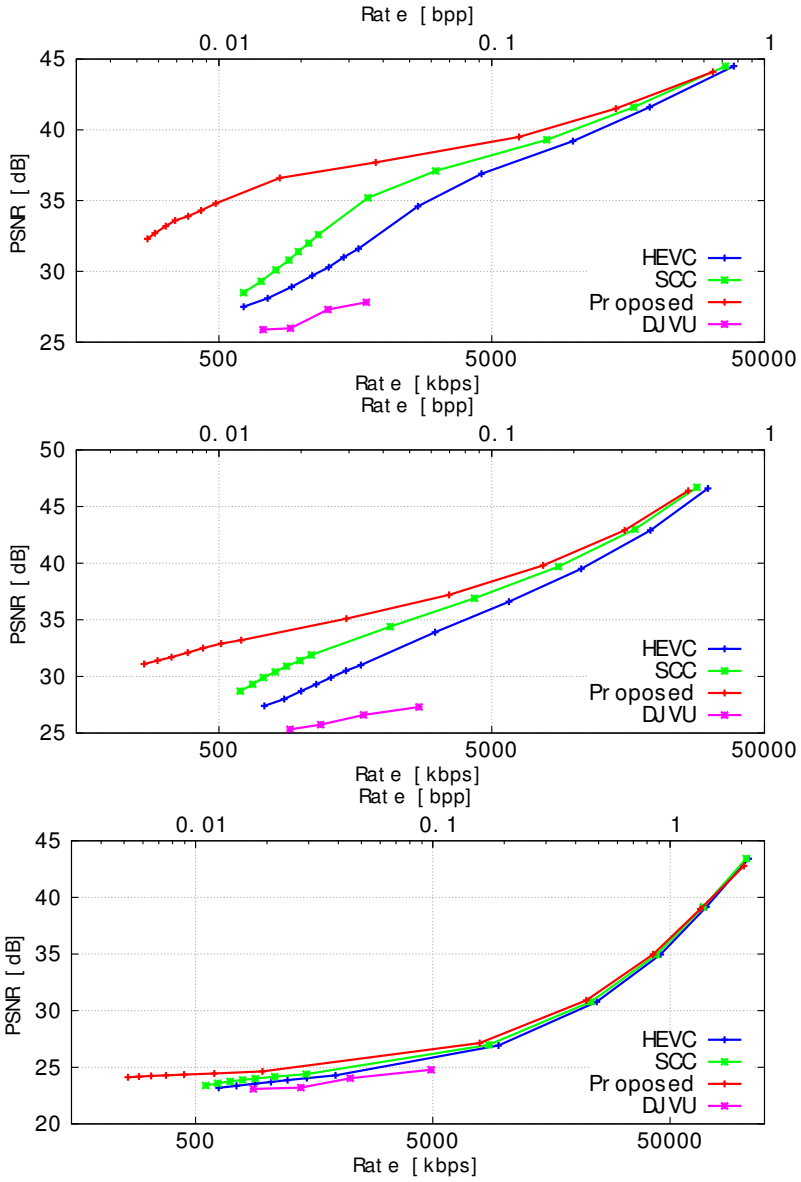
The rate-distortion curves of the four methods (the proposed one and the three references) are shown in Fig. 3.5 for the six video sequences. From these figures, a few interesting facts emerge.

First, the proposed scheme is able to attain a minimum rate much smaller than HEVC and SCC. Typical 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, which is the object of our interest. This is already visible in Fig. 3.5, but for a more precise comparison we also compute the Bjontegaard metrics (BD-Rate and BR-PSNR) [44] for the four class-A sequences, see Tab. 4.3. At low bitrate, we gain no less than 30 % with respect to SCC and up to more than 80 % with respect to HEVC. Bjontegaard metrics cannot be computed for class-B sequences since the rate

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


Rate [bpp]
0.01
0.1
1
PSNR [dB]
45
40
35
30
25
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed
DJVU
Rate [bpp]
0.01
0.1
1
PSNR [dB]
50
45
40
35
30
25
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed
DJVU
Rate [bpp]
0.01
0.1
1
PSNR [dB]
45
40
35
30
25
20
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed
DJVU

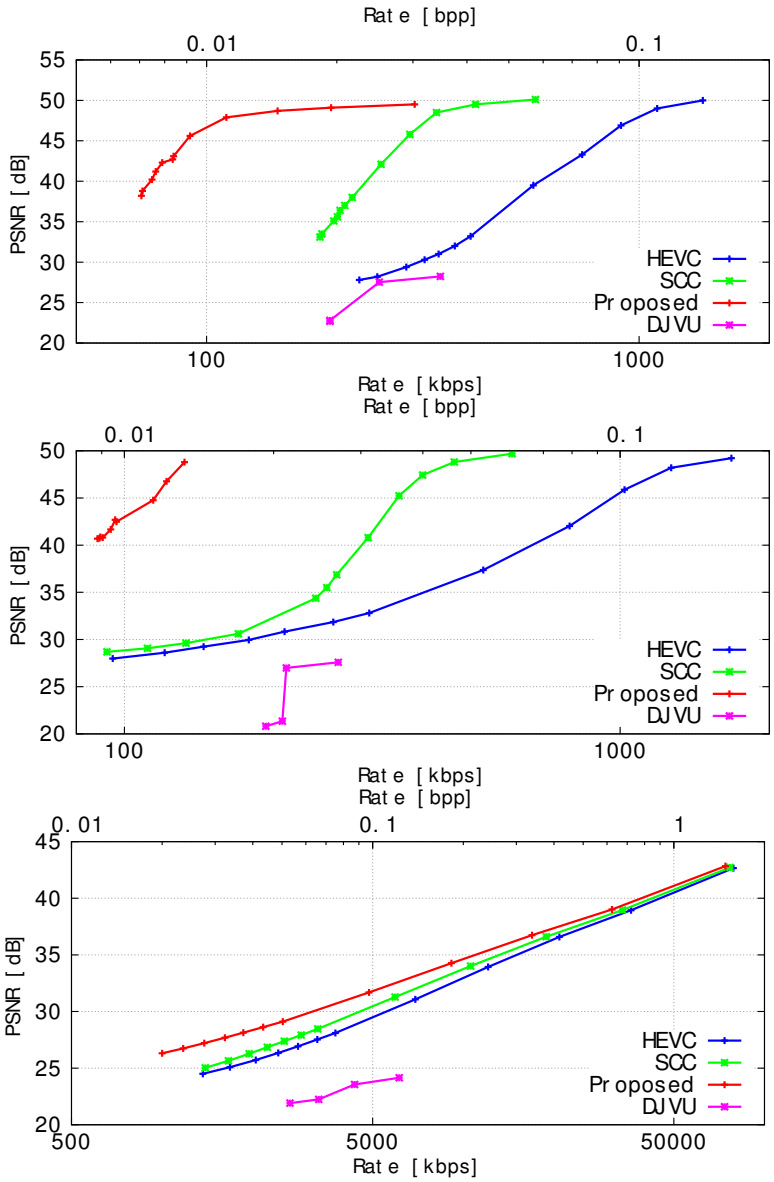


Figure 3.5: PSNR vs. video bitrate. First row: class-A color (Fig. 3.1(a)); Second row: class-A gray-0 (Fig. 3.1(b)); Third row: class-A gray-1 (Fig. 3.1(c)); Fourth row: class-B 0 (Fig. 3.1(d)); Fifth row: class-B 1 (Fig. 3.1(e)); Sixth: class-A synthetic (Fig. 3.1(f)).

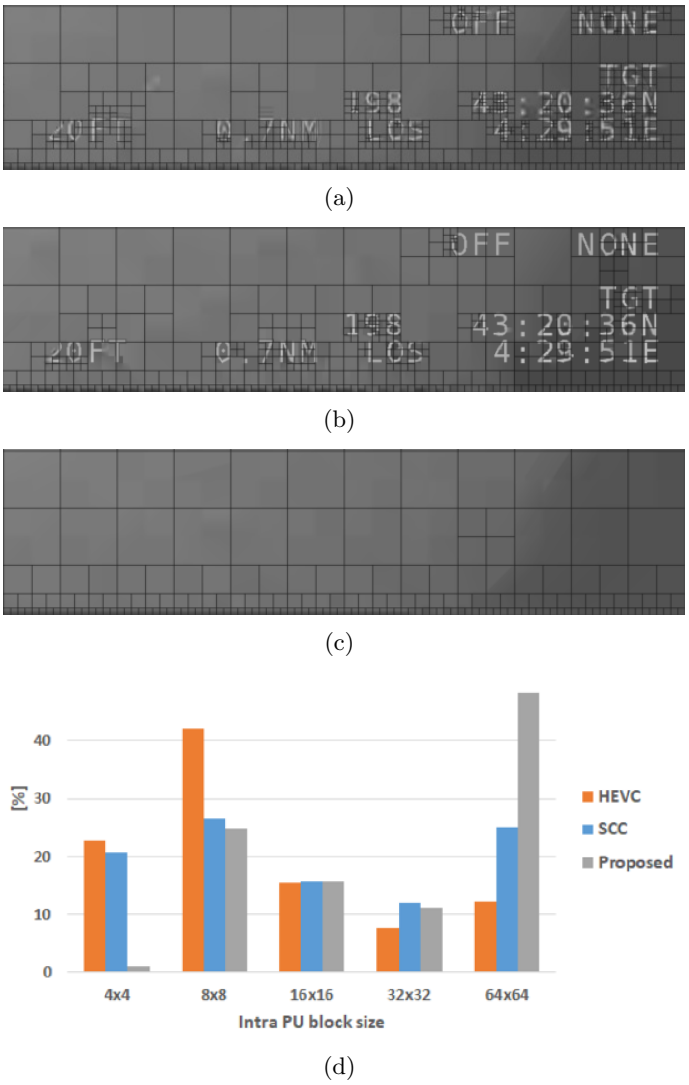


Figure 3.6: PUs for the first frame of the class-A color sequence for QP 50. Top: HEVC; Middle: SCC; Bottom: proposed scheme. The corresponding PUs size distribution is reported in the last figure.

overlap is too small. However, we can see from the RD curve in Fig. 3.4.1, 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 compressed with the *ad hoc* algorithms described in Sect. 3.3.3 and 3.3.4. To gain further insight about the reason of our gains, we show in Fig. 3.6 the partitions of one frame from a class-A sequence together with the histogram of 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.

Table 3.3: 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 $QP = [50, 45, 40, 35]$	
Prop. vs.	Sequence	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
SCC	color	1.04 dB	-37.66 %	4.59 dB	-69.04 %
HEVC	color	1.86 dB	-53.25 %	7.12 dB	-81.98 %
SCC	gray-0	0.84 dB	-20.43 %	2.18 dB	-52.69 %
HEVC	gray-0	2.06 dB	-39.58 %	4.30 dB	-72.36 %
SCC	gray-1	0.32 dB	-5.55 %	0.43 dB	-39.98 %
HEVC	gray-1	0.63 dB	-10.2 %	0.75 dB	-46.83 %
SCC	synth.	0.74 dB	-16.34 %	1.5 dB	-30.46 %
HEVC	synth.	1.3 dB	-26.4 %	2.42 dB	-42.96 %

We also report in Fig. 3.7 and 3.8 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. 3.7, the resulting coding rates are 0.015 bpp for HEVC (left), 0.014 bpp for SCC (center) and 0.005 bpp for ours (right), and the associated PSNRs are respectively 28.1 dB, 29.3 dB and 32.7 dB. Similar values are obtained for the image in Fig. 3.8. 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. 3.9 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.

3.4.2 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 3.7 and 3.8. 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

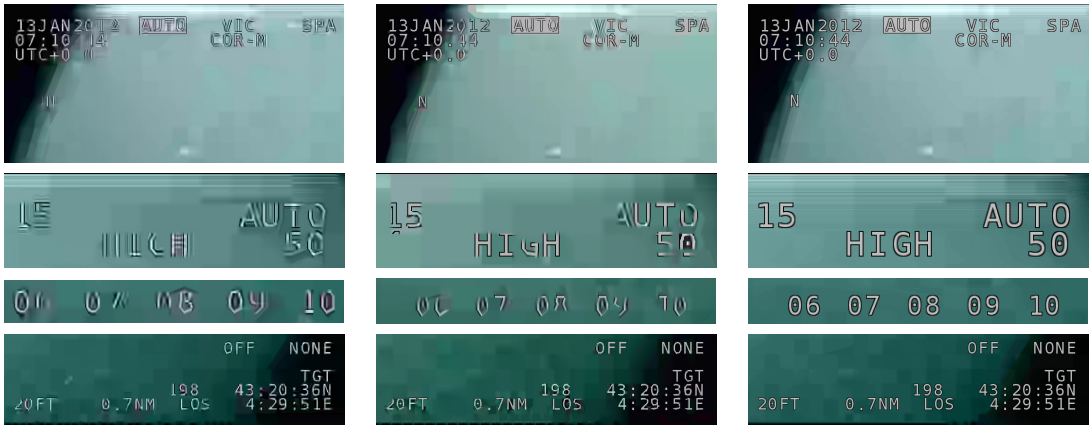


Figure 3.7: Reconstruction artifacts for the class-A color decoded video at QP=50. Left: HEVC; Center: SCC; Right: our proposed scheme.

Figure 3.8: Reconstruction artifacts for the class-A grayscale-0 decoded video at QP=50. Left: HEVC; Center: SCC; Right: our proposed scheme.

(a)

(b)

(c)

Figure 3.9: Reconstruction artifacts in background video, class-A color sequence encoded at 0.0018 bpp. Top: HEVC; Middle: SCC; Bottom: our proposed scheme.

on character readability can be obtained much more easily by partly exploiting the ideas of automatic character recognition shown in Sect. 3.3.2.

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. 3.3.2. 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. The ground truth was considered the robust proposed scheme.

In all the cases, we use the same training procedure described in Sec. 3.3.2, 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. 3.10(a), 3.10(b), 3.10(c), 3.10(f) for class-A color, class-A gray-0, class-A gray-1 and Class-A synthetic; and in Fig. 3.10(d) and 3.10(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 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

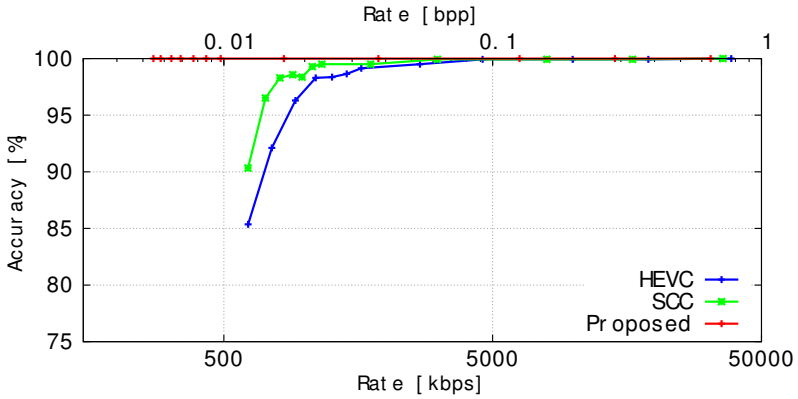
in the left and central column of Fig. 3.7 and 3.8, the decoded characters are affected by heavy reconstruction artifacts that severely compromise the character readability.

3.4.3 Power Consumption

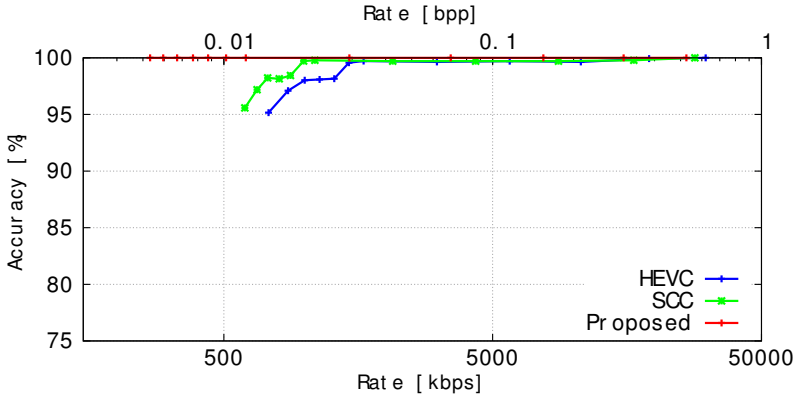
Finally, we estimate the the power consumption of our proposed video coding system. To allow for passive heating, the power consumption of our complete system shall fall in the 20 to 35 W range. To meet such constraint, computationally complex operations will be offloaded to a specialized hardware such as FPGAs and ad-hoc ICs. Concerning character recognition, in [45] an optimized LeNet5 implementation with a 3W power consumption envelope is proposed, leveraging Xilinx deep learning library [46]. 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 [47] 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 inpainting, such operations could also be efficiently offloaded to a FPGA relying on standard computer vision libraries for a few additional Watts [48]. 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.

3.5 Conclusions

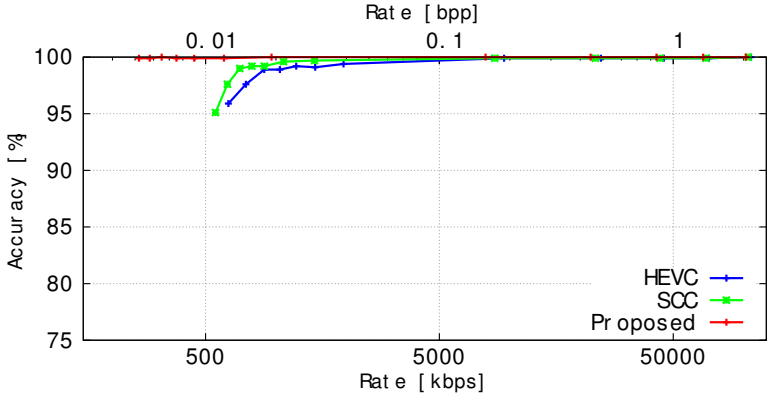
In this work we proposed a scheme for compressing airplane cockpit video at low bitrates without affecting the readability of computer generated graphics (text, lines). At the encoder, we first compress the graphics exploiting their semantics, then the residual video obtained by inpainting the graphics 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 will exploit, in the next chapter, the inter frame temporal correlation.


Rate [bpp]
0.01
0.1
1
100
95
90
85
80
75
Accuracy [%]
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed

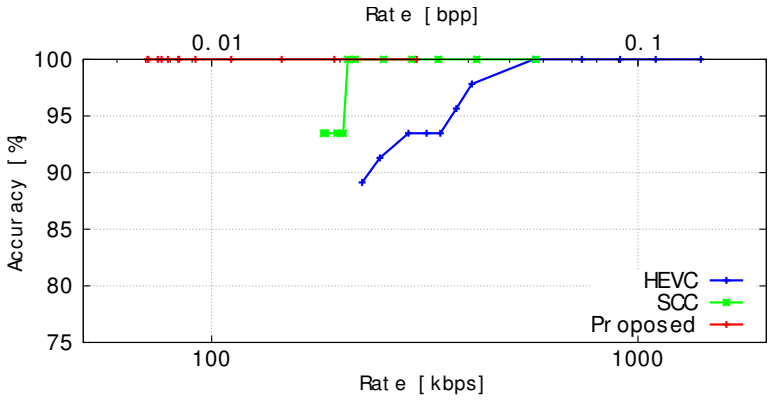
(a)


Rate [bpp]
0.01
0.1
1
100
95
90
85
80
75
Accuracy [%]
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed

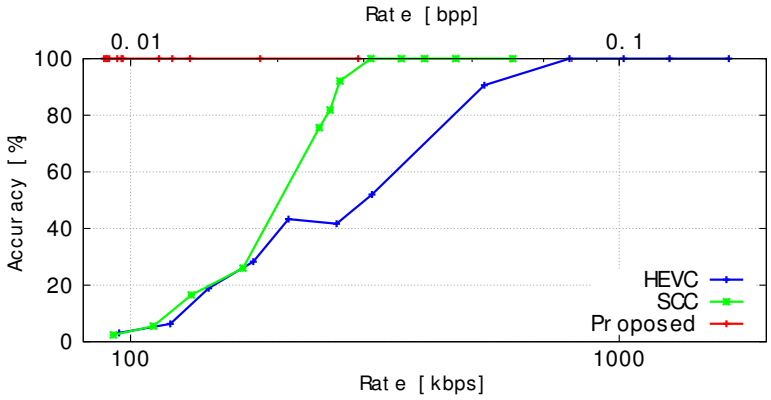
(b)


Rate [bpp]
0.01
0.1
1
100
95
90
85
80
75
Accuracy [%]
500
5000
50000
Rate [kbps]
HEVC
SCC
Proposed

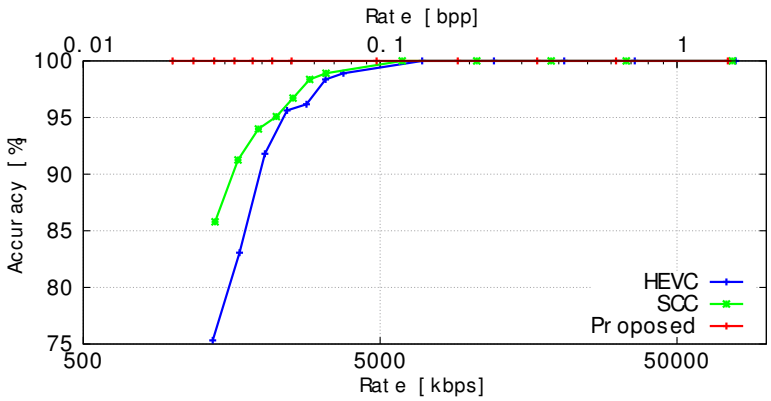
(c)



(d)



(e)



(f)

Figure 3.10: Accuracy in character recognition as a function of video bitrate. First row: class-A color (Fig. 3.1(a)); Second row: class-A gray-0 (Fig. 3.1(b)); Third row - Left: class-A gray-1 (Fig. 3.1(c)); Fourth row: class-B 0 (Fig. 3.1(d)); Fifth row - Left: class-B 1 (Fig. 3.1(e)); Sixth row: class-A synthetic (Fig. 3.1(f)).

Chapter 4

Cockpit Video Coding with Temporal Prediction

In the previous chapter we proposed a scheme for semantic compression of airplane cockpit video that preserves the readability of text while meeting bitrate and encoder complexity constraints. Within each frame, text is segmented from the video and encoded as character strings rather than as pixels. Text in the screen is then inpainted, producing a residual video with few high frequency components easily encodable with standard codecs. The residual video is transmitted with the encoded text as side-information. At the receiver side, characters are synthesized atop the decoded residual video, leaving the text unaffected by compression artefacts. In this chapter, we evaluate our scheme with multiple video codecs with different prediction schemes, producing novel experimental evidence in terms of attainable rate-distortion performance and highlighting directions for future research.

4.1 Introduction

We have shown in the previous chapter a screen compression scheme where computer-generated graphics (characters, lines) are segmented from the video at the source. The characters are recognized via a Convolutional Neural Network (CNN) and are encoded as text rather than as pixels. Characters are then encoded using a rate-efficient intra-frame predictive scheme and delivered to the receiver. Characters are removed from the video via inpainting [18] and the *residual video* is compressed with H.265/HEVC video codec. At the receiver, the characters are synthesized from scratch and overlaid on the decoded residual video, recovering the original stream. Removing high-frequency components from the video reduces the video bitrate, whereas the characters synthesized at the receiver are not affected by compression artefacts.

This chapter extends and improves upon the results of the Chapter 3 in two ways. First, we explore the potential gains enabled by an inter-frame predictive scheme in the context of cockpit video compression. The method in Chapter 3 being intra-frame only, the gains were limited by the fact that we disregarded the temporal correlation among neighbouring frames. Second, we explore a lower complexity solution where the residual video is compressed via AVC/H.264 instead of HEVC/H.265. Such solution addresses the need for an encoder [1] implementable in FPGA, while it enjoys the benefits of the well established licensing program of the AVC/H.264 standard. Our experiments show that AVC/H.264 retrofitted with our semantic coding scheme is competitive with a reference

HEVC/H.265 encoder in rate-distortion sense, plus they highlight promising directions for further enhancing the performance of our scheme.

4.2 Semantic Cockpit Video Compression

This section overviews the key features of our method for low-bitrate cockpit video semantic compression, illustrated in Fig. 3.2.

4.2.1 Character Localization

As a first step, we localize the characters in the screen exploiting some features of cockpit screens to keep complexity low. Namely, characters aspect in cockpit screens changes depending on the background type (natural image or uniform colour) to improve readability. Characters are in fact outlined when superimposed on a natural images (typically, white text with black background), whereas there is no outline when the background is monochrome.
Therefore, we first coarsely classify each screen either as having natural background (*natural* screen) or monochrome background (*computer-generated* screen) as follows. Each frame is first subdivided in tiles and for each tile we compute the colour histogram: if most histograms are spiky, the screen is labelled as synthetic, otherwise the screen is labelled as natural.
Then, depending on the screen class, pixels corresponding to text are segmented with an appropriate thresholding algorithm. Concerning natural screens, characters are segmented leveraging the knowledge they have an outline. Two thresholding operations are combined, one on the letters colour and another one on the contour colour. Otherwise, the Otsu thresholding [36] algorithm is used for computer-generated screens. Both thresholding algorithms produce a binary *threshold mask* where , e.g. white, pixels correspond to characters.
Further, Connected Components Analysis (CCA) clusters the white pixels in the threshold mask assigning the same label to pixels in the same *neighbourhood*. E.g., the comma and the dot forming a semicolon are assigned the same label because they belong to the same letter.
Next, a rectangular bounding box is casted around each pixel cluster with identical label, representing the location of each candidate character as a rectangular bounding box.
The output of the character localization algorithm is finally a set of bounding boxes where each box represents a candidate letter or digit.

4.2.2 Character Reading via Convolutional Neural Network

Each potential character detected above is recognized using a neural network. In the previous chapter we explored three different architectures with different performance-complexity trade-offs. We describe here the solution based on the *LeNet5* architecture [17], which showed a character recognition accuracy of 99.84% at reading computer generated characters and with bearable complexity for FPGA implementation. The network is composed by two convolutional layers and three fully connected layers. Each convolutional layer includes 6 and 16 5×5 filters each layer followed by a max-pooling feature map subsampling layer. The first connected layer includes 120 units whereas the second layer includes 84 units. The output layer finally includes 41 units as the network classifies each character image according to C=41 labels (letters, digits, symbols plus one *no*

character class). We train the network over character samples we extracted from a set of hand-annotated airplane cockpit videos. Training samples are augmented by randomly shifting each character to achieve robustness to character localization errors. The networks is trained to minimize the classification error across the classes using the SGD method over batches of 128 samples and with a learning rate of 10^{-2}. The network outputs a most likely class for each potential localized character, or it rejects the input in the case of, e.g., background clutter erroneously localized as a character, achieving robustness against noisy backgrounds.

4.2.3 Predictive Character Encoding

For each video frame, detected characters are encoded together with their coordinates and aspect information using a rate-efficient predictive scheme. In typical cockpit screen videos, we observed that the probability distributions of the characters coordinate differences are very spiky. So, our scheme exploits the regularity of each set of characters by differentially encoding their horizontal and vertical coordinates as follows. First, because characters are aligned in rows, the vertical coordinate of two successive characters are either identical or differ by $\pm$ one pixel. Thus, just 4 different symbols are required to signal the differentially encoded vertical coordinate: 3 of them indicate the same coordinate of the previous character accounting for the localization noise, while a fourth symbol will signal that the coordinate will be explicitly signalled. Second, on the horizontal axis, the characters are often shifted with a constant number of pixels corresponding to the character width, c_w (plus some spacing). This allows us to differentially encode this coordinate by using three symbols: one for the default width c_w, a second that allows to account for a one-pixel tolerance (*i.e.*, it encodes c_w+1) and a third that indicates explicit signalling will follow. Our previous experiments showed that our scheme allows to reduce the characters coding rate to less than an average of 10 bits per character for different real airplane cockpit test video sequences.

4.2.4 Residual Video Compression

As a final step, the text is erased from the cockpit video and the resulting *residual video* is compressed. Each pixel in the threshold map is inpainted using a low complexity colour inpainting technique [18]. Inpainting fills each character pixel with the most likely colour according to its neighbourhood. The inpainting result is a *residual video* without computer-generated graphics, and thus with fewer high frequency components to encode. The residual video is then compressed with some standard video codec (see Sect. 4.3.1) and, for example, stored on a removable support. The encoded characters are stored alongside the compressed video. At the receiver side, the inpainted residual video is first recovered. Then, for each frame, we synthesize the characters that were transmitted as side information. The result is a cockpit video sequence where characters are not affected by compression artefacts, thus preserving their readability.

4.3 Experimental Results

In this section we experiment with our cockpit video semantic compression scheme over multiple video sequences and using different codecs and video compression schemes.

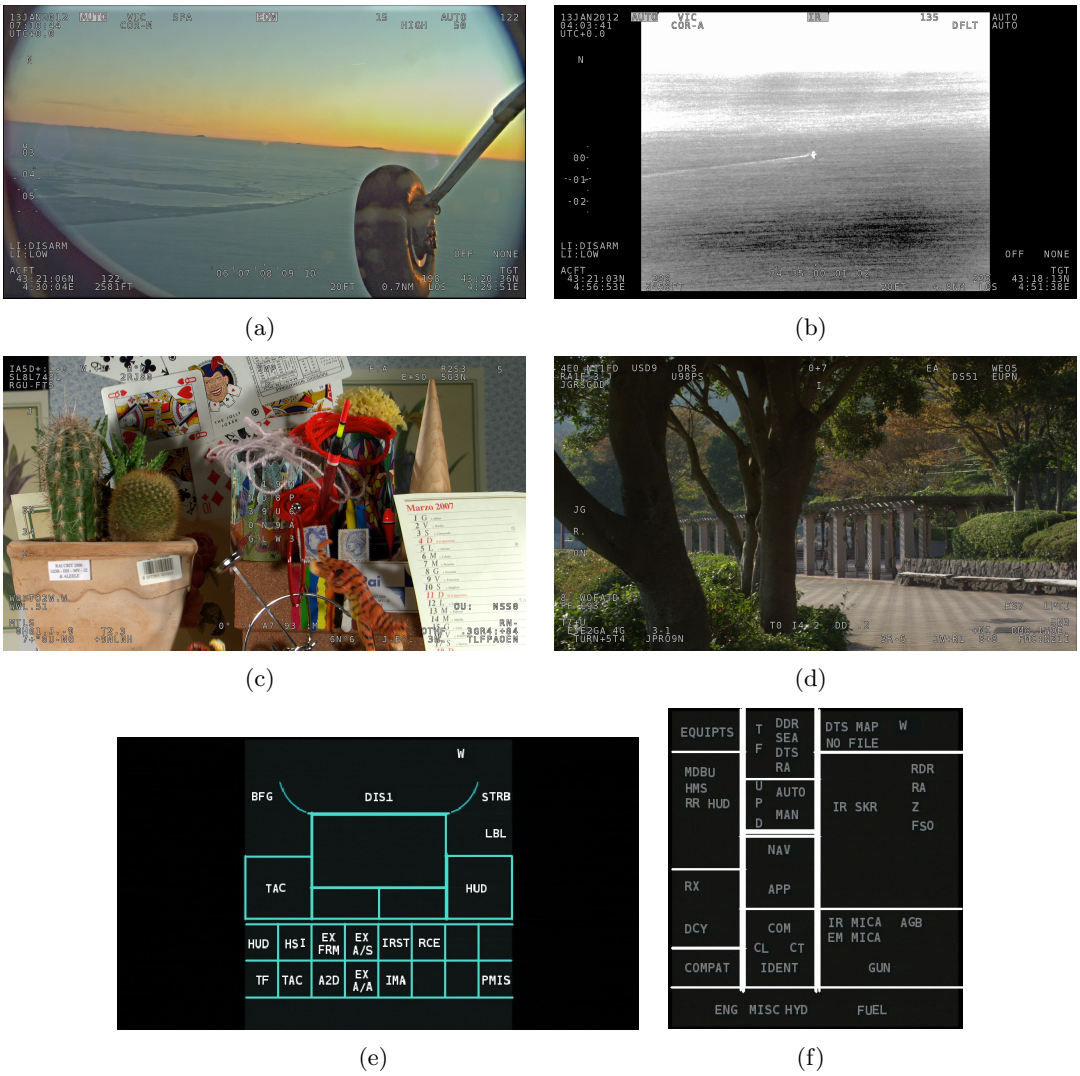


Figure 4.1: The six airplane cockpit screens video sequences used in our experiments (see Tab. 4.1 for the relative characteristics).

4.3.1 Experimental Setup

We experiment over the six cockpit video sequences illustrated in Fig. 4.1. Those sequences can be classified according to the type of the background. Two of them (Seq. 5 and 6) have complex computer-generated text and graphics overlaid on black background. Four of them (Seq. 1, 2, 3 and 4) include text superimposed on natural background, which can be either captured with surveillance cameras installed outside the airplane operating in the visible light (Seq. 1) and infrared spectrum (Seq. 2), or it can be synthetic represented by HEVC/H.265 test sequences *Cactus* (Seq. 3) and *Park* (Seq. 4). The purpose of using the latter two sequences is to stress the resilience of the character detector to background clutter. Table 4.1 summarizes the characteristics of our test sequences. Each sequence is processed according to our semantic compression scheme [2] summarized in the previous section (*Proposed* scheme, in the following).

Concerning the residual video compression in Sec. 4.2.4, we recall that we operate in

Table 4.1: Characteristics of our six test sequences.

# Seq.	Resolution	Frame Rate	Background
1 (Fig. 5.6(a))	1920x1080	24 fps	Natural
2 (Fig. 5.6(b))	1920x1080	24 fps	Natural
3 (Fig. 5.6(c))	1920x1080	24 fps	Natural
4 (Fig. 5.6(d))	1920x1080	24 fps	Natural
5 (Fig. 5.6(e))	720x576	24 fps	Black
6 (Fig. 5.6(f))	720x576	24 fps	Black

an embedded avionics scenario where the overall power consumption must be kept under control. As the codec used to compress the residual video is one of the major computational complexity sources, Tab. 4.2 estimates the complexity of some HEVC/H.265 and AVC/H.264 encoders implemented in FPGA technology. The first two rows compare two implementations of HEVC/H.265 and H.264/AVC encoders in inter-mode, showing that AVC/H.264 power consumption is one third of its HEVC/H.265 counterpart. The last two rows detail two all-intra and inter-enabled low-memory implementations of the AVC/H.264 codec: in this case the H.264/AVC complexity can be less than one fifth the HEVC/H.265 counterpart. Thus, while AVC/H.264 compression efficiency is lower than HEVC/H.265, its lower complexity makes it an interesting option for compressing the residual video in our power-constrained scenario. For this reason, in the following we experiment both with the HEVC/H.265 codec (HM-16.14) plus its Screen Content Coding extension (SCM-8.3) and with the earlier yet less complex AVC/H.264 codec (JM 19.0).

Table 4.2: FPGA complexity estimate for AVC/H.264 and HEVC/H.265 encoders (1920x1080, 30 fps).

Codec	Logic [kALM]	RAM [kbits]	Power [W]
AVC-Inter [4]	30-60	5000	< 1
HEVC-Inter [5]	90	10000	< 3
LowMem AVC-Intra [3]	5	58	0.5
LowMem AVC-Inter [3]	8.6	114	0.6

In the following, we refer to our proposed scheme with HEVC/H.265 and with AVC/H.264 compression of the residual video as *Prop-HEVC* and *Prop-AVC* respectively.
As reference schemes, we consider the case where each sequence is encoded with the standard HEVC/H.265 codec and its SCC extension (*HEVC* and *SCC*, respectively) and AVC/H.264 codec (*AVC*). We recall that with such schemes characters are encoded as pixels exactly as the rest of the screen, so such schemes do allow us to evaluate the effect of compression artefacts on the characters readability.

We evaluate the rate-distortion (called R-D) performances of each scheme at different quantization (QP) values corresponding to high-bitrates (QP from 20 to 45 with steps of 5) and low-bitrates (QP from 45 to 51 with steps of 1) ranges. We objectively evaluate the quality of the recovered video in terms of PSNR, plus we visually inspect the characters in the recovered video to assess how compression artefact impair their readability.

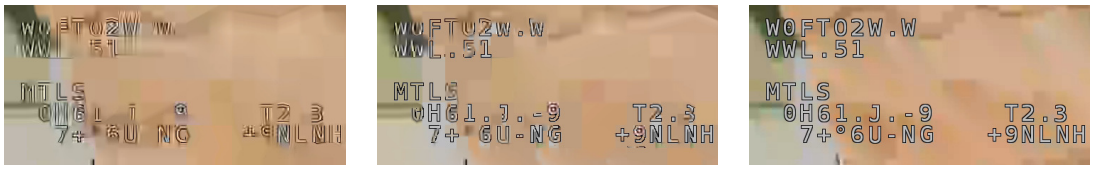


Figure 4.2: Reconstruction artefacts at QP=50 for All-Intra coding configuration. Left: HEVC (PSNR 25 dB); Center: SCC (PSNR 25.6 dB); Right: Prop-AVC (PSNR 26.7 dB). Our proposed scheme preserves the characters readability since they are not affected by compression artefacts, resulting in better PSNR.

4.3.2 Experiments with All-Intra Coding

To start with, we experiment in the case where the video codec operates in Intra-only mode, i.e. no temporal correlation whatsoever is exploited. Despite the reduced compression efficiency, Intra-coding saves on computational complexity by skipping motion search and on memory complexity not having to keep the decoded frames in the reference buffer.

Fig. 4.3 shows the rate-distortion curve for each video sequence and each compression scheme.

Concerning the three reference schemes, HEVC has better R-D performance than AVC by reason of having better coding tools. In turn, SCC outperforms HEVC due to the coding tools tailored for screen compression. SCC performs particularly well especially with the two sequences in Fig. 4.3(e) and 4.3(f) which consists entirely of computer graphics on a black background (and thus the PSNR close to 50 dB). However, Fig. 4.2 (left, centre) shows that neither HEVC nor SCC preserve the readability of the characters at high QP values. The characters are hardly readable because the coarse quantization of the transform coefficients at high QP values makes impossible to correctly recover high frequency primitives such as edges, generating compression artefacts.

Concerning our two proposed schemes, Prop-AVC shows constantly better R-D performance than its AVC counterpart in Fig. 4.3. Most interesting, the experiments reveal that Prop-AVC outperforms HEVC in all natural sequences at medium to low coding rate and outperforms even SCC for low bitrate. Our explanation of these unexpected results is as follows. First, in our proposed scheme, only the inpainted residual video is encoded, so there are no high-frequency components to encode, making it more suitable to be coded with AVC. Therefore, the coding rate of the video is significantly reduced due to the fewer high-frequency elements to encode. Second, at the decoder, the characters are synthesized and overlaid on the decoded residual video and the original-decoded PSNR is computed. So, our proposed method achieves far better PSNR in character areas than the reference schemes and thus overall. In the same time, HEVC pays the cost in rate and distortion of preserving the shapes of computer-generated text. Fig. 4.2 (right) confirms that Prop-AVC preserves the readability of the text, since the synthesized characters are not affected by compression artefacts. That is, these experiments reveal that proposed semantic compression scheme allows the AVC codec to outperform the more recent and more complex HEVC codec both in terms of background video quality and characters readability, whereas it performs close to its SCC extension. Finally, the Prop-HEVC curves show that our method allows plain HEVC to outperforms even its specialized SCC extension, as already verified in our previous research.

Concerning the different classes of video sequences, our proposed method achieves the best results over the sequences 5 and 6 (Prop-AVC largely outperforms even SCC). These

sequences contain only computer-generated graphics over black background, thus the residual video to encode is almost all black, explaining the large gains offered by our scheme. Sequences 1 and 2 obtain consistent gains, however these gains are lower (albeit in excess of 5 dB at most) since their natural background with moderate motion is more complex to encode than a black background. Finally, sequences 3 and 4 show the least improvements due to the high amount of motion details in the background video, which makes the savings in high frequency elements encoding less relevant with respect to the cost of encoding the background in Intra mode.
Notice that for the the Prop-AVC and Prop-HEVC schemes, the rate includes also the rate of the encoded characters. Such rate is equal to about 1.7 kbit per frame, i.e. about 41 kbit/s (for natural screens), which represents a negligible fraction of the overall video rate.

4.3.3 Experiments with Inter-frame Coding

Next, we repeat our experiments allowing the AVC and the HEVC video codecs to exploit the temporal correlation among adjacent frames in the (residual) video. In order to keep the encoder computational complexity low, we use a low-delay configuration, with a Group of Pictures (GOPs) of 4 frames. Also, the encoder is allowed to keep just one frame in the decoded picture buffer to keep low the memory complexity.

Fig. 4.4 shows the corresponding rate-distortion curves. First and unsurprising, exploiting temporal correlation largely improves the rate-distortion performance of all schemes when compared with the corresponding graphs in Fig. 4.3. The better temporal prediction tools of HEVC clearly enable better R-D performance than AVC. However, when we compare the Prop-AVC and Prop-HEVC schemes with the corresponding references, we observe different trends with respect to the Intra-only case. In particular, we observe that for sequences 5 and 6 (Fig. 4.4(e) and 4.4(f)), Prop-AVC outperforms HEVC only for a subset of the tested QP values. By comparison, in Intra-only mode Prop-AVC outperformed even SCC at any bitrate. Also, Prop-AVC cannot achieve the low bitrates achieved by HEVC any more. In any case, SCC outperforms both Prop-AVC and Prop-HEVC. Looking at sequences 1 and 2, Prop-AVC does not outperform HEVC but for very few QP values and only for the second sequence.
We explain these results as follows. The video bitrate, in our scheme, accounts both for the residual video coding rate and for the encoded characters rate. The characters rate is identical to the Intra-only experiments since no temporal correlation is exploited in our character encoding scheme. As the residual video is encoded exploiting temporal correlation, the ratio between characters rate and residual video rate increases. In this scenario, the characters rate is not negligible any more with respect to the residual video rate, explaining the less competitive performance of our scheme when temporal prediction is enabled. These results call for exploiting the temporal redundancy between co-located text in temporally adjacent video frames. Our analysis of real cockpit video sequences verified the intuition that text co-located characters in neighbour frames are strongly correlated. We postulate that by extending our semantic encoding scheme to exploit temporal redundancy, we could drastically reduce the character rate, making it negligible with respect to inter-predicted residual video.

Finally, we also evaluate the performance of our Proposed-SCC scheme with respect to the SCC scheme for the five class-A sequences using the Bjontegaard metrics. Tab. 4.3 shows that our proposed method achieves consistent R-D gains with respect to the reference when the Inter prediction is enabled. Depending on the considered QP ranges,

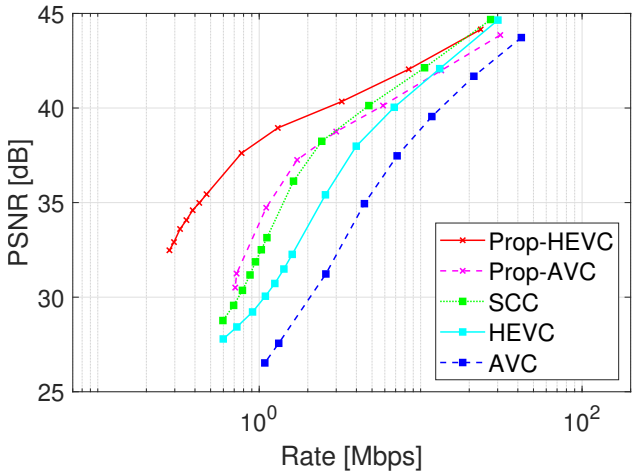
our gains range from 18 % for low QPs values to 57 % for high QP values.

Table 4.3: Bjontegaard metrics (BD-PSNR and BD-RATE) for class-A sequence, computed for Proposed-SCC vs. SCC codecs using Inter coding configuration.

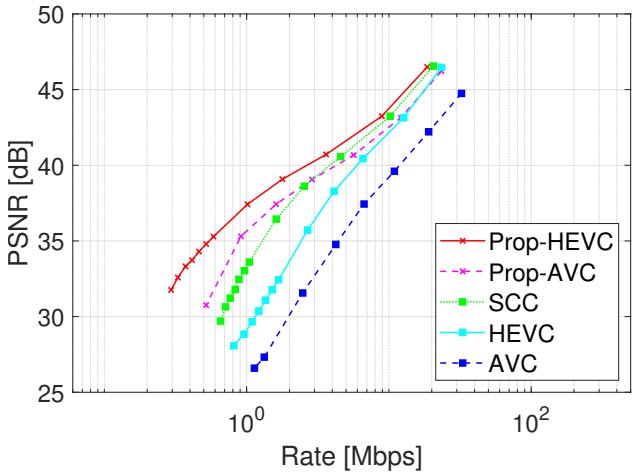
	High Quality $QP = [40, 35, 30, 25]$		Low Quality $QP = [50, 45, 40, 35]$	
Video	BD−PSNR	BD−Rate	BD−PSNR	BD−Rate
Seq. 1	0.69 dB	−34.3%	3.05 dB	−40.0%
Seq. 2	0.43 dB	−17.9%	1.49 dB	−30.0%
Seq. 3	1.66 dB	−38.2%	3.66 dB	−57.1%
Seq. 4	1.31 dB	−31.2%	2.60 dB	−55.5%

4.4 Conclusions

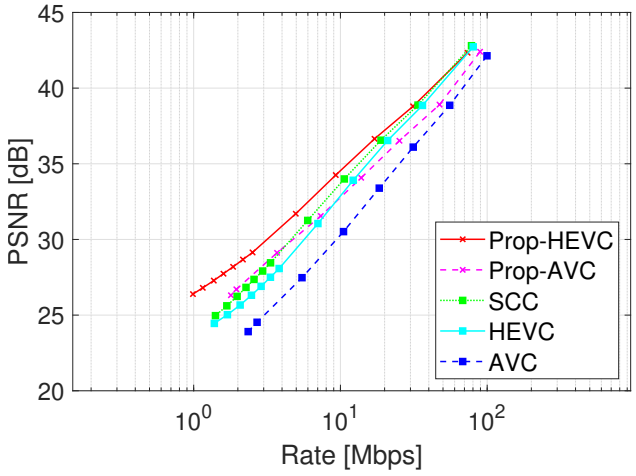
In this chapter, we experimentally evaluated our semantic scheme for airplane cockpit video compression with two different video codecs. Availing different coding configurations, several interesting results emerge.
When the video is encoded in Intra-only mode, the AVC codec retrofitted with our proposed scheme outperforms the more recent HEVC codec and performs close to its SCC extension. Thus, we can exploit the modular structure of the semantic coding scheme to improve the coding efficiency keeping the complexity suitable for avionic applications.
Contrary, when temporal inter-frame prediction is used, the competitive advantage of our semantic video compression scheme decreases as the rate of the encoded characters is not negligible any more with respect to the rate of the residual video.


45
40
35
30
25
PSNR [dB]
10^0
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

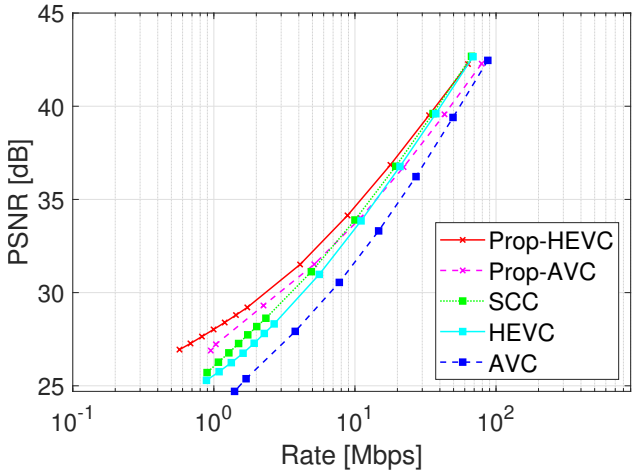
(a)


50
45
40
35
30
25
PSNR [dB]
10^0
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

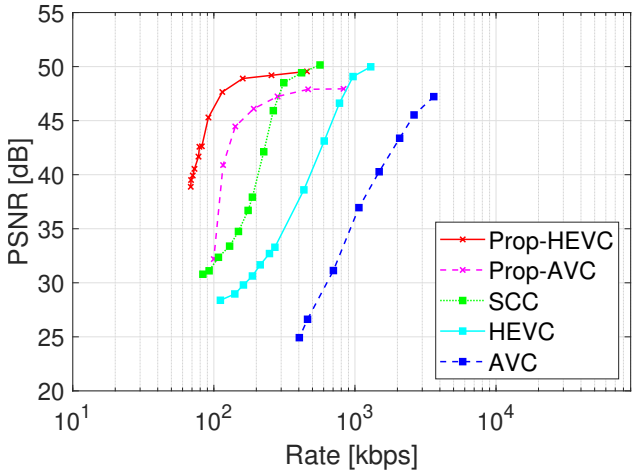
(b)


45
40
35
30
25
20
PSNR [dB]
10^0
10^1
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

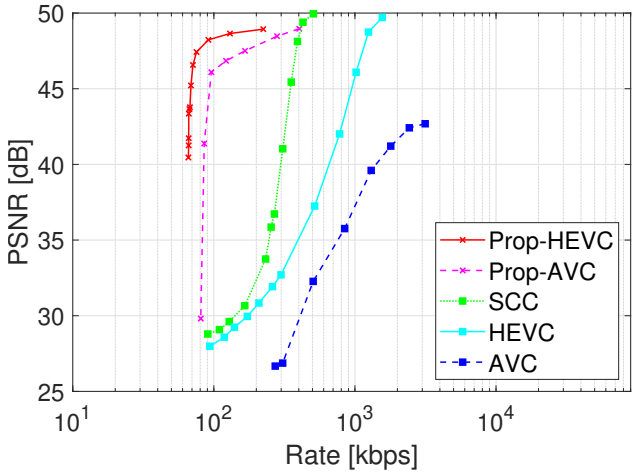
(c)



(d)

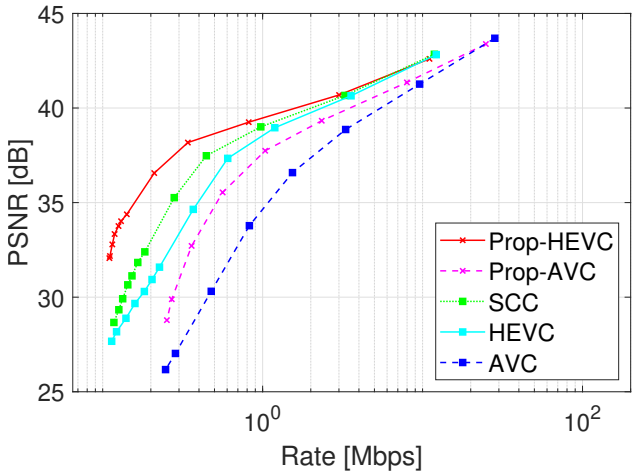


(e)

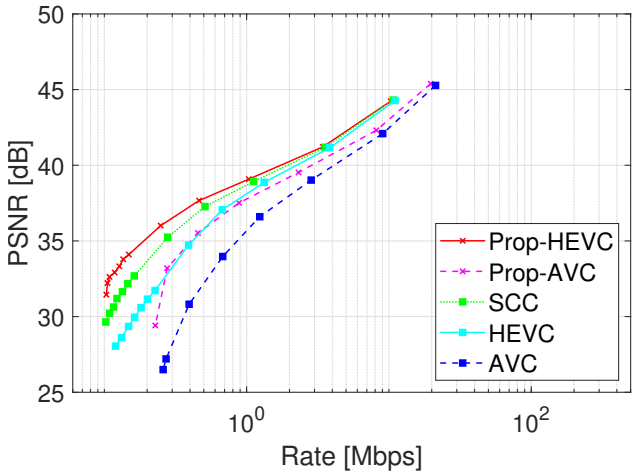


(f)

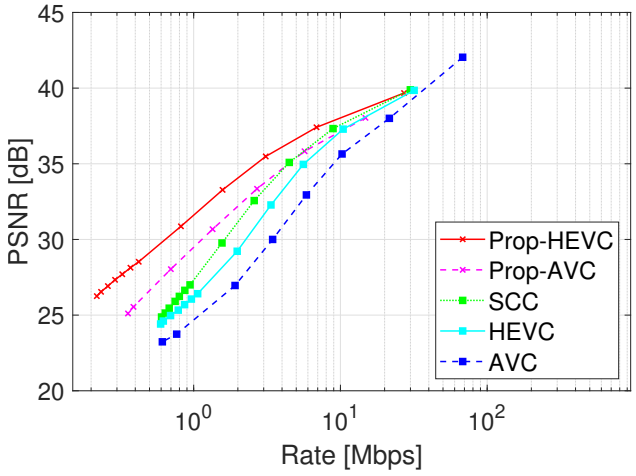
Figure 4.3: PSNR vs. video bitrate for All-Intra coding configuration. First row: Seq. 1 (Fig. 5.6(a)); Second row: Seq. 2 (Fig. 5.6(b)); Third row: Seq. 3 (Fig. 5.6(c)); Fourth row: Seq. 4 (Fig. 5.6(d)); Fifth row: Seq. 5 (Fig. 5.6(e)); Sixth row: Seq. 6 (Fig. 5.6(f)).


45
40
35
30
25
PSNR [dB]
10^0
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

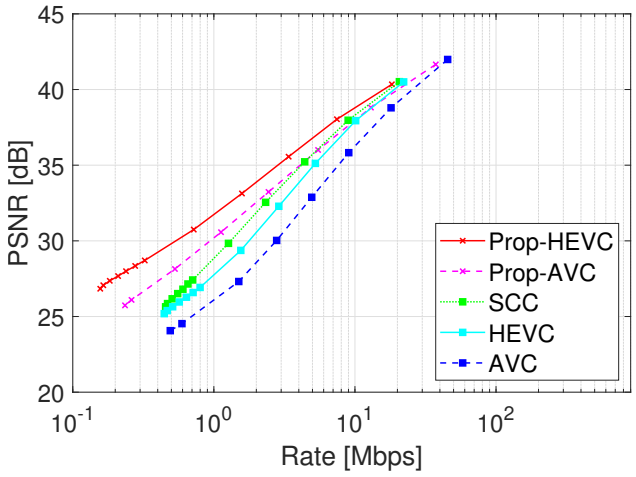
(a)


50
45
40
35
30
25
PSNR [dB]
10^0
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

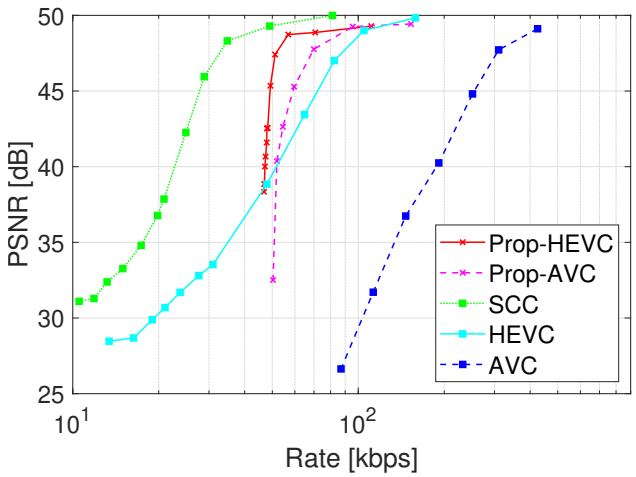
(b)


45
40
35
30
25
20
PSNR [dB]
10^0
10^1
10^2
Rate [Mbps]
Prop-HEVC
Prop-AVC
SCC
HEVC
AVC

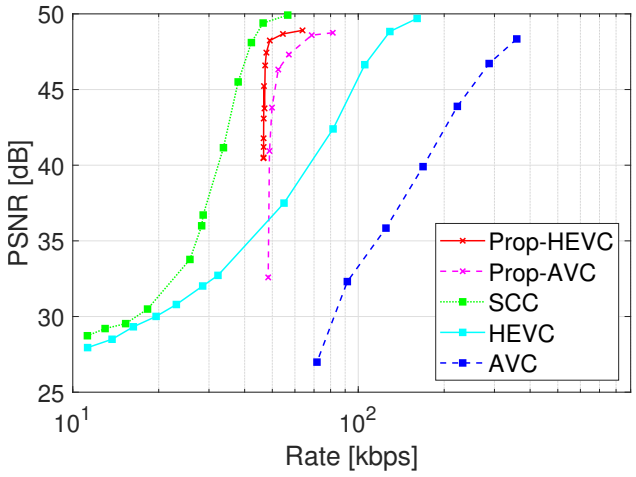
(c)



(d)



(e)



(f)

Figure 4.4: PSNR vs. video bitrate for Inter coding configuration. First row: Seq. 1 (Fig. 5.6(a)); Second row: Seq. 2 (Fig. 5.6(b)); Third row - Left: Seq. 3 (Fig. 5.6(c)); Fourth row: Seq. 4 (Fig. 5.6(d)); Fifth row: Seq. 5 (Fig. 5.6(e)); Sixth row: Seq. 6 (Fig. 5.6(f)).

Chapter 5

Classification and Compression using Temporal Correlation

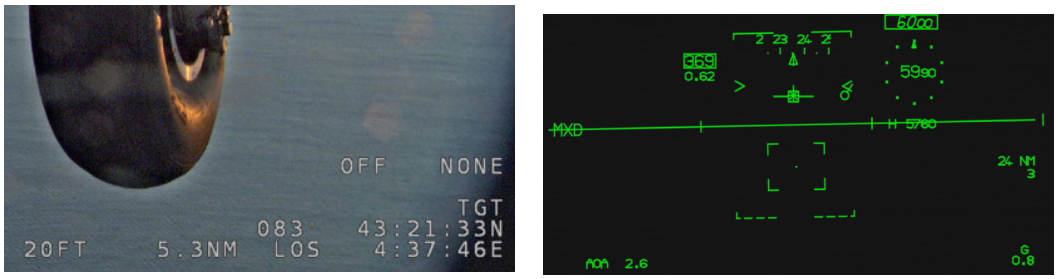
This chapter address the problem of encoding the video generated by the screen of an airplane cockpit considering temporal redundancy both for improving the coding efficiency and for recovering wrongly classified characters occluded by other moving graphical elements, improving the proposed coding scheme. We achieved significant bitrate savings with respect to the references and quasi-errorless character recognition under each screen class prior characteristics.

This chapter is organised as follows. We will introduce the proposed methods in Sec. 5.2. We will experiment with several cockpit computer generated screens and we will discuss the achieved results in Sec. 5.3. The Sec. 5.4 will draw the conclusions of the chapter.

5.1 Introduction

In applications such as training, mission control, *etc.*, it is key to record the cockpit video aboard the aircraft preserving the fine details of text and graphical objects (TGOs, in the following). Due to the characteristic of cockpit images (sharp color transitions at the boundaries between TGOs and background yielding high frequency components) as in Fig. 5.1, coding tools specific for screen contents were standardized [15]. The tension between the requirements and constraints in airplane cockpit video compression calls for ad-hoc video coding schemes.
An emerging requirement for civil air transport is realtime remote tracking of plane position and vital data. The Aircraft Communications Addressing and Reporting System (ACARS) protocol allows a plane to constantly communicate from all over the globe over a combination of 2.4 kbps VHF links with ground stations and a 0.6 to 10.5 kbps satellite links. Other technologies based internet are able to provide higher data links, achieving 30 to 500 times greater than ACARS, being available for other services [140, 141]. Such tight bit budgets to deliver realtime vital plane information motivates the quest of bandwidth-efficient data coding techniques tailored to avionic requirements.

In our previous chapters performance was consistently better than the state of the art, yet we ran into two major limitations: i) our "intra" character lossless coding hinders performance at low bit-rate ii) in some cases character recognition fails, in particular when characters are occluded by other objects.



(a) (b)

Figure 5.1: 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.

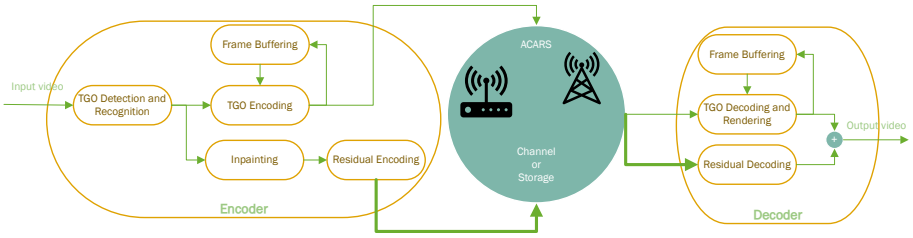


Figure 5.2: Architecture of our proposed scheme for cockpit video compression. On the left, the encoder yields a residual video without TGOs and a side stream with the TGOs encoded according to their semantics. At the receiver side, the decoder superimposes newly synthesized TGOs over the residual video, recovering the original frame.

In this chapter, we leverage the correlation among temporally adjacent frames to address both efficient video coding and robust character recognition. As for the first issue, we tailor a specific method that takes into account the characteristics of the cockpit screen videos. As for the second, we state the problem as a MAP classification (namely maximum a posteriori probability estimate) given the present and the past, in order to take into account the temporal redundancy. Our experiments with real airplane cockpit video sequences show that our semantic coding scheme achieves bitrate savings up to 62 % for some H.265/HEVC-SCC [16, 15] encoded sequences. Also, an H.264/AVC [142] encoder retrofitted with our semantic coding scheme achieves competitive compression efficiency to H.265/HEVC-SCC and H.266/VVC [143] at a fraction of the encoder complexity. As for text recognition, our MAP-based 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.

5.2 Proposed Architecture and Contributions

We first recall the general architecture of the semantic encoder used in [2], then we detail the key novelty elements of this work, i.e.:

- A method for improving text recognition accuracy by taking into account specific characteristics of airplane cockpit videos

- A method for lossless text coding exploiting temporal redundancy and dealing with static (fixed-position) or moving strings

To sum up, the proposed contributions significantly improve the performance of the proposed architecture, as experimentally demonstrated in the next section.

5.2.1 Semantic Encoder Architecture

Fig. 5.2 illustrates the architecture of our semantic encoder. In the following we recapitulate the main functional blocks.

The architecture of the encoder (left part of the figure 5.2) is as follows. The first step consists in detecting and recognizing TGOs in the current frame. Detection is made easier by the fact that some TGOs must appear in predefined locations to allow the pilot to spot them with ease. Recognition is addressed via a LeNet5-based CNN that in our experiments did strike a favorable tradeoff between character recognition accuracy under some conditions and computational complexity. Second, TGOs are encoded in the semantic domain with suitable lossless methods rather than in the pixel domain. Third, residual images are generated by removing pixels belonging to TGOs with a suitable inpainting method. The residual image lacks the high frequency components corresponding to the inpainted TGOs. Fourth, residual images are encoded with an off-the-shelf video encoder such as H.264/AVC, H.265/HEVC (possibly with its SCC extension) or H.266/VVC, depending on the computational complexity 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 figure 5.2) 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 the above architecture, this work introduces robust character recognition and predictive TGO compression at the encoder side, as we detail below.

5.2.2 Robust Character Recognition

As shown in Fig. 5.1(b), characters in cockpit screens may be occluded by graphical objects in foreground and background clutter. As shown in the experimental part, the LeNet5 convolutional neural network (CNN) [144] that performed very well when the characters are not occluded [2], has some performance degradation when indeed they are. Our approach towards robust character recognition rests on two complementary legs that are described in the following.

5.2.2.1 Training on synthetic Occluded Samples

In our previous endeavours, 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. 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

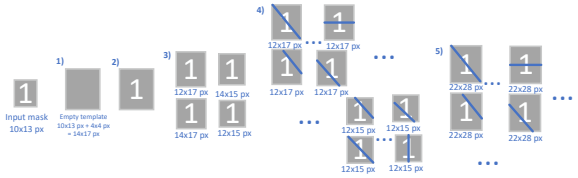


Figure 5.3: The procedure to generate synthetic training samples with occlusions for character recognition.

any sample size. Fig. 5.3 illustrates the whole procedure. First, we randomly draw one character or symbol among those that shall be recognized. The character is then rendered within a 10x13 frame with alpha channel over 256 levels of pixel intensity with color, font face and size depending on the specific situation 1). Next, the frame is superimposed over a 22×28 empty canvas at a randomly drawn position, to enable robustness against loose character detection 2). 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 allows for generating a large dataset with controlled occlusions. In our experiments, we draw a dataset of 5000 samples that is divided in 80% training and 20% testing. In the experimental section, we show the benefits in terms of robustness of a LeNet5 trained with the above approach.

5.2.2.2 Exploiting temporal redundancy

In cockpit screens, digits on screen must change slowly enough for the pilot read them. So, characters as the digits and letters on virtual instruments can be statistically modeled. For example, if a digit is equal to m at time $t-1$, at time t it will have a relatively large probability to keep the same value or to switch to $m \pm 1 \mod 10$, while the probability to switch to any other value is smaller. This model could be further refined considering neighboring digits.

We propose now a method to leverage the temporal statistical dependence among symbols to improve the character recognition accuracy with negligible complexity increase. The method is inspired from [59, 60], whose the authors correct the *a posteriori* probabilities of the classes, exploiting some updated *a priori* probabilities of the data. In our case, the updated probabilities are based on the temporal dependency model $P_{Y_t|Y_{t-1}}(y_t|y_{t-1})$, which we suppose is known. In the following we will omit the subscripts when this does not create ambiguity in notation. We refer to the random vector representing the pixel patch of the current character as X_t, to the random vector of the current output as Y_t and to the random vector of the past output as Y_{t-1}. The y_t is the current label, y_{t-1} is the past label and x_t is the current input image. We call M as the number of the characters as the CNN output. Thus, the baseline CNN is supposed to provide the conditional probability $P(y_t|x_t)$, and to infer from it the MAP estimator of the character value:

$$\bar{y}_t = \arg\max_{y_t} P(y_t|x_t). \tag{5.1}$$

On the other hand, we would like to find the MAP estimator also consider the past value y_{t-1} :

$$\widehat{y}_n = \arg\max_{y_t} P(y_t|x_t, y_{t-1}). \tag{5.2}$$

The key point of the following is that, under some very reasonable hypothesis, we can compute $\widehat{y}_t$ without having to modify or even re-train the CNN providing $P(y_t|x_t)$.

We assume an hypothesis of *conditional independence*: given the value of the character at time t, Y_t, the patch X_t is independent from the value at 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 $Y_{t-1} \to Y_t \to X_t$). The past value Y_{t-1} only influences the current patch through Y_t, but once given the latter, the former has not statistical influence on X_t. This hypothesis seems very reasonable and in this case, we can develop the conditional probability in Eq. 5.2 as follows:

$$\begin{aligned} P(y_t|x_t, y_{t-1}) &= P(x_t|y_t, y_{t-1})\frac{P(y_t, y_{t-1})}{P(x_t, y_{t-1})} \\ &= P(x_t|y_t)\frac{P(y_t, y_{t-1})}{P(x_t, y_{t-1})} \qquad (5.3) \\ &= P(x_t|y_t)\frac{P(y_t|y_{t-1})P(y_{t-1})}{P(x_t, y_{t-1})} \\ &= \frac{P(x_t|y_t)}{P(x_t|y_{t-1})}P(y_t|y_{t-1}) \qquad (5.4) \end{aligned}$$

where in Eq.(5.3) we used the conditional independence hypothesis. Now, using the Bayes' rule,

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

and Eq. 5.4 becomes:

$$\begin{aligned} P(y_t|x_t, y_{t-1}) &= \frac{P(y_t|x_t)}{P(y_{t-1}|x_t)}\frac{P(y_{t-1})}{P(y_t)}P(y_t|y_{t-1}) \\ &= \frac{P(y_t|x_t)P(y_{t-1}|y_t)}{P(y_{t-1}|x_t)} \qquad (5.5) \end{aligned}$$

Finally, our target MAP estimator can be written as

$$\widehat{y}_n = \arg\max_{y_t} P(y_t|x_t)P(y_{t-1}|y_t) \qquad (5.6)$$

where we can neglect denominator of Eq. (5.5) 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 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 observe that in Eq. (5.6) we should use the *backward* conditional probability $P_{Y_{t-1}|Y_t}(y_{t-1}|y_t)$, and this is very easy if, besides the *forward* probability model $P_{Y_t|Y_{t-1}}$ (which is known by hypothesis), also the marginals

P_{Y_t} are known. All in all, very often it could be reasonable to assume that the backward conditional probability is equal to the forward probability.

We also observe that, in order to compute the *weights* $P(y_{t-1}|y_t)$ we are supposed to use the true value y_{t-1} of the character at time $t-1$. In practice, we never have access to this information, but only to $\widehat{y}_{t-1}$. We then propose to use $\widehat{y}_{t-1}$ in place of y_{t-1} for computing the weights. This of course can lead to error propagation in case of bad estimation of y_{t-1} However, we observe that in absence of occlusions, the error rate probability is extremely small [2]. Therefore, there are practical ways to reduce error propagation: weights could be used only when confidence on y_{t-1} is high. In our experiments, we observed that the proposed method brings very significant improvement in character recognition even in the case where we do not resort to specific protection against error propagation.

Finally, concerning the complexity, we observe that all the weights can be computed off line for a given cockpit screen. Therefore, the additional complexity with respect to the baseline classifier, is equal to N_C multiplications per character, where N_C is the size of the character alphabet. In the experimental section we will show some simple yet effective models for the conditional probabilities $P(y_{t-1}|y_t)$.

5.2.2.3 Models for Character Conditional Probabilities

As told in Sec. 5.2.2, in order to take into account temporal dependence in character recognition, we need a model for 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 fairly well in many practical situations. We discriminate between two cases: if y_{t-1} is a letter or a digit. In the first case, we assume

$$P(y_t|y_{t-1}) = \begin{cases} p & \text{if } y_t = y_{t-1} \\ \frac{1-p}{M-1} & \text{otherwise} \end{cases} \tag{5.7}$$

that is, we assume a probability p that the symbol does not change; if the symbol changes, it can switch with uniform probability to any of the remaining $M-1$ characters. The probability p can be estimated by observing how frequently a letter changes in some training data. In principle p should depend on t, but in our experiments we use a constant value.

In the case of digit, we propose the following model:

$$P(y_t|y_{t-1}) = \begin{cases} p & \text{if } y_t = y_{t-1} \\ q & \text{if } y_t = y_{t-1} \pm 1 \mod 10 \\ \frac{1-p-2q}{M-3} & \text{otherwise} \end{cases} \tag{5.8}$$

where our parameters are the probability p that the digit does not change and the probability q to switch to next or previous digit; if neither case happens, we assume a uniform probability to switch to any of the remaining $M-3$ characters. As in the previous case, we assume that we can estimate p and q from some training data. As for the case of letters, the distribution parameters p and q could depend on time. It is not difficult to propose some simple adaptive estimation of them, but we used constant values and we achieved very good results.

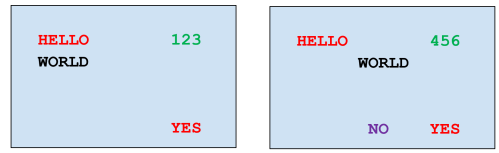


Figure 5.4: Toy example of temporally consecutive cockpit screen frames: frame at time t is predicted from frame at time $t-1$.

5.2.3 Efficient Character Coding using Temporal Redundancy

In this section we describe a method to encode characters in a frame exploiting the temporal redundancy with respect the previous frame(s). In order to describe the proposed approach we start with an example. Fig. 5.4 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.

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 Tab. 5.1. As one can remark, the number of recognized characters can be different from one image to the other. However, often in cockpit screens a strong correlation between the content of adjacent frames exists. For example, a character or a small string of characters often retains both position and value across multiple neighbor frames (red text in Fig. 5.4) In other cases, a character may change value and retain position, as the text in green (this happens for example for 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 a reference one. 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 or tuples in the two tables as N_{CC} for the current table and N_{CR} for the reference.

The proposed iterative algorithm (see Fig. 5.5) 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
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 (set by visually analyzing the distance of moving characters in several cockpit classes)

Table 5.1: Reference table, current table and corresponding encoding primitives for the example in Fig. 5.4

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")`

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 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 algorithm goes back to step 1.

For the fourth case, we have to 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

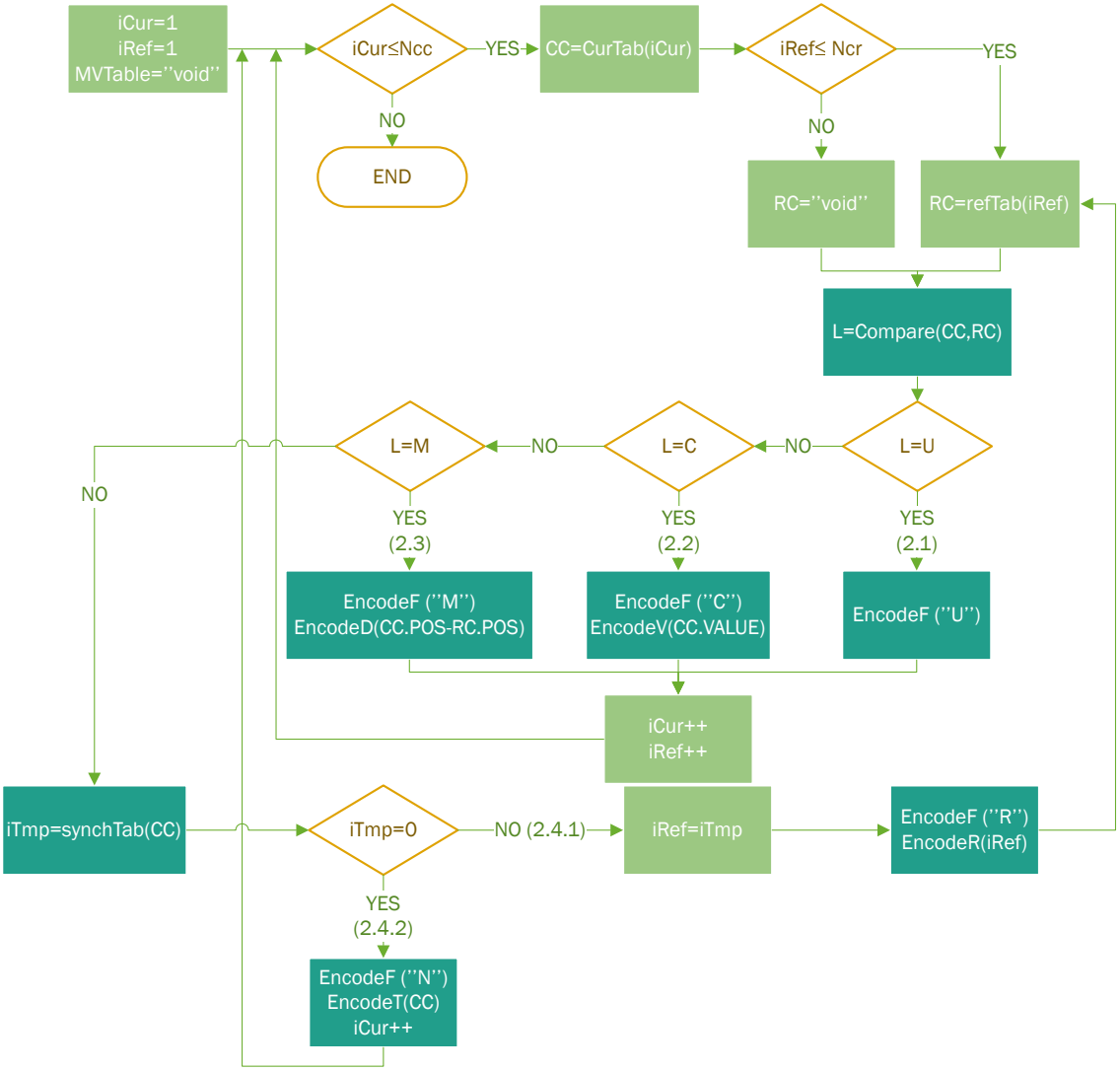


Figure 5.5: Iterative algorithm for characters coding.

cases happens, it means that the two indexes must be realigned, so we set `IRef`=`ITmp`, we 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 Tab. 5.1 the primitives to be used in order encode the current character table. Each primitive must be able to 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 Huffman code for the flags (see Tab. 5.2), but more sophisticated method such as context-based, adaptive arithmetic encoder 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 techniques for that problem could be

Table 5.2: The `EncodeF()` primitive behavior

Input Flag	Output Binary code
`U`	`0`
`C`	`100`
`D`	`101`
`N`	`110`
`R`	`1110`
`END`	`1111`

used, from a simple fixed-length code to more sophisticated arithmetic or Exp-Golomb techniques. 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 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.

We remark the the `EncodeT()`primitive is the less effective, because in this case we cannot take advantage from temporal redundancy. However, this is only used for characters tagged as "new". Very often, a tuple in the current table will be encoded with only a few bits when the flag "U", "C" or "M" are selected.

The decoder behavior is very straightforward, since it has to decode the flags and update the table according to them. The decoder has to keep 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 method for character encoding. Likewise Inter coding of images, we can easily extend it to the case of multiple references: in that case, we have to keep multiple reference indexes, and each time we access a reference table, we have also to specify which one it is by encoding a suitable label in the stream. Finally, we also remark that we need an "Intra" method in order to start the encoding: even though the proposed method would work (it suffices to use a void table as reference: for the current Intra table all the characters are encoded as "new"), it is more efficient to use the encoding technique proposed in [2].

5.3 Experiments

In this section we provide a comprehensive set of experiments in order to validate the proposed method. 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 encoder configurations.

5.3.1 Experimental Setup

We experiment over the ten cockpit video sequences illustrated in Fig. 5.6. Four of them (Seq. 1, 2, 3, 4) have text superimposed to a background that has been captured with

cameras installed outside the airplane operating either in the visible spectrum (Seq. 1) or in the infrared (Seq. 2); sequences 3 and 4 have been generated by adding some text to MPEG test sequences *Cactus* (Seq. 3) and *Park* (Seq. 4) such that they are similar to actual cockpit videos 1 and 2. As in our previous work [2], these natural-background sequences are referred to as "Cockpit HD" sequences. 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 (namely BQTerrace from the MPEG test set) with computer generated complex graphics as often can be found in videos mixing head-up displays with natural content ("Cockpit HUD").

The TGOs in 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. In particular, digits can switch to the next or previous value (modulo 10). Concerning the graphic objects, we generate lines and circles for sequences 8-10. In the left part of the screen, we have 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 5.3 summarizes the characteristics of our test sequences. They all have a frame rate of 24 fps. Please note that our definition of sequence classes is different from the classes used in MPEG standardization.

Table 5.3: Characteristics of the test sequences.

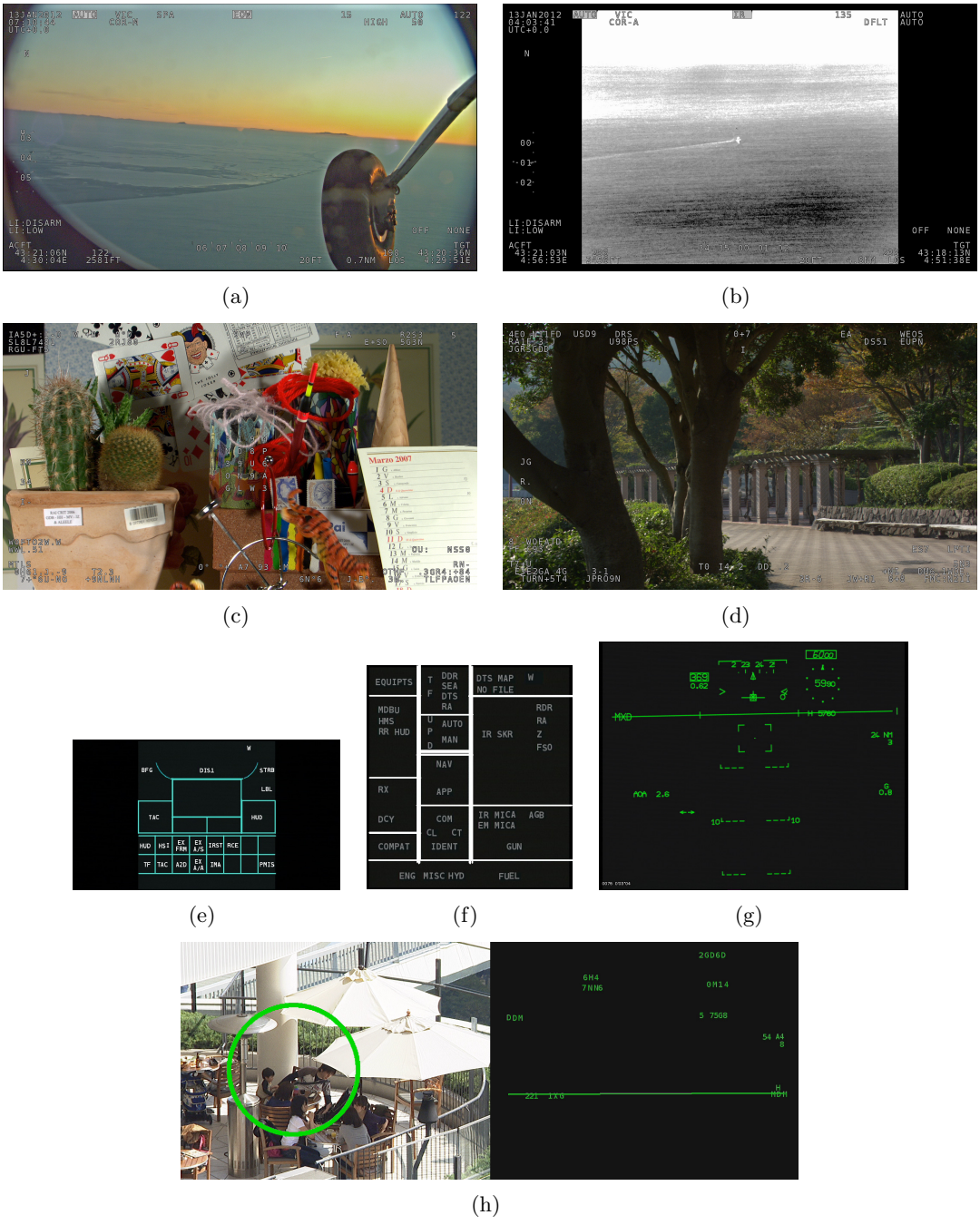
# Seq.	Resolution	Class	Source
1 Fig. 5.6(a)	1920x1080	Cockpit HD	Actual footage
2 Fig. 5.6(b)	1920x1080	Cockpit HD	Actual footage
3 Fig. 5.6(c)	1920x1080	Cockpit HD	Synthetic
4 Fig. 5.6(d)	1920x1080	Cockpit HD	Synthetic
5 Fig. 5.6(e)	720x576	Cockpit SD	Actual footage
6 Fig. 5.6(f)	720x576	Cockpit SD	Actual footage
7 Fig. 5.6(g)	720x576	Cockpit SD	Actual footage
8 Fig. 5.6(h)	1440x576	Cockpit HUD	Synthetic, mild occlusions
9 Fig. 5.6(h)	1440x576	Cockpit HUD	Synthetic, some occlusions
10 Fig. 5.6(h)	1440x576	Cockpit HUD	Synthetic, high occlusions

5.3.2 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 CNN in [2] recognizes all characters anyway without a flaw.

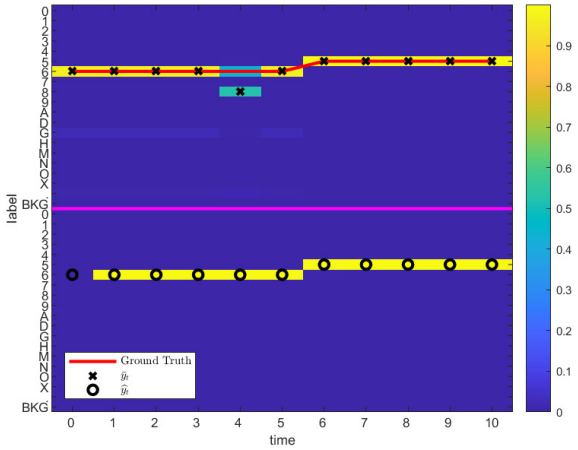
The first row of Tab. 5.4 is relative to the case where the CNN is trained without occlusions and class scores are not weighted according to the conditional probabilities, as in [2] or [145]. The average character recognition error rate is 3.15 %, topping nearly 5% for sequence 10, where 2 moving lines occlude on screen characters.

The second row of the table refers to the case where the CNN is trained over occluded samples as in Sec. 5.2.2, however the class scores are not weighted according to the conditional probabilities as in as in Eq. (5.6). Training the network over occluded samples reduces the rate of errors to 2.42 %, however most heavily occluded characters may still



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5.6: The ten airplane cockpit screens video sequences used in our experiments (see Tab. 5.3 for the relative characteristics).



(a) Digit "6" occluded by the line in Seq. 9.

Figure 5.7: Estimation of the class probability distribution $P(y_t|x_t)$ by the NN (top) and of $P(y_t|x_t)P(y_{t-1}|y_t)$ (bottom). The estimated classes are the $\arg\max_{y_t}$ of those distributions.

be misrecognized.
The third row of the table is for our proposed scheme where the CNN 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 characters, errors are 3 times less frequent and in sequence 10 where 2 lines may occlude characters one third of the errors is corrected. Speaking about sources of errors, notably difficult situations to recover is when an occluding line is static or the line occludes a digit while it has just changed its value.
We recall that our robust character method entails only a negligible complexity increase as weights can be computed offline.

Table 5.4: Character recognition error rate for sequences with occlusions.

	Seq. 7	Seq. 8	Seq. 9	Seq. 10
Baseline [2]	2.34 %	3.08 %	2.31 %	4.87 %
Training with occlusions	1.73 %	2.31 %	1.67 %	3.97 %
Proposed	0.13 %	1.03 %	0.77 %	3.21 %
Error rate variation wrt [2]	-94.4 %	-66.6 %	-66.7 %	-34.1 %

To gain some insight about the error correction capability of the proposed method, we show in Fig. 5.7 the ground truth, the probability distributions $P(y_t|x_t)$ (in the upper part) and $P(y_t|x_t, y_{t-1})$ (in the lower part) provided respectively by the NN and after weighting with the conditional probabilities as in Eq. (5.6). We may observe (in the upper part of the image) that even if the NN trained with occlusions produce an error, we are able to recover it (in the lower part of the image) by considering temporal redundancy.

5.3.3 Character Coding using Temporal Redundancy

We now evaluate the efficiency of the predictive character encoding method proposed in Sect. 5.2.3. 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 [133], Burrows-Wheeler transform (BWT, also block-sorting compression [135]), PPM (Dmitry Shkarin's prediction by partial matching method [136]), LZMA algorithm (Lempel-Ziv-Markov chain algorithm [134]). We also compare our proposed method with the baseline method we previously described in [2], where no temporal redundancy is taken into account. Tab. 5.5 shows the encoding rate in bits per character (*i.e.*, the average number of bits needed to encode a tuple in the character table). We recall that if we encoded character using a simple constant-length code, for example, for Cockpit HD sequences, that would require 29 bits per character, including the horizontal and vertical coordinates (e.g. each on 11 bits), the label (e.g. on 6 bits) and the color (e.g. on 1 bit). The gains obtained with our proposed predictive character encoding scheme are huge. For example, for HD sequences (Seq. 1-4), our predictive coding scheme requires around 30% less rate than our baseline. For HD and SD sequences, the bit-rate saving is as high as 80% with respect to the references.

Moreover, our proposed method allows us to stream the entire set of on screen characters at several frames per seconds over low bandwidth links between plane and the base stations. For example, for cockpit HD screens we can sent over 170 characters at 10 fps, making near 10 kbps. For cockpit SD and HUD screens we can encode over 70 characters obtaining no more than 2 kbps at 10 fps.

Table 5.5: Characters coding rates for different characters coding modes. Measurement unit is the number of bits per character.

	Huffman	BWT	PPM	LZMA	Baseline [2]	Proposed
Seq. 1	27.64	25.33	27.08	18.94	10.24	**5.80**
Seq. 2	27.67	25.40	27.15	18.98	10.35	**6.02**
Seq. 3	27.88	26.69	27.24	19.05	10.37	**8.12**
Seq. 4	27.90	26.71	27.42	19.23	10.48	**8.34**
Seq. 5	8.55	12.39	16.09	7.08	12.35	**0.73**
Seq. 6	8.57	12.41	16.11	7.10	12.47	**0.82**
Seq. 7	9.61	14.31	16.23	8.71	12.24	**1.74**
Seq. 8	9.65	10.3	9.32	9.25	11.73	**1.80**
Seq. 9	9.72	10.41	9.04	8.73	11.56	**2.01**
Seq. 10	10.62	10.97	9.83	9.28	11.64	**2.78**

5.3.4 Video Coding Experiments

Finally, we experiment at encoding the residual video generated by inpainting the detected graphics and we measure the end-to-end video compression efficiency of our scheme. Concerning the coding of the residual video, we consider two alternative implementations. In the *Prop-Intra* implementation, the residual video is encoded with an All-Intra configuration, no matter the specific codec. In the *Prop-Inter* implementation, the residual video is encoded enabling the video codec inter-frame prediction. These two implementations allow different tradeoffs between encoding efficiency on one side, and encoder complexity

and robustness against error propagation on the other.

Concerning the video codec, we experiment with the AVC/H.264 encoder JM 19.0, the H.265/HEVC encoder HM-16.14 with Screen Content Coding extension SCM-8.3 (HEVC-SCC for short) and the H.266/VVC encoder VTM13.2. These three codecs enable different tradeoffs between encoder complexity on one side and complexity on the other. H.266/VVC encoder complexity is currently out of the budget of typical avionic application [143], however we keep it into our experiments as an upper bound reference.

All encoding experiments consider quantization values in the QP range from 20 to 45 with steps of 5, plus we focus on the very low bit-rate rage QP from 45 to 51 with steps of 1. The results are shown in terms of rate-PSNR curves and of Bjontegaard Delta Rate (BD-Rate) [44].

5.3.4.1 Experiments with Prop-Intra

Now we plug the proposed character compression method in the full semantic encoder architecture. We have 3 methods: the proposed technique with HEVC-SCC or AVC (Prop-HEVC and Prop-AVC) compression; and the one reference HEVC-SCC. We start from the case where the residual is encoded in All-Intra configuration, and compare it to reference codecs, which equally are in All-Intra.

Fig. 5.8 shows the rate-distortion curves for each video sequence and residual video compression scheme. We observe that Prop-AVC has consistently better R-D performance than HEVC-SCC at low bitrates in all Cockpit HD and SD sequences. For the more difficult Cockpit HUD sequences, it has not better performances than the other codecs. Prop-HEVC achieves even better performance, having by far the best RD performances for all bit-rates and sequences. These results are expected, since those experiments are very similar to those in [2], except that we improve our characters coding method (namely the characters compression gains are between no less than 40% and around 90% by comparing the last two columns in Tab. 5.5 corresponding to the two types of proposed character coding algorithms).

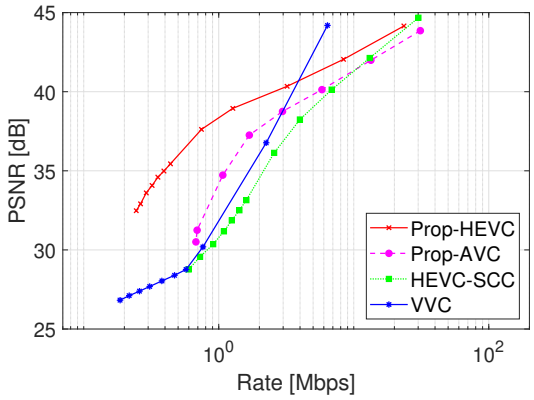
For the sake of completeness we also report BD-Rate and BD-PSNR for Prop-HEVC with respect to HEVC-SCC. In Tab. 5.6 we report the results for Cockpit HD sequences for medium-to-high ($QP = [40, 35, 30, 25]$) and for low-to-medium ($QP = [50, 45, 40, 35]$) rate ranges. The gains are remarkably high, in particular at low rate, up to -70 % in BD-Rate.

For Cockpit SD sequences, as we can see, for example, from the RD curve in Fig. 5.8(e), that in order to achieve a PSNR of about 40dB, SCC needs three times as rate as the Proposed-AVC and HEVC more than five times. At higher PSNRs, gaps are somewhat smaller but still very remarkable.

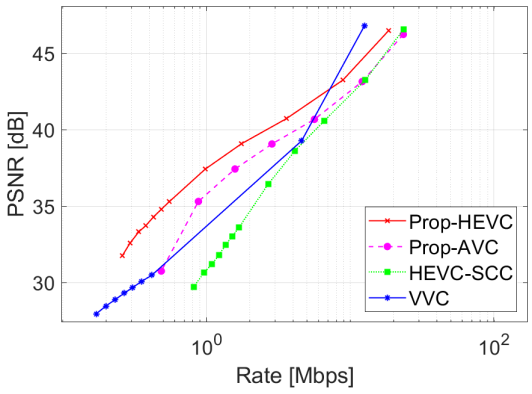
Finally, for Cockpit HUD, we show the results in Tab. 5.6 (the same QP ranges as Tab. 5.6). Again, we have consistent gains in all cases. Moreover, for medium-to-high quality the gains are up to 8% and for low-to-medium quality up to 17% in bitrate reduction.

5.3.4.2 Experiments with Prop-Inter

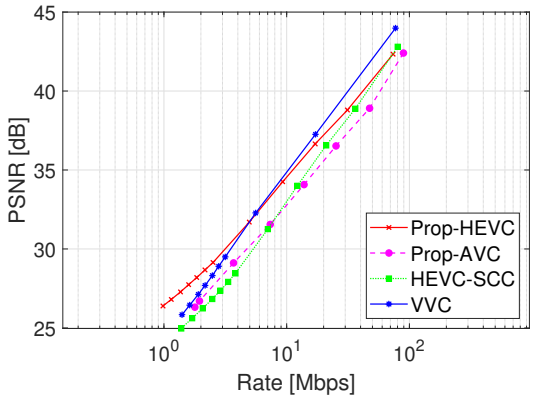
Next, we repeat the previous experiments enabling inter-frame prediction for residual video coding. In order to keep the complexity limited, we use a low-delay configuration with an Intra period of 4 frames. Also, the encoder is allowed to keep just one frame in the decoded picture buffer to reduce the memory footprint.



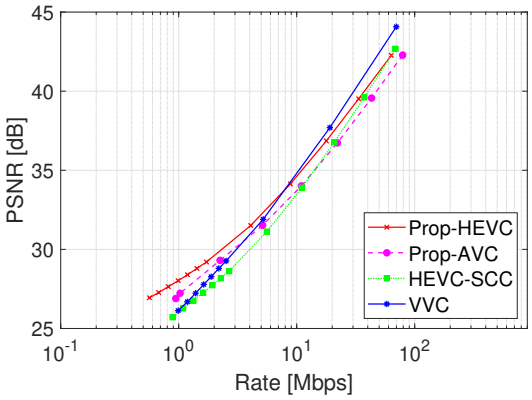
(a) Seq. 1



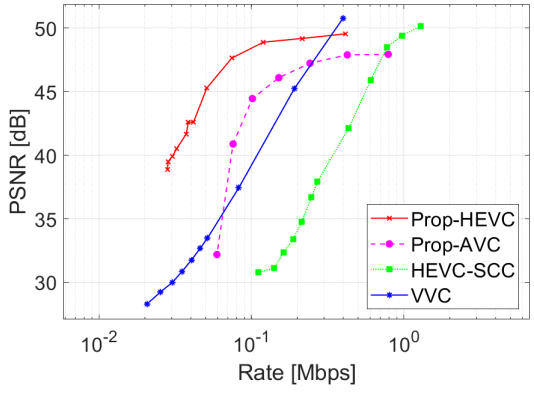
(b) Seq. 2



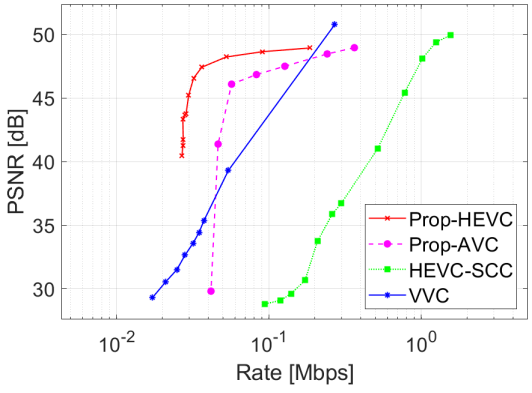
(c) Seq. 3



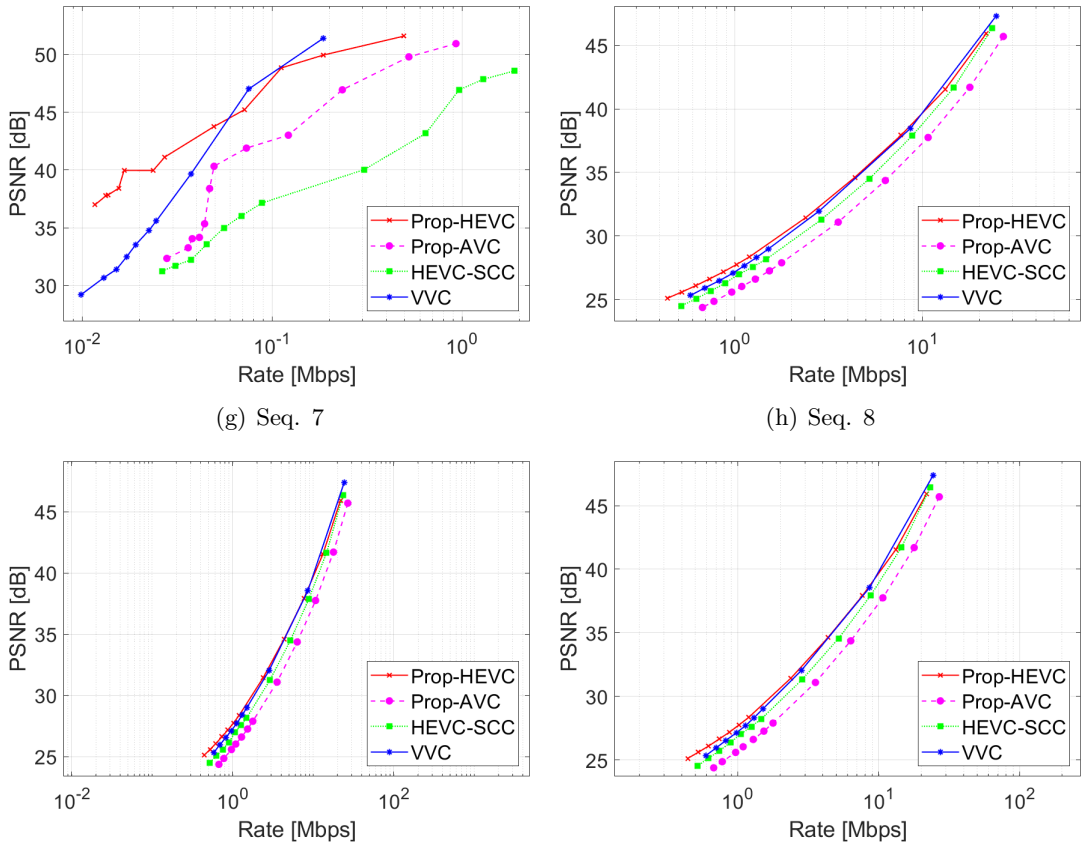
(d) Seq. 4



(e) Seq. 5



(f) Seq. 6



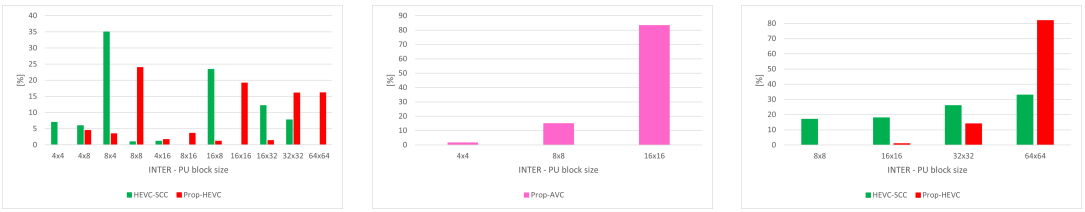
(g) Seq. 7 (h) Seq. 8

(i) Seq. 9 (j) Seq. 10

Figure 5.8: PSNR vs. video bitrate for Prop-Intra coding configuration.

Table 5.6: Bjontegaard metrics for Cockpit HD and Cockpit HUD screens for Prop-Intra coding scheme.

Prop-HEVC.		Medium-to-High $QP = [40, 35, 30, 25]$		Low-to-Medium $QP = [50, 45, 40, 35]$	
vs.	Sequence	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
HEVC-SCC	Seq. 1	1.30 dB	-46.5 %	6.48 dB	-69.88 %
HEVC-SCC	Seq. 2	1.06 dB	-30.28 %	3.82 dB	-57.33 %
HEVC-SCC	Seq. 3	0.67 dB	-15.13 %	1.56 dB	-31.45 %
HEVC-SCC	Seq. 4	0.60 dB	-13.79 %	1.23 dB	-32.99 %
HEVC-SCC	Seq. 8	0.63 dB	-6.2 %	0.86 dB	-17.06 %
HEVC-SCC	Seq. 9	0.50 dB	-7.99 %	0.88 dB	-17.39 %
HEVC-SCC	Seq. 10	0.38 dB	-6.20 %	0.79 dB	-15.73 %



(a) QP 35 (b) QP 35 (c) QP 50

Figure 5.9: Inter PUs sizes distribution for the Seq. 8-9-10.

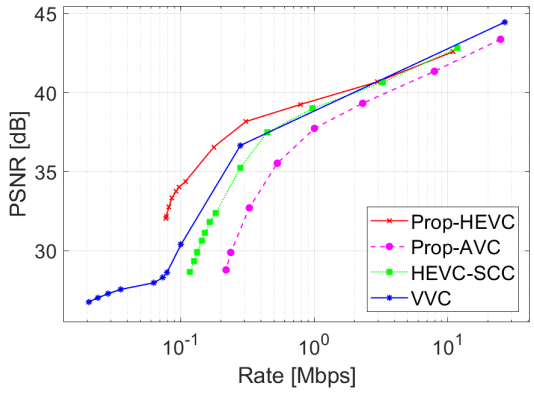
Fig. 5.10 shows the corresponding rate-distortion curves. Overall, the results are similar to those of the previous case, except that now Prop-AVC is worse than HEVC-SCC also in a couple of Cockpit HD sequences. The reason is in the large improvement of temporal prediction tools in HEVC with respect to AVC. Further insight can be gained looking at PU size distributions of the five codecs in Fig. 5.9. Here we consider sequences 8,9 and 10 at QP 35 (first two images) and QP 50 (next two images). We observe that, while Prop-HEVC is able to better use larger PU sizes than its standard counterparts, this is not the case of Prop-AVC, explaining its limited performance gain.

However, the Prop-HEVC scheme has still largely better performances than references at all rates and for all sequences. This also can be seen from Bjontegaard metrics, computed in a similar way as in the previous section, see Tab. 5.7. We observe gains up to -63 % in BD-Rate. For Cockpit SD sequences, the gains are larger. For Cockpit HUD, for very high quality the gains are no less than 25% and around 40% for very low quality.

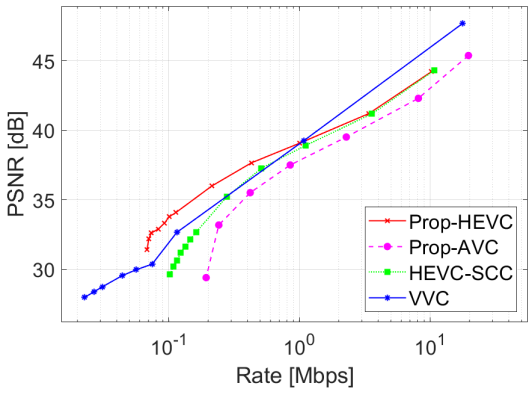
Here, we may conclude that one of our goals is accomplished: while in [145] we observed that the semantic encoder suffers from relatively high coding cost of character encoding in Intra when the residual is encoded in Inter, with the novel predictive character coding technique we achieve again large improvements with respect to the references.

5.3.4.3 Cockpit Screen Content Coding using H.266/VVC

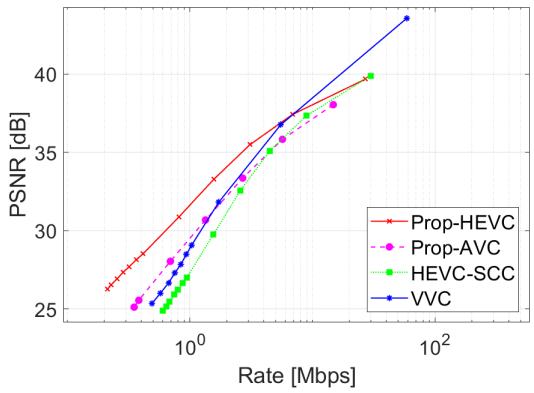
In this section we compare our proposed Prop-Intra and Prop-Inter with VVC in respectively All-Intra and Inter (low-delay) configurations. For a better understanding, we recall that the residual video is encoded with HEVC-SCC.



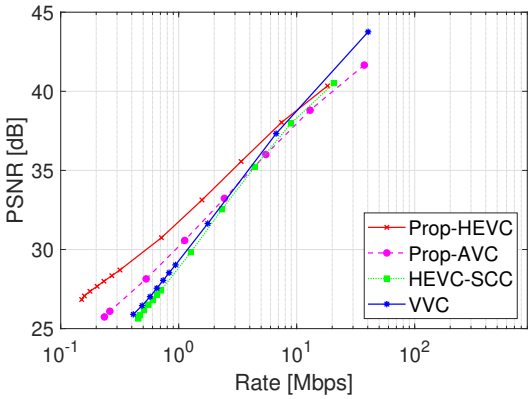
(a) Seq. 1



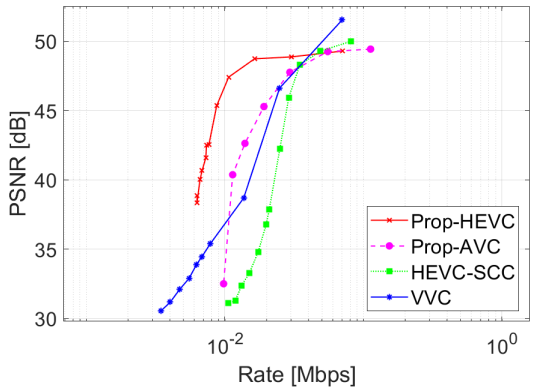
(b) Seq. 2



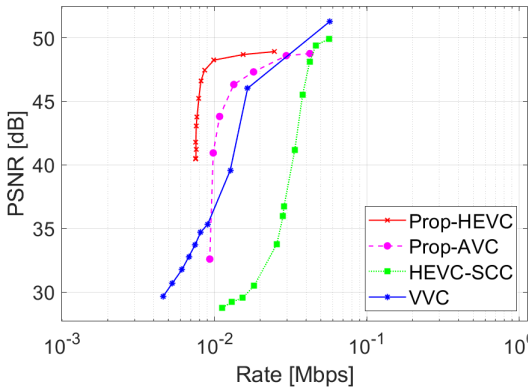
(c) Seq. 3



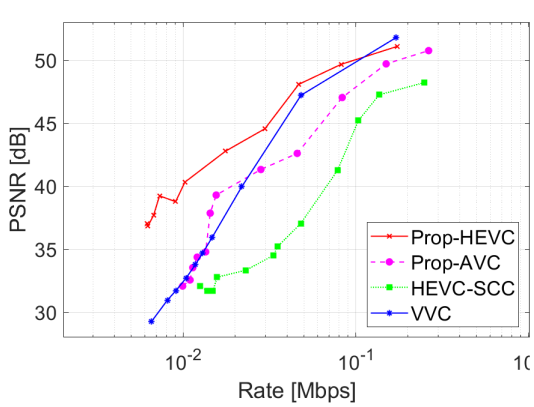
(d) Seq. 4



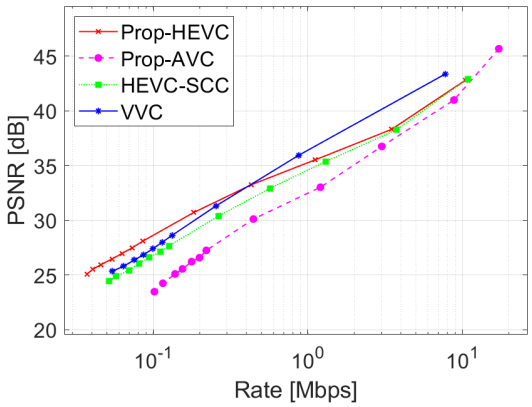
(e) Seq. 5



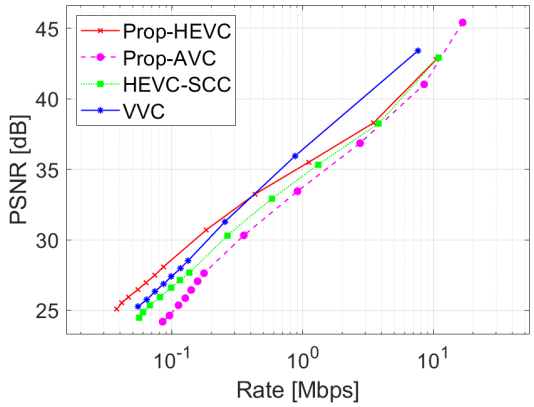
(f) Seq. 6



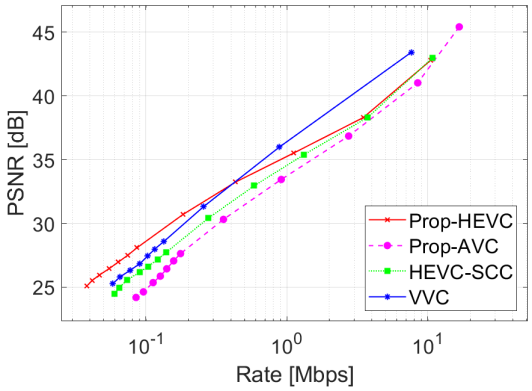
(g) Seq. 7



(h) Seq. 8



(i) Seq. 9



(j) Seq. 10

Figure 5.10: PSNR vs. video bitrate for Prop-Inter coding configuration.

Table 5.7: Bjontegaard metrics for Cockpit HD and Cockpit HUD screens for Prop-Inter configuration.

Prop-HEVC		Medium-to-High $QP = [40, 35, 30, 25]$		Low-to-Medium $QP = [50, 45, 40, 35]$	
vs.	Sequence	BD-PSNR	BD-Rate	BD-PSNR	BD-Rate
HEVC-SCC	Seq. 1	0.77 dB	-39.47 %	4.03 dB	-53.74 %
HEVC-SCC	Seq. 2	0.52 dB	-21.75 %	2.04 dB	-43.52 %
HEVC-SCC	Seq. 3	2.0 dB	-43.9 %	4.27 dB	-62.98 %
HEVC-SCC	Seq. 4	1.58 dB	-36.23 %	3.07 dB	-61.28 %
HEVC-SCC	Seq. 8	0.74 dB	-25.28 %	1.66 dB	-38.28 %
HEVC-SCC	Seq. 9	0.75 dB	-24.94 %	1.81 dB	-40.49 %
HEVC-SCC	Seq. 10	0.76 dB	-25.28 %	1.85 dB	-41.03 %

Figure 5.11: Reconstruction artefacts at QP=50 for Prop-Inter. From left to right: HEVC-SCC (PSNR 25.86 dB), VVC (PSNR 26.45), Prop-HEVC (PSNR 27.07 dB) and Prop-AVC (PSNR 26.1 dB).

Firstly, in figures 5.8 and 5.10 we illustrated the compression experiments using PSNR vs. rate curves. We have computed the Bjontegaard metric for both sequences and both methods. The results are displayed in the tables 5.8, 5.9, 5.10 and 5.11. The results in the tables 5.10 and 5.11 are computed for PSNR=[27-45] dB such that the QP= [40 - 51], because the overlapping areas corresponding to QP intervals in tables 5.8 and 5.9 was to little. As we can see, our proposed coding method achieves more than 15% bitrate savings with respect to H.266/VVC for both methods and up to 60% bitrate reduction.

Table 5.8: Bjontegaard metrics for Cockpit HD (Seq. 3 and 4) and Cockpit HUD screens for Prop-Intra.

		Very Low Quality $QP = [51, 50, 49, 48]$	
Codecs	Sequence	BD-PSNR	BD-Rate
Prop. vs. VVC	Seq. 3	0.81 dB	-32.28 %
Prop. vs. VVC	Seq. 4	1.36 dB	-37.79 %
Prop. vs. VVC	Seq. 8	0.53 dB	-15.94 %
Prop. vs. VVC	Seq. 9	0.49 dB	-14.26 %
Prop. vs. VVC	Seq. 10	0.56 dB	-15.80 %

Secondly, we have studied the impact of compression artefacts on characters readability, illustrated in figure 5.11. The figure 5.11 (the third image from left to right) confirms again that Prop-HEVC preserves the readability of the text, since the synthesized characters are not affected by compression artefacts, resulting in better PSNR. These experiments reveal

Table 5.9: Bjontegaard metrics for Cockpit HD (Seq. 3 and 4) and Cockpit HUD screens for Prop-Inter.

		Very Low Quality $QP = [51, 50, 49, 48]$	
Codecs	Sequence	BD-PSNR	BD-Rate
Prop. vs. VVC	Seq. 3	1.45 dB	-46.12 %
Prop. vs. VVC	Seq. 4	1.89 dB	-51.06 %
Prop. vs. VVC	Seq. 8	1.13 dB	-30.88 %
Prop. vs. VVC	Seq. 9	1.22 dB	-30.48 %
Prop. vs. VVC	Seq. 10	1.46 dB	-31.86 %

Table 5.10: Bjontegaard metrics for Cockpit HD (Seq. 1 and 2) and Cockpit SD screens for Prop-Intra.

		Very Low Quality $QP = [40 - 51]$	
Codecs	Sequence	BD-PSNR	BD-Rate
Prop. vs. VVC	Seq. 1	5.93 dB	-61.78 %
Prop. vs. VVC	Seq. 2	3.07 dB	-46.60 %
Prop. vs. VVC	Seq. 5	9.87 dB	-56.33 %
Prop. vs. VVC	Seq. 6	9.94 dB	-57.43 %
Prop. vs. VVC	Seq. 7	6.81 dB	-20.56 %

that proposed semantic compression scheme allows the HEVC-SCC codec to outperform the more recent and more complex H.266/VVC codec both in terms of background video quality and characters readability at low coding rates.

5.4 Conclusions

In this work we proposed to exploit the temporal redundancy to recover the occluded characters through a low-complexity method based on simple conditional probability models for the characters appearing in the video. We also propose an *ad hoc* method for character compression that take advantage from the peculiar temporal dependence of text in cockpit screen videos.

The proposed method is codec-agnostic, in the sense that any technique can be used to encode the residual images. We achieve large rate reductions when the same video codec is used as residual encoder and as reference. As a consequence, the proposed method using H.264/AVC (resp. HEVC/SCC) have in same cases better performance than the HEVC/SCC (resp. VVC) standard.

As for character recognition, the proposed method based on a model of conditional probabilities, allows a remarkable reduction of the character recognition error rate, in particular when the occlusions are sparse, as it can happen in a real cockpit screen.

As future works several directions are possible. One of them is to improve the correction algorithm by employing other models of conditional probabilities such as the usage of several past references to estimate the correct probability or to consider a period between the references. This is useful for cases in which the characters blink, for example. We

Table 5.11: Bjontegaard metrics for Cockpit HD (Seq. 1 and 2) and Cockpit SD screens for Prop-Inter.

		Very Low Quality $QP = [40 - 51]$	
Codecs	Sequence	BD-PSNR	BD-Rate
Prop. vs. VVC	Seq. 1	3.89 dB	-53.85 %
Prop. vs. VVC	Seq. 2	2.05 dB	-43.06 %
Prop. vs. VVC	Seq. 5	5.82 dB	-52.37 %
Prop. vs. VVC	Seq. 6	5.08 dB	-43.19 %
Prop. vs. VVC	Seq. 7	1.12 dB	-41.55 %

may consider also adaptive models that learn from the past behaviour the changing rate of the digits or letters. Another future perspective is to use temporal axis of the videos to improve inpainting, helping us in better reconstruct the missing pixel areas improving the quality with few percentages (i.e. around 5% as our preliminary experiments shows). This can be achieved with a context encoder-decoder architecture.

Chapter 6

Conclusions

In this chapter, we will describe the future research perspective for the components of our semantic coding scheme.

6.1 Summary

In Chapter 1 provides an introduction about screen content coding. We begin with motivating the thesis development and continue with describing airplane cockpit screens videos. Further, we introduce our contributions. Finally, we studied both the references in mixed content image and video compression techniques and on character recognition via convolutional neural networks highlighting the complexity dimension.

In Chapter 3 we illustrate our proposed scheme for semantic compressing airplane cockpit video at low bitrates that preserves the readability of computer generated graphics (text, lines). It 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 cannot preserve 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 symbols, the semantics of each character are extracted with a convolutional neural network and are predictively encoded. The symbols are then removed via inpainting, the residual background video being compressed with a standard codec and transmitted to the receiver together with the text semantics. At the decoder side, the original frame is recovered by superimposing the synthesized text using the decoded semantics over the decoded residual video.

In Chapter 4 we introduce our enhanced proposed method to encode lossless cockpit screens videos. We experimentally evaluated our semantic scheme for airplane cockpit video compression with two different video codecs, AVC/H.264 codec and H.265/HEVC codec. Moreover, we exploit the modular structure of the semantic coding scheme by retrofitting the AVC/H.264 codec with our proposed scheme. The purpose is to improve the coding efficiency keeping the complexity suitable for avionic applications. As results, in Intra-only mode our proposed scheme being able to outperform the more recent H.265/HEVC codec and performs close to its SCC extension. Contrary, when temporal inter-frame prediction is used, the competitive advantage of our semantic video compression scheme decreases as the rate of the encoded characters is not negligible any more with

respect to the rate of the residual video.

In Chapter 5 we aim to explore the temporal axis to recover the occluded characters through a low complex method and to compress the airplane cockpit video at low bitrates without affecting the readability of computer generated graphics (text, lines). By adjusting the outputs of the character classifier, we always increased classification accuracy significantly, becoming quasi-errorless by introducing some additional rules and keeping the complexity low.

In Chapter 6 we bring to the conclusions of this thesis and we discuss the possible future research perspectives.

6.2 Future Research Perspectives

Throughout this thesis, a novel semantic solution has been proposed, to compress the video displayed by the airplane cockpit screens. Nonetheless, the technological evolution of the hardware that supports this method will lead to room of improvements or changes in terms of character recognition and coding rate, the character readability being preserved by default by our proposed coding scheme. Thus, the complexity requirement of avionic applications will become more accessible in the future, increasing the accessibility to integration of more and more features.
In this section, we will discuss the future research perspective opportunities. We will refer to our proposed method in chapter 3 as the main components of discussion.

6.2.1 Character Localization and Recognition

Character localization and recognition tasks could be integrated and accomplished by means of the deep convolutional neural networks.
Currently, the state-of-the-art real time object detector is represented by YOLO, meaning You Only Look Once [115]. For example, it was used as optical character recognition system by the authors in [116] for Automatic License Plate Recognition (ALPR). The authors propose an architecture having two levels of computations. In the first level they obtain the position of the licence plate in an input image by using YOLO detector. Secondly, they extract the text by using an Optical Character Recognition method like Tesseract OCR Engine [117].
In the chapter 3, we proposed a low complex localization and detection scheme of the character in the airplane cockpit video screens.

6.2.2 Classification using Temporal Correlation

In chapter 5.2.2 we proposed an algorithm to improve the character recognition accuracy in the context of strong occlusions. The additional rules, based on the definition of the selected screen class, could be further applied in postprocessing adjustment, for example, (1) employing the correction method just when the character detector output probability is smaller than a certain threshold (2) a given character always comes from a computer screen area where its position has a meaning (i.e. its position represents the number of units, the number of tens or the number of hundreds); thus, its variation depends on the variation of the neighbouring characters, for a given computer screen (3) depending on the value of corrected estimation probability for a given character, we can employ a sliding window approach to reevaluate locally the character using the convolutional neural

network detector; therefore, for each character position we get a set of bins representing the estimated slided window label; in order to determine the character we consider the bin with the highest frequency of occurrence.

6.2.3 Inpainting

We can improve the current implementation by merging character detector and the inpainting steps by using a specialised CNN like a context encoder-decoder architecture. The purpose of it would be to classify a given input character box and to inpaint the characters pixels, improving the overall residual video quality and decreasing the necessary background coding bitrate.
In chapter 3, we proposed a low complex character and graphics removal method using classical computer vision technique for inpainting.

6.2.4 Character and Graphics Coding

In chapter 5 we proposed an algorithm to encode the characters and graphical primitives tables exploring both spatial and temporal redundancy. This solution is taking into account a past reference at $t-1$ in order to encode a current identified list of characters at t, using a set of labels.

The improvement of this algorithm can take into account multiple past or future references lists of characters.

6.2.5 Integrated Approaches

As the discussed literature proposed, we can layer the input video frames dividing the pixels in "background" or "foreground" type, by means of several techniques. The purpose is to perform content segmentation at pixel level and then to split the segmented symbols based on a dictionary of symbols and to recognise them using template matching technique, for example, or any other recognition technique that is proposed by the literature. Further, for the case of very complex graphical primitives that could be found in "foreground", the compression could be, also, easily done using some shape encoding techniques.

Firstly, we can use a structure like an autoencoder that could be able to separate the computer-generated elements (in "foreground") obtaining a residual (in "background").

Secondly, by using methods based on deep networks for pixel-wise image segmentation (e.g. fully connected networks like U-Net, SegNet, ENet), video segmentation (e.g. Clockwork Net, ICNet, variations of MobilNet with UNet) or video instance segmentation (e.g Mask R-CNN, Deep Mask) [120, 121, 122, 123] we can achieve an accurate foreground-background representation.
From another perspective, another idea could be to learn the predictibility of the feature either at video level or at semantic level. This can be integrated to improve the rate-distortion efficiency or to correct the detection task, for example.
As an instance, in the article [132], the authors propose an approach to learn from input video data which features are predictable. They trained their model on unlabeled video and the results show that that action hierarchies emerge in the representation.
In this thesis, we proposed a low-complexity semantic coding architecture for compressing airplane screens videos that is able to preserve the character readability at low to very low coding rates.

Bibliography

[1] F. Bossen, B. Bross, K. Suhring, D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Cir. and Sys. for Video Tech.*, vol. 22, pp. 1685–1696, 2012.

[2] I. Mitrica, E. Mercier, C. Ruellan, A. Fiandrotti, M. Cagnazzo, and B. Pesquet-Popescu, "Very low bitrate semantic compression of airplane cockpit screen content," *IEEE Transactions on Multimedia*, vol. 21, no. 9, pp. 2157–2170, 2019.

[3] Cast, "H264-E-BPS: Low-Power AVC/H.264 Baseline Profile Encoder," Tech. Rep., Cast, 2019.

[4] SoC Technologies, "H.264 HD Video Encoder IP Core," Tech. Rep., SoC Technologies, 2019.

[5] SoC Technologies, "H.265/HEVC HD Encoder IP Core," Tech. Rep., SoC Technologies, 2019.

[6] R. L. de Queiroz, "Compression of compound documents," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 1. IEEE, 1999, pp. 209–213.

[7] R. L. D. Queiroz, Z. Fan, and T. D. Tran, "Optimizing block-thresholding segmentation for multilayer compression of compound images," *Proceed. of IEEE Intern. Conf. Image Proc.*, vol. 9, no. 9, pp. 1461–1471, 2000.

[8] A. Said and A. Drukarev, "Simplified segmentation for compound image compression," in *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on*, vol. 1. IEEE, 1999, pp. 229–233.

[9] M. Cagnazzo, S. Parrilli, G. Poggi, and L. Verdoliva, "Costs and advantages of object-based image coding with shape-adaptive wavelet transform," *EURASIP J. Image Video Proc.*, vol. 2007, pp. Article ID 78 323, 13 pages, 2007, doi:10.1155/2007/78323.

[10] Z. Pan, H. Shen, Y. Lu, S. Li, and N. Yu, "A low-complexity screen compression scheme for interactive screen sharing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 6, pp. 949–960, 2013.

[11] T. Lin and P. Hao, "Compound image compression for real-time computer screen image transmission," *IEEE Trans. Image Processing*, vol. 14, no. 8, pp. 993–1005, 2005.

[12] C. Yang, Y. Niu, Y. Xia, and X. Cheng, "A fast and efficient codec for multimedia applications in wireless thin-client computing," in *2007 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, June 2007, pp. 1–12.

[13] W. Ding, D. Liu, Y. He, and F. Wu, "Block-based fast compression for compound images," in *Proceed. of IEEE Intern. Conf. on Multim. and Expo*, July 2006, pp. 809–812.

[14] M. Decombas, Y. Fellah, F. Dufaux, B. Pesquet-popescu, F. Capman, and E. Renan, "Seam carving modeling for semantic video coding in security applications," *IEEE Trans. Signal Processing*, vol. 4, pp. 1–24, Aug. 2015.

[15] R. A. C. Jizheng Xu, Rajan Joshi, "Overview of the emerging HEVC screen content coding extension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, pp. 50–62, Jan. 2016.

[16] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.

[17] Y. Le Cun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. [Online]. Available: http://leon.bottou.org/papers/lecun-98h

[18] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Processing Mag.*, vol. 31, no. 1, pp. 127–144, 2014.

[19] W.-H. Peng, F. G. Walls, R. A. Cohen, J. Xu, J. Ostermann, A. MacInnis, and T. Lin, "Overview of screen content video coding: Technologies, standards, and beyond," *IEEE J. Emerg. Select. Circuits Syst.*, vol. 6, pp. 393–408, Dec. 2016.

[20] R. L. de Queiroz, R. R. Buckley, and M. Xu, "Mixed raster content (mrc) model for compound image compression," in *Visual Communications and Image Processing'99*, vol. 3653. International Society for Optics and Photonics, 1998, pp. 1106–1118.

[21] P. Haffner, L. Bottou, P. G. Howard, and Y. LeCun, "Djvu: analyzing and compressing scanned documents for internet distribution," in *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*, Sept 1999, pp. 625–628.

[22] S. Wang, X. Zhang, X. Liu, J. Zhang, S. Ma, and W. Gao, "Utility-driven adaptive preprocessing for screen content video compression," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 660–667, 2017.

[23] W. Zhu, W. Ding, J. Xu, Y. Shi, and B. Yin, "Hash-based block matching for screen content coding," *IEEE Trans. Multimedia*, vol. 17, no. 7, pp. 935–944, July 2015.

[24] W. Zhu and W. Ding and J. Xu and Y. Shi and B. Yin, "Screen content coding based on HEVC framework," *IEEE Trans. Multimedia*, vol. 16, no. 5, pp. 1316–1326, Aug 2014.

[25] L. Zhao, T. Lin, K. Zhou, S. Wang, and X. Chen, "Pseudo 2d string matching technique for high efficiency screen content coding." *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 339–350, 2016.

[26] J.-W. Kang, S.-K. Ryu, N.-Y. Kim, and M.-J. Kang, "Efficient residual dpcm using an l_1 robust linear prediction in screen content video coding," *IEEE Trans. Multimedia*, vol. 18, no. 10, pp. 2054–2065, 2016.

[27] M. Mrak and J. Z. Xu, "Improving screen content coding in hevc by transform skipping," in *Proceed. of Europ. Sign. Proc. Conf.*, Aug 2012, pp. 1209–1213.

[28] L. Guo, D. Zhou, and S. Goto, "A new reference frame recompression algorithm and its vlsi architecture for uhdtv video codec," *IEEE Trans. Multimedia*, vol. 16, no. 8, pp. 2323–2332, 2014.

[29] H.-C. Kuo and Y.-L. Lin, "A hybrid algorithm for effective lossless compression of video display frames." *IEEE Trans. Multimedia*, vol. 14, no. 3-1, pp. 500–509, 2012.

[30] V. E. S. Association *et al.*, "Vesa display stream compression(dsc) standard v1. 2," 2016.

[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[32] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[33] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," *Europ. Conf. Computer Vision*, 2014.

[34] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *International Journal of Computer Vision*, vol. 116, no. 1, pp. 1–20, 2016.

[35] X.-C. Yin, Z.-Y. Zuo, S. Tian, and C.-L. Liu, "Text detection, tracking and recognition in video: A comprehensive survey," *IEEE J. Emerg. Select. Circuits Syst.*, vol. 25, pp. 2752–2773, Jun. 2016.

[36] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, pp. 62–66, Jan. 1979.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.

[38] G. E. H. David E. Rumelhart and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[39] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *IEEE J. Emerg. Select. Circuits Syst.*, vol. 105, no. 12, pp. 2295–2329, Jun. 2017.

[40] T. A. Welch, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19, Jun. 1984.

[41] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, no. 1, pp. 11–15, Jan. 1972.

[42] J. Matas, C. Galambos, and J. Kittler, "Robust detection of lines using the progressive probabilistic Hough transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.

[43] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[44] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," in *VCEG Meeting*, Austin, USA, Apr. 2001.

[45] G. Feng, Z. Hu, S. Chen, and F. Wu, "Energy-efficient and high-throughput fpga-based accelerator for convolutional neural networks," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Oct 2016, pp. 624–626.

[46] Xilinx, "Deep learning on fpga," Xilinx, Tech. Rep., Nov. 2018. [Online]. Available: https://www.xilinx.com/support/documentation/white_papers/wp504-accel-dnns.pdf

[47] Xilinx, "H.264/H.265 video codec unit v1.0," Xilinx, Tech. Rep., Nov. 2017. [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/vcu/v1_0/pg252-vcu.pdf

[48] Xilinx, "Responsive and reconfigurable vision systems," Xilinx, Tech. Rep., Nov. 2018. [Online]. Available: https://www.xilinx.com/products/design-tools/embedded-vision-zone.html

[49] L. L. K. A. Rickard Sjoberg, Jacob Ström, "Versatile video coding explained - the future of video in a 5g world," *Review Ericsson Technology*, 2021.

[50] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc)," *Proceedings of the IEEE*, pp. 1–31, 2021.

[51] Bitmovin. (2021) Versatile video coding: Vvc (h266) codec is finalized. [Online]. Available: https://bitmovin.com/webinars/state-of-compression-vvc/?submissionGuid=0aaa12e1-14fc-4871-b3f5-5df8951761fb

[52] S. Liu, X. Xu, S. Lei, and K. Jou, "Overview of hevc extensions on screen content coding," *APSIPA Transactions on Signal and Information Processing*, vol. 4, p. e10, 2015.

[53] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. USA: Kluwer Academic Publishers, 1997.

[54] Franhofer. (2021) Rate distortion optimization (rdo) for encoder control. [Online]. Available: https://www.hhi.fraunhofer.de/en/departments/vca/research-groups/video-coding-technologies/research-topics/past-research-topics/rate-distortion-optimization-rdo-for-encoder-control.html

[55] Y. Q. Shi and H. Sun, *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, 2nd ed. USA: CRC Press, Inc., 2008.

[56] W.-H. Peng, F. Walls, R. Cohen, J. Xu, J. Ostermann, A. G. MacInnis, and T. Lin, "Overview of screen content video coding: Technologies, standards, and beyond," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 6, pp. 393–408, 2016.

[57] M. Sulc and J. Matas, "Improving cnn classifiers by estimating test-time priors," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[58] R. Alaiz-Rodriguez and J. Cid-Sueiro, "Minimax strategies for training classifiers under unknown priors," in *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, 2002, pp. 249–258.

[59] M. Saerens, P. Latinne, and C. Decaestecker, "Adjusting the outputs of a classifier to new a priori probabilities: A simple procedure," *Neural computation*, vol. 14, pp. 21–41, 02 2002.

[60] G. J. McLachlan and T. Krishnan, *The EM algorithm and extensions*. John Wiley & Sons, 2007, vol. 382.

[61] X.-H. Liu, A. Skidmore, and H. Van Oosten, "Integration of classification methods for improvement of land-cover map accuracy," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 56, no. 4, pp. 257 – 268, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271602000618

[62] J. C. Perez-Cortes, J. C. Amengual, J. Arlandis, and R. Llobet, "Stochastic error-correcting parsing for ocr post-processing," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, vol. 4, 2000, pp. 405–408 vol.4.

[63] W. Duch and Ł. Itert, "A posteriori corrections to classification methods," in *Neural Networks and Soft Computing*, L. Rutkowski and J. Kacprzyk, Eds. Heidelberg: Physica-Verlag HD, 2003, pp. 406–411.

[64] J. Kittler, "Combining classifiers: A theoretical framework," *Pattern analysis and Applications*, vol. 1, no. 1, pp. 18–27, 1998.

[65] Z. C. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," *arXiv preprint arXiv:1802.03916*, 2018.

[66] S. Destercke and B. Quost, "Correcting binary imprecise classifiers: Local vs global approach," in *Scalable Uncertainty Management*, E. Hüllermeier, S. Link, T. Fober, and B. Seeger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 299–310.

[67] R. Ghani, "Using error-correcting codes for text classification," in *ICML*. Citeseer, 2000, pp. 303–310.

[68] A. Berger, "Error-correcting output coding for text classification," in *IJCAI-99: Workshop on machine learning for information filtering*. Citeseer, 1999.

[69] Q. Ye, J. Liang, and J. Jiao, "Pedestrian detection in video images via error correcting output code classification of manifold subclasses," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 193–202, 2012.

[70] S. Tao and Z. L. W. JianLin, "Design and application of an xml data compression algorithm based on huffman coding," *Journal of Beijing University of Chemical Technology (Natural Science Edition)*, vol. 4, 2013.

[71] M. Sharma *et al.*, "Compression using huffman coding," *IJCSNS International Journal of Computer Science and Network Security*, vol. 10, no. 5, pp. 133–141, 2010.

[72] J. Yang, Z. Zhang, N. Zhang, M. Li, Y. Zheng, L. Wang, Y. Li, J. Yang, Y. Xiang, and Y. Zhang, "Vehicle text data compression and transmission method based on maximum entropy neural network and optimized huffman encoding algorithms," *Complexity*, vol. 2019, 2019.

[73] K. C. Barr and K. Asanović, "Energy-aware lossless data compression," *ACM Transactions on Computer Systems (TOCS)*, vol. 24, no. 3, pp. 250–291, 2006.

[74] J. M. Jou and P.-Y. Chen, "A fast and efficient lossless data-compression method," *IEEE Transactions on Communications*, vol. 47, no. 9, pp. 1278–1283, 1999.

[75] E. J. Leavline and D. Singh, "Hardware implementation of lzma data compression algorithm," *International Journal of Applied Information Systems (IJAIS)*, vol. 5, no. 4, pp. 51–56, 2013.

[76] N. Ranganathan and S. Henriques, "High-speed vlsi designs for lempel-ziv-based data compression," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 2, pp. 96–106, 1993.

[77] B. Mohan and V. Govindan, "Idbe-an intelligent dictionary based encoding algorithm for text data compression for high speed data transmission over internet," *arXiv preprint cs/0601077*, 2006.

[78] A. Moffat, "Implementing the ppm data compression scheme," *IEEE Transactions on Communications*, vol. 38, no. 11, pp. 1917–1921, 1990.

[79] S. Wang, J. Fu, Y. Lu, S. Li, and W. Gao, "Content-aware layered compound video compression," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2012, pp. 145–148.

[80] H. Shen, Y. Lu, F. Wu, and S. Li, "A high-performanance remote computing platform," in *2009 IEEE International Conference on Pervasive Computing and Communications*. IEEE, 2009, pp. 1–6.

[81] B. Han, D. Wu, and H. Zhang, "Block-based method for real-time compound video compression," in *Mobile Multimedia/Image Processing, Security, and Applications 2010*, vol. 7708. International Society for Optics and Photonics, 2010, p. 77080S.

[82] D. Miao, J. Fu, Y. Lu, S. Li, and C.-W. Chen, "Layered screen video coding leveraging hardware video codec," 07 2013, pp. 1–6.

[83] D. Miao, J. Fu, Y. Lu, S. Li, and C. W. Chen, "A high-fidelity and low-interaction-delay screen sharing system," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 3, pp. 1–23, 2016.

[84] "ISO/IEC 14496-17:2006 Information technology — Coding of audio-visual objects — Part 17: Streaming text format," *ISO/IEC 14496-17:2006*, pp. i –21, 2006-04.

[85] D. C. A. Bulterman, A. J. Jansen, P. Cesar, and S. Cruz-Lara, "An efficient, streamable text format for multimedia captions and subtitles," in *Proceedings of the 2007 ACM Symposium on Document Engineering*. New York, NY, USA: Association for Computing Machinery, 2007, p. 101–110. [Online]. Available: https://doi.org/10.1145/1284420.1284451

[86] I. Tsukagoshi and M. Yamashita, "Subtitle encoding/decoding method and apparatus," Dec. 8 1998, uS Patent 5,848,217.

[87] Y. Yagasaki, "Apparatus and method for encoding and decoding a subtitle signal," Dec. 8 1998, uS Patent 5,847,770.

[88] K. Mori, K. Imahashi, K. Tanaka, T. Setogawa, and H. Yamauchi, "Device for compressing audio and video data and method therefor," Jun. 14 2001, uS Patent App. 08/860,110.

[89] L. Zhao, X. Zhang, X. Zhang, S. Wang, S. Wang, S. Ma, and W. Gao, "Intelligent analysis oriented surveillance video coding," in *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, pp. 37–42.

[90] J. Xiao, R. Hu, L. Liao, Y. Chen, Z. Wang, and Z. Xiong, "Knowledge-based coding of objects for multisource surveillance video data," *IEEE Transactions on Multimedia*, vol. 18, no. 9, pp. 1691–1706, 2016.

[91] K. Ogasawara, T. Miyazaki, Y. Sugaya, and S. Omachi, "Object-based video coding by visual saliency and temporal correlation," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[92] N. AL-SHAKARJI, F. Bunyak, H. Aliakbarpour, G. Seetharaman, and K. Palaniappan, "Multi-cue vehicle detection for semantic video compression in georegistered aerial videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[93] S. Luo, Y. Yang, Y. Yin, C. Shen, Y. Zhao, and M. Song, "Deepsic: Deep semantic image compression," in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds. Cham: Springer International Publishing, 2018, pp. 96–106.

[94] M. Akbari, J. Liang, and J. Han, "Dsslic: Deep semantic segmentation-based layered image compression," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2042–2046.

[95] T. Strutz and P. Möller, "Screen content compression based on enhanced soft context formation," *IEEE Transactions on Multimedia*, vol. 22, no. 5, pp. 1126–1138, 2019.

[96] T. Bell, I. H. Witten, and J. G. Cleary, "Modeling for text compression," *ACM Computing Surveys (CSUR)*, vol. 21, no. 4, pp. 557–591, 1989.

[97] S. Shanmugasundaram and R. Lourdusamy, "A comparative study of text compression algorithms," *International Journal of Wisdom Based Computing*, vol. 1, no. 3, pp. 68–76, 2011.

[98] R. Y. K. Isal and A. Moffat, "Word-based block-sorting text compression," in *Proceedings 24th Australian Computer Science Conference. ACSC 2001.* IEEE, 2001, pp. 92–99.

[99] A. Moffat and R. Y. K. Isal, "Word-based text compression using the burrows–wheeler transform," *Information processing & management*, vol. 41, no. 5, pp. 1175–1192, 2005.

[100] J. Platoš, V. Snášel, and E. El-Qawasmeh, "Compression of small text files," *Advanced engineering informatics*, vol. 22, no. 3, pp. 410–417, 2008.

[101] R. Pizzolante, B. Carpentieri, A. Castiglione, A. Castiglione, and F. Palmieri, "Text compression and encryption through smart devices for mobile communication," in *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing.* IEEE, 2013, pp. 672–677.

[102] E. Satir and H. Isik, "A compression-based text steganography method," *Journal of Systems and Software*, vol. 85, no. 10, pp. 2385–2394, 2012.

[103] J. Schmidhuber and S. Heil, "Sequential neural text compression," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 142–146, 1996.

[104] Z. He and S. K. Mitra, "From rate-distortion analysis to resource-distortion analysis," *IEEE Circuits and systems magazine*, vol. 5, no. 3, pp. 6–18, 2005.

[105] J. Stottrup-Andersen, S. Forchhammer, and S. M. Aghito, "Rate-distortion-complexity optimization of fast motion estimation in h. 264/mpeg-4 avc," in *2004 International Conference on Image Processing, 2004. ICIP'04.*, vol. 1. IEEE, 2004, pp. 111–114.

[106] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE transactions on circuits and systems for video technology*, vol. 15, no. 5, pp. 645–658, 2005.

[107] G. Corrêa, P. Assuncao, L. Agostini, and L. A. da Silva Cruz, "Complexity control of high efficiency video encoders for power-constrained devices," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 4, pp. 1866–1874, 2011.

[108] J. A. L. d. Castillo and N. Couture, "The aircraft of the future: towards the tangible cockpit," in *Proceedings of the International Conference on Human-Computer Interaction in Aerospace*, 2016, pp. 1–8.

[109] T. A. Blog. (2021) Glass cockpit. [Online]. Available: https://en.wikipedia.org/wiki/Glass_cockpit

[110] F. Global. (2021) Looking at the thinking cockpit. [Online]. Available: https://www.flightglobal.com/looking-at-the-thinking-cockpit/12897.article

[111] S. D. Systems. (2021) Airborne mission data recorder and server. [Online]. Available: https://www.safrandatasystemsus.com/datasheets/Avionics/VS1410_Datasheet.pdf

[112] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Optical Pattern Recognition XII*, vol. 4387. International Society for Optics and Photonics, 2001, pp. 95–102.

[113] A. Kaso, "Computation of the normalized cross-correlation by fast fourier transform," *PloS one*, vol. 13, no. 9, p. e0203434, 2018.

[114] J. Luo and E. E. Konofagou, "A fast normalized cross-correlation calculation method for motion estimation," *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, vol. 57, no. 6, pp. 1347–1357, 2010.

[115] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[116] J. Gandhi, P. Jain, and L. Kurup, "Yolo based recognition of indian license plates," in *Advanced Computing Technologies and Applications*. Springer, 2020, pp. 411–421.

[117] Google. (2021) Tesseract OCR. [Online]. Available: https://opensource.google/projects/tesseract

[118] H.-H. Chang and H. Yan, "Vectorization of hand-drawn image using piecewise cubic bezier curves fitting," *Pattern recognition*, vol. 31, no. 11, pp. 1747–1755, 1998.

[119] V. Egiazarian, O. Voynov, A. Artemov, D. Volkhonskiy, A. Safin, M. Taktasheva, D. Zorin, and E. Burnaev, "Deep vectorization of technical drawings," in *European Conference on Computer Vision*. Springer, 2020, pp. 582–598.

[120] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[121] R. Yao, G. Lin, S. Xia, J. Zhao, and Y. Zhou, "Video object segmentation and tracking: A survey," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 4, pp. 1–47, 2020.

[122] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[123] J. Qi, Y. Gao, X. Liu, Y. Hu, X. Wang, X. Bai, P. H. Torr, S. Belongie, A. Yuille, and S. Bai, "Occluded video instance segmentation," *arXiv preprint arXiv:2102.01558*, 2021.

[124] J. Redmon and A. Farhadi. (2021) Yolo. [Online]. Available: https://pjreddie.com/darknet/yolo/

[125] C. Ding, S. Wang, N. Liu, K. Xu, Y. Wang, and Y. Liang, "Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 33–42.

[126] D. T. Nguyen, T. N. Nguyen, H. Kim, and H. Lee, "A high-throughput and power-efficient fpga implementation of yolo cnn for object detection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 8, pp. 1861–1873, 2019.

[127] G. Wei, Y. Hou, Q. Cui, G. Deng, X. Tao, and Y. Yao, "Yolo acceleration using fpga architecture," in *2018 IEEE/CIC International Conference on Communications in China (ICCC)*, 2018, pp. 734–735.

[128] N. Zhang, X. Wei, H. Chen, and W. Liu, "Fpga implementation for cnn-based optical remote sensing object detection," *Electronics*, vol. 10, no. 3, p. 282, 2021.

[129] S. Mittal, "A survey of fpga-based accelerators for convolutional neural networks," *Neural computing and applications*, vol. 32, no. 4, pp. 1109–1139, 2020.

[130] A. Azarmi Gilan, M. Emad, and B. Alizadeh, "Fpga-based implementation of a real-time object recognition system using convolutional neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 4, pp. 755–759, 2020.

[131] Yuan Jing, B. Youssefi, M. Mirhassani, and R. Muscedere, "An efficient fpga implementation of optical character recognition for license plate recognition," in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2017, pp. 1–4.

[132] D. Suris, R. Liu, and C. Vondrick, "Learning the predictability of the future," *arXiv preprint arXiv:2101.01600*, 2021.

[133] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[134] I. Pavlov. (2021) 7z format. [Online]. Available: https://www.7-zip.org/7z.html

[135] M. Burrows and D. Wheeler, "A block-sorting lossless data compression algorithm," in *Digital SRC Research Report*. Citeseer, 1994.

[136] D. Shkarin, "Ppm: One step to practicality," 02 2002, pp. 202 – 211.

[137] G. Wallace, "The jpeg still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, 1992.

[138] L. Chiariglione, *Even the stars die: The history of MPEG and how it made digital media happen*. Kindle Edition, 2021.

[139] M. Wien, "High efficiency video coding," *Coding Tools and specification*, vol. 24, 2015.

[140] Airbus. (2021) Getting to grips with datalink. [Online]. Available: https://www.cockpitseeker.com/wp-content/uploads/goodies/ac/a320/pdf/data/datalink.pdf

[141] W. M. Organization. (2021) Satellite data telecommunication handbook. [Online]. Available: https://library.wmo.int/doc_num.php?explnum_id=5222

[142] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h. 264/AVC video coding standard," *IEEE Transactions on circuits and systems for video technology*, vol. 13, no. 7, pp. 560–576, 2003.

[143] B. Bross, Y.-K. Wang, Y. Ye, S. Liu, J. Chen, G. J. Sullivan, and J.-R. Ohm, "Overview of the versatile video coding (VVC) standard and its applications," *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.

[144] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[145] I. Mitrica, A. Fiandrotti, M. Cagnazzo, E. Mercier, and C. Ruellan, "Cockpit video coding with temporal prediction," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*, 2019, pp. 28–33.

ECOLE
DOCTORALE

Titre : Compression vidéo du contenu des écrans de cockpit d'avion

Mots clés : HEVC, AVC, codage de contenu d'écran, vidéo de cockpit d'avion, réseaux de neurones convolutifs, codage vidéo sémantique

Résumé : Cette thèse aborde le problème de l'encodage de la vidéo des cockpits d'avion. Le cockpit des avions de ligne modernes consiste en un ou plusieurs écrans affichant l'état des instruments de l'avion (par exemple, la position de l'avion telle que rapportée par le GPS, le niveau de carburant tel que lu par les capteurs dans les réservoirs, etc.,) souvent superposés au naturel images (par exemple, cartes de navigation, caméras extérieures, etc.). Les capteurs d'avion sont généralement inaccessibles pour des raisons de sécurité, de sorte que l'enregistrement du cockpit est souvent le seul moyen de consigner les données vitales de l'avion en cas, par exemple, d'un accident. Les contraintes sur la mémoire d'enregistrement disponible à bord nécessitent que la vidéo du cockpit soit codée à des débits faibles à très faibles, alors que pour des raisons de sécurité, les informations textuelles doivent rester intelligibles après le décodage. De plus, les contraintes sur l'enveloppe de puissance des dispositifs avioniques limitent la complexité du sous-système d'enregistrement du poste de pilotage.
Au fil des ans, un certain nombre de schémas de codage d'images ou de vidéos avec des contenus mixtes générés par ordinateur et naturels ont été proposés. Le texte et d'autres graphiques générés par ordinateur produisent des composants haute fréquence dans le domaine transformé. Par conséquent, la perte due à la compression peut nuire à la lisibilité de la vidéo et donc à son utilité. Par exemple, l'extension récemment normalisée SCC (Screen Content Coding) de la norme H.265/HEVC comprend des outils conçus explicitement pour la compression du contenu de l'écran. Nos expériences montrent cependant que les artefacts persistent aux bas débits ciblés par notre application, incitant à des schémas où la vidéo n'est pas encodée dans le domaine des pixels.
Cette thèse propose des méthodes de codage d'écran de faible complexité où le texte et les primitives graphiques sont codés en fonction de leur sémantique plutôt que sous forme de blocs de pixels. Du côté du codeur, les caractères sont détectés et lus à l'aide d'un réseau neuronal convolutif. Les caractères détectés sont ensuite supprimés de l'écran via le pixel inpainting, ce qui donne une vidéo résiduelle plus fluide avec moins de hautes fréquences. La vidéo résiduelle est codée avec un codec vidéo standard et est transmise du côté récepteur avec une sémantique textuelle et graphique en tant qu'informations secondaires. Du côté du décodeur, le texte et les graphiques sont synthétisés à l'aide de la sémantique décodée et superposés à la vidéo résiduelle, récupérant finalement l'image d'origine. Nos expériences montrent qu'un encodeur AVC/H.264 équipé de notre méthode a de meilleures performances de distorsion-débit que H.265/HEVC et se rapproche de celle de son extension SCC.
Si les contraintes de complexité permettent la prédiction inter-trame, nous exploitons également le fait que les caractères co-localisés dans les trames voisines sont fortement corrélés. À savoir, les symboles mal classés sont récupérés à l'aide d'une méthode proposée basée sur un modèle de faible complexité des probabilités de transition pour les caractères et les graphiques. Concernant la reconnaissance de caractères, le taux d'erreur chute jusqu'à 18 fois dans les cas les plus faciles et au moins 1,5 fois dans les séquences les plus difficiles malgré des occlusions complexes. En exploitant la redondance temporelle, notre schéma s'améliore encore en termes de distorsion de débit et permet un décodage de caractères quasi sans erreur. Des expériences avec de vraies séquences vidéo de cockpit montrent des gains de distorsion de débit importants pour la méthode proposée par rapport aux normes de compression vidéo.

Title : Video Compression of Airplane Cockpit Screens Content

Keywords : HEVC, AVC, screen content coding, airplane cockpit video, convolutional neural networks, semantic video coding

Abstract : This thesis addresses the problem of encoding the video of airplane cockpits. The cockpit of modern airliners consists in one or more screens displaying the status of 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.,) often superimposed over natural images (e.g., navigation maps, outdoor cameras, etc.). Plane sensors are usually inaccessible due to security reasons, so recording the cockpit is often the only way to log vital plane data in the event of, e.g., an accident. Constraints on the recording storage available on-board require the cockpit video to be coded at low to very low bitrates, whereas safety reasons require the textual information to remain intelligible after decoding. In addition, constraints on the power envelope of avionic devices limit the cockpit recording subsystem complexity.

Over the years, a number of schemes for coding images or videos with mixed computer-generated and natural contents have been proposed. Text and other computer generated graphics yield high-frequency components in the transformed domain. Therefore, the loss due to compression may hinder the readability of the video and thus its usefulness. For example, the recently standardized Screen Content Coding (SCC) extension of the H.265/HEVC standard includes tools designed explicitly for screen contents compression. Our experiments show however that artifacts persist at the low bitrates targeted by our application, prompting for schemes where the video is not encoded in the pixel domain.

This thesis proposes methods for low complexity screen coding where text and graphical primitives are encoded in terms of their semantics rather than as blocks of pixels. At the encoder side, characters are detected and read using a convolutional neural network. Detected characters are then removed from screen via pixel inpainting, yielding a smoother residual video with fewer high frequencies. The residual video is encoded with a standard video codec and is transmitted to the receiver side together with text and graphics semantics as side information. At the decoder side, text and graphics are synthesized using the decoded semantics and superimposed over the residual video, eventually recovering the original frame. Our experiments show that an AVC/H.264 encoder retrofitted with our method has better rate-distortion performance than H.265/HEVC and approaches that of its SCC extension.

If the complexity constraints allow inter-frame prediction, we also exploit the fact that co-located characters in neighbor frames are strongly correlated. Namely, the misclassified symbols are recovered using a proposed method based on low-complexity model of transitional probabilities for characters and graphics. Concerning character recognition, the error rate drops up to 18 times in the easiest cases and at least 1.5 times in the most difficult sequences despite complex occlusions. By exploiting temporal redundancy, our scheme further improves in rate-distortion terms and enables quasi-errorless character decoding. Experiments with real cockpit video footage show large rate-distortion gains for the proposed method with respect to video compression standards.

Institut Polytechnique de Paris
91120 Palaiseau, France