Video for events : Compression and transport of the next generation video codec

Mourad Aklouf

université PARIS-SACLAY

Video for events: Compression and transport of the next generation video codec

Vidéo pour l'événementiel : Compression et transport de la nouvelle génération de codec vidéo

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580 : Sciences and Technologies of Information and Communication (STIC)
Spécialité de doctorat : Traitement du signal et des images
Graduate School : Informatique et Sciences du Numérique (ISN)
Référent : CentraleSupélec

Thèse préparée dans le Laboratoire des signaux et systèmes (Université Paris-Saclay, CentraleSupélec). Sous la direction de Frédéric DUFAUX, Directeur de recherche CNRS, le co-encadrement de Michel KIEFFER, Professeur des universités UPSay, et la co-supervision de Marc LENY, Directeur R&D Ektacom

Thèse soutenue à Paris-Saclay, le 04 Mai 2022, par

Mourad AKLOUF

Composition du jury

MOKRAOUI Anissa Professeur des universités, Université Sorbonne Paris Nord	Présidente
COUDOUX François-Xavier Professeur des universités, Université Polytechnique Hauts-de-France	Rapporteur & examinateur
ROUMY Aline Directrice de recherche, INRIA Rennes	Rapporteur & examinatrice
TIMMERER Christian Professeur associé, Université de Klagenfurt	Examinateur
DUFAUX Frédéric Directeur de recherche, Laboratoire des Signaux et Systèmes (L2S), CNRS – UPSay	Directeur de thèse
KIEFFER Michel Professeur des universités, Université Paris-Saclay	Co-directeur
LENY Marc Directeur R&D, PhD, Ektacom	Co-encadrant

THESE DE DOCTORAT
NNT : 2022UPASG029

Contents

List of Figures

List of Tables

À ma chère mère.

Remerciements

La réalisation de cette thèse a été possible grâce aux efforts de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je voudrais dans un premier temps remercier mon directeur de thèse, Frédéric DUFAUX, pour sa disponibilité et ses nombreux conseils durant la réalisation de cette thèse.

Je tiens aussi à exprimer ma gratitude à mon encadreur, Michel KIEFFER, et le remercier pour son aide et sa disponibilité au long de cette thèse.

Merci à mon superviseur industriel au sein de Ektacom, Marc LENY, pour son attention, son encouragement et son accompagnement tout au long de ces années. Je remercie également tous les membres de l'équipe R&D d'Ektacom pour m'avoir ouvert les portes de l'entreprise sans aucune limite.

Enfin, je suis redevable à ma mère, Assia MATIB, pour son soutien moral et sa confiance indéfectible dans mes choix.

Résumé

Contexte de la thèse

Le streaming vidéo à faible latence est une application clé dans la diffusion d'événements sportifs, la visioconférence ou encore le contrôle et la conduite à distance. Selon le rapport annuel sur Internet de Cisco, le trafic de la vidéo en direct a augmenté de 93% en 2020 et représentera 17% du trafic vidéo sur Internet en 2022 [1]. Le streaming en direct deviendra encore plus populaire grâce aux progrès des réseaux de télécommunication et des codecs vidéo.

De nombreuses méthodes de streaming exploitent le protocole HTTP pour Diffuser la vidéo. Ces méthodes sont connues par "HTTP Adaptive Streaming" (HAS) [2]. Avec HAS, les clips vidéo sont découpés en segments de quelques secondes, encodés à plusieurs débits, et stockés sur des serveurs média HTTP. En utilisant HAS, un client peut demander les segments vidéo avec les débits d'encodage appropriés selon la condition de son réseau. Néanmoins, les méthodes HAS ne précisent pas de logique d'adaptation. Dans les applications de streaming en liaison descendante, des algorithmes d'adaptation sont généralement mis en œuvre chez le client pour sélectionner le débit optimal de chaque segment demandé.

Ces algorithmes exploitent des mesures instantanées ou moyennées du canal de transmission et/ou du niveau de tampon client. Ils visent à maximiser la Qualité d'Expérience (QoE) du client. Ceci est généralement obtenu en maximisant la qualité de la vidéo reçue tout en minimisant le nombre de gels et de changements de qualité vidéo.

Dans le streaming vidéo en direct, l'acquisition, l'encodage et la transmission vidéo sont effectués en temps réel. Minimiser le délai entre l'acquisition d'une image et son affichage chez le client est une exigence QoE supplémentaire. Le streaming en direct est nettement plus difficile que le streaming

vidéo à la demande (VOD) classique. Tout d'abord, de grands tampons sont généralement implémentés au niveau du client pour atténuer les variations de bande passante. Ces tampons induisent un délai important dans le contexte du streaming en direct, leur taille doit donc être minimisée. De plus, les algorithmes de contrôle de débit au niveau du client basés sur HAS peuvent entraîner des retards importants car les décisions de contrôle sont prises à la même période que la durée du segment vidéo. Lorsque la bande passante varie avec une échelle de temps plus petite, de mauvaises décisions peuvent induire des retards de téléchargement importants.

Enfin, dans plusieurs applications telles que le streaming vidéo en direct à l'intérieure d'une voiture pendant une course ou le contrôle à distance d'un drone, le transmetteur transmet le flux vidéo compressé sur un réseau d'accès sans fil 4G/5G au client. La mobilité du transmetteur induit des variations importantes et rapides des caractéristiques du canal sans fil. Dans un tel contexte, les approches de contrôle côté émetteur apparaissent mieux adaptées pour sélectionner les paramètres de codage vidéo. Cela évite également d'attendre des rapports sur les états du réseau et de la mémoire tampon fournis par le client.

En outre, la nouvelle génération de codecs vidéo présente un avantage significatif dans le contexte du streaming à faible latence. Le Versatile Video Coding (VVC) atteint un gain de compression de 50% par rapport à son prédécesseur, le High-Efficiency Video Coding (HEVC), pour la même qualité PSNR. Cependant, VVC se concentre principalement sur le contenu vidéo Ultra-Haute Définition (UHD). Une variété d'outils de codage a été utilisée pour atteindre une efficacité de codage élevée. Ces nouveaux outils impliquent une quantité importante de complexité de calcul. La version actuelle de l'encodeur VVC (VTM5.0) a 10 fois le temps d'exécution de l'encodeur HEVC (HM16) [3]. Par conséquent, le codeur VVC n'est pas adapté aux applications en temps réel, même pour l'encodage à faible débit et les contenus à faible résolution qui sont encore utilisés pour le streaming vidéo aujourd'hui (*e.g.*, 480p et 360p).

Objectifs de la thèse

Dans cette thèse, nous abordons la problématique du streaming vidéo à faible latence à partir d'un transmetteur mobile dans des conditions de réseau variables.

- Nous proposons un algorithme d'adaptation du débit vidéo piloté par le transmetteur pour le streaming vidéo à faible latence à partir d'un émetteur mobile. L'algorithme proposé ajuste le débit vidéo au niveau de l'image et en fonction de l'état de la liaison montante du canal et du niveau de tampon de l'émetteur.
- Le débit de l'image est ajusté au moyen d'un modèle de débit qui détermine le paramètre de quantification pour avoir un bitstream de taille au plus égale au budget bits alloué à l'image. Par conséquent, nous proposons un nouveau modèle Rate-QP pour ajuster le débit vidéo image par image.
- Enfin, nous proposons une méthode pour réduire le temps d'encodage du codeur VVC dans le cas d'un encodage à faible débit et pour séquences vidéo de résolution inférieure à HD sans trop sacrifier l'efficacité de compression. Cet objectif est atteint en désactivant les outils de codage non efficaces pour ces cas d'usages.

Structure de la thèse

Les contributions et la structure de cette thèse sont les suivantes :

- Chapter 2 présente au lecteur les notions de base sur le codage vidéo et le streaming adaptatif HTTP.
- Chapter 3 présente un nouveau modèle de débit, noté R- (QP, D), de relation entre la taille de l'image après codage R_n, le paramètre de quantification QP_n, et la distorsion de l'erreur quadratique moyenne (MSE) D_{n-1} de l'image de référence $n-1$. Notre modèle proposé est utilisé pour déterminer les QP optimaux pour coder les images. Une partie du matériel du Chapitre 3 a été présenté dans

    - Mourad Aklouf, Marc Leny, Michel Kieffer, and Frédéric Dufaux. "Interframe-Dependent Rate-QP-Distortion Model for Video Coding and Transmission." In 2021 IEEE International Conference on Image Processing (ICIP), pp. 2019-2023. IEEE, 2021.

- Chapter 4 présente un algorithme d'adaptation de débit d'encodage, mis en œuvre du côté de l'émetteur et adapté aux applications de streaming à faible latence. Il vise à contrôler la marge de lecture du client (*i.e.*, le nombre d'images dans la mémoire tampon du client). L'algorithme proposé utilise des mesures du débit de transmission et du niveau du tampon de l'émetteur.
- Chapter 5 présente une méthode d'optimisation de type branch-and-prune pour réduire la complexité de l'encodeur VVC. Notre méthode vise à identifier un ensemble d'outils de codage à désactiver tout en satisfaisant une contrainte sur l'efficacité du codage. Le matériel du chapitre 5 a été présenté dans

    - Mourad Aklouf, Marc Leny, Frederic Dufaux, and Michel Kieffer. "Low complexity versatile video coding (VVC) for low bitrate applications." In 2019 8th European Workshop on Visual Information Processing (EUVIP), pp. 22-27. IEEE, 2019.

- Chapter 6 résume les résultats de nos recherches et suggère plusieurs sujets possibles pour de futures recherches.

État de l'art

Le chapitre 2 fournit les notions de bases du codage vidéo et du streaming adaptatif HTTP. Nous discutons également des principaux composants de la latence de bout en bout dans les systèmes de diffusion vidéo. Nous fournissons une revue de la littérature sur les algorithmes d'adaptation de débit de pointe et leurs classifications selon l'emplacement de la logique d'adaptation de débit (pilotée par le serveur ou pilotée par le client) et l'entrée utilisée pour l'adaptation (approches basées sur la bande passante, approches basées sur le niveau de remplissage des tampons ou approches hybrides).

Nous avons indiqué que dans le contexte du streaming à faible latence à partir d'un émetteur mobile, les approches pilotées par l'émetteur sont mieux adaptées pour sélectionner le débit d'encodage de la vidéo en fonction de l'état du réseau et du niveau de mémoire tampon de l'émetteur. En effet, l'émetteur mobile peut facilement estimer l'état du canal et n'a pas à attendre les rapports retardés sur l'état du réseau et de la mémoire tampon fournis par le client. De plus, une adaptation de débit au niveau de l'image est nécessaire pour obtenir une transmission à faible latence.

L'adaptation au niveau image est réalisée à l'aide d'un modèle de la relation entre la taille du flux binaire résultant du codage de l'image et le paramètre de quantification sélectionné. Par conséquent, nous rappelons plusieurs modèles de débit paramétriques de pointe utilisés pour le contrôle du débit.

Enfin, certaines caractéristiques techniques de la nouvelle norme Versatile Video Coding (VVC) et les méthodes d'optimisation de pointe du codeur VVC ont été revues.

Modèle Rate-QP-Distortion pour le streaming et la compression vidéo

Le contrôle du débit de codage vidéo repose sur un modèle de la relation entre la taille du flux binaire résultant du processus de codage et les paramètres de codage vidéo. Le modèle de débit permet de déterminer le paramètre de codage optimal pour avoir un flux binaire de taille au plus égale au budget bits alloué dans les contraintes de faible latence. Le paramètre de quantification QP est généralement considéré pour le contrôle du débit vidéo car il a un impact direct sur la taille du flux binaire résultant.

Des modèles paramétriques entre le débit de l'image et son QP ont été proposés dans la littérature. Cependant, la précision de ces modèles n'est pas fiable pour la transmission dans des canaux à bande limitée. Ces modèles ne tiennent pas compte de la dépendance temporelle entre les images.

Nous proposons le modèle suivant de la relation entre le débit d'encodage R_n de la trame n et son paramètre de quantification QP_n en fonction de la distorsion de l'erreur quadratique moyenne (MSE) D_{n-1} de l'image référentielle $n-1$:

$$R_n\left(QP_n, D_{n-1}\right) = g_1\left(QP_n\right) + g_2\left(QP_n\right)\left(\tanh\left(g_3\left(QP_n\right)\log(D_{n-1}) - g_4\left(QP_n\right)\right) + 1\right), \quad (1)$$

Avec,

$$g_1\left(QP_n\right) = p_1 \exp\left(-p_2 QP_n\right), \quad (2)$$

$$g_2\left(QP_n\right) = p_3\left(-p_4 \log\left(QP_n\right) + 1\right), \quad (3)$$

$$g_3\left(QP_n\right) = p_5 QP_n, \quad (4)$$

$$g_4\left(QP_n\right) = \left(p_6 QP_n - p_7\right)^2, \quad (5)$$

Notre modèle proposé, noté R- (QP, D), implique un vecteur de 7 paramètres $\mathbf{p} = (p_1, \ldots, p_7)$, dont la valeur dépend de l'image et doit être déterminée pour prédire avec précision R_n en fonction de QP_n et D_{n-1}.

La performance du modèle proposé pour prédire R_n en fonction de QP_n est comparée aux modèles de référence dans [4], [5] et [6]. Nous évaluons les performances des modèles dans deux scénarios de codage : codage à QP constant et codage avec QP variable dans le temps. Dans le deuxième scénario, la variation de QP est choisie pour simuler le cas d'un codage vidéo pour une transmission sur un canal de transmission instable, où le QP change toutes les quelques trames suite à une chute ou une augmentation du débit de transmission.

Dans les deux scénarios de codage, le modèle proposé surpasse les autres modèles de la littérature. Par exemple, dans la séquence vidéo *Tango*, 90% de toutes les erreurs de prédiction sont inférieures à 8,6% lors de l'utilisation de l'encodage QP constant, et 90% de toutes les erreurs de prédiction sont inférieures à 12% lors de l'utilisation de la variable Encodage QP.

Les gains sont particulièrement significatifs à des débits faibles, cet attribut montre que notre modèle est exceptionnellement fiable dans le cas du codage pour transmission dans un canal à faible débit ou lorsque des chutes soudaines se produisent.

Modèle d'adaptation prédictive du débit vidéo pour le streaming en direct à faible latence

Le chapitre propose un algorithme de contrôle du débit d'encodage adapté aux applications de streaming en direct pour des délais de 100 à 200 ms. Le contrôle est effectué au niveau de l'image, ce qui nécessite l'utilisation du modèle R- (QP, D). À l'aide de mesures du débit de transmission et du niveau de remplissage du tampon de transmetteur, une approche de type Model Predictive Control (MPC) est utilisée pour déduire le budget bits de l'image à coder pour une marge cible de lecture (target playback margin), le contrôleur peut alors sélectionner la valeur appropriée de QP pour cette image. La marge de lecture est la différence entre l'instant de fin de décodage et l'affichage de l'image chez le client.

La figure 1 illustre les composants de l'architecture proposée pour le streaming en direct. Le transmetteur se compose d'une caméra, un encodeur vidéo, un contrôleur de débit d'encodage et un tampon de transmission. Le client dispose d'un décodeur, d'un tampon de réception et d'un player.

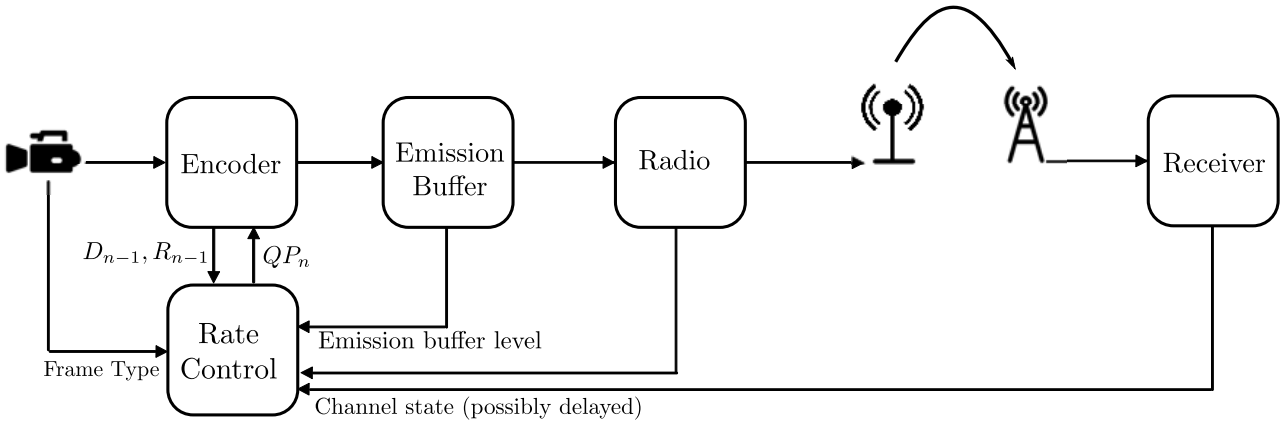

Figure 1: L'architecture proposé pour le streaming en direct pilotée par le transmetteur.

Une fois l'image n est acquise, elle est transmise à l'encodeur et compressée. Le flux résultant du codage est segmenté en paquets RTP et placé dans le tampon de transmission. Les paquets sont transmis via le réseau 4G/5G jusqu'au récepteur. Au côté client, une fois que tous les paquets liés à l'image n ont été reçus, le décodage démarre et introduit un délai de dé-

codage T_{d}. L'image est ensuite mise en mémoire tampon avant d'être affichées au temps $t_n + \Delta_{\mathsf{p}}$, où Δ_{p} est le délai d'acquisition à la lecture.

Le contrôle est effectué de manière à empêcher le tampon contenant les images décodées au niveau du client de se vider. Cela garantit que les images sont affichées à temps. Le contrôleur de débit prend en entrée la quantité de bits B_n stockés dans le tampon de transmission ainsi que l'état du canal et utilise l'algorithme l'adaptation du débit pour estimer le débit de codage cible R_n^* de l'image n. Le contrôleur gère aussi le modèle R-(QP, D) pour déterminer la valeur QP_n à partir de la distorsion D_{n-1} de l'image précédente et R_n^*.

Puisque les caractéristiques temporelles et spatiales des images évoluant avec le temps, une mise à jour en ligne des paramètres du modèle R-(QP, D) est effectuée en utilisant des essais de codage supplémentaires. Des informations complémentaires liées aux caractéristiques de l'image (type, complexité), qui peuvent impacter les paramètres du modèle R-(QP, D) peuvent également être prises en compte.

Le débit de codage cible R_{n+1}^* de l'image $n+1$ est déterminé en définissant une marge de lecture cible τ^* qui est une petite marge temporelle dans laquelle tous les paquets du l'image doiventt être reçue avant le début du décodage :

$$\begin{aligned} R_{n+1}^* &= \frac{\tau_n - \tau^*}{T_{\mathsf{f}}} C_{n+1} + \frac{B_n}{T_{\mathsf{f}}} \frac{C_{n+1}}{C_n} - \frac{B_n + (R_n - C_n)\, T_{\mathsf{f}}}{T_{\mathsf{f}}} + \frac{C_{n+1}}{C_n} R_n \\ &= \frac{\tau_n - \tau^*}{T_{\mathsf{f}}} C_{n+1} + \left(\frac{C_{n+1}}{C_n} - 1 \right) \left(\frac{B_n}{T_{\mathsf{f}}} + R_n \right) + C_n, \end{aligned} \tag{6}$$

avec T_{f} est la durée de l'image, C_n et C_{n+1} sont les débits de transmission aux instants n et $n+1$ respectivement, B_n est lle niveau de remplissage de la memoire tampon de l'émetteur, et R_n est le débits réel de l'image n.

L'évaluation de R_{n+1}^* à l'aide de (6) est effectuée côté transmetteur. La marge de lecture τ_n pour l'image n est estimée côté émetteur. On utilise également les estimations de débit de transmission $\widehat{C}_n$ et $\widehat{C}_{n+1}$. Alors, (6) devient :

$$R_{n+1}^* = \frac{\widehat{\tau}_n - \tau^*}{T_{\mathsf{f}}} \widehat{C}_{n+1} + \left(\frac{\widehat{C}_{n+1}}{\widehat{C}_n} - 1 \right) \left(\frac{B_n}{T_{\mathsf{f}}} + R_n \right) + \widehat{C}_n. \tag{7}$$

Nous considérons une configuration de simulation composée d'un serveur (transmetteur) et d'un client comme décrit ci-dessus. Le serveur reçoit les images vidéo à encoder, exécute l'encodeur x265 [7] et alimente le tampon de transmission avec les paquets des images compressées. le serveur comprend également le modèle R-(QP, D) et l'algorithme d'adaptation du débit. Le client contient un tampon de réception et un décodeur HEVC [8]. La transmission de paquets est simulé à l'aide de traces 4G extraites de [9].

L'algorithme de contrôle du débit proposé est comparé à quatre algorithmes d'adaptation de pointe Festive [10], Panda [11], BOLA [12] et BBA [13]. Pour garantir une comparaison équitable des performances, tous ces algorithmes ont été adaptés pour fonctionner côté transmetteur et pour ajuster le débite codage vidéo au niveau de l'image. Tous les algorithmes partagent le même modèle R-(QP, D). L'approche proposée surpasse ces algorithmes à la fois en termes de PSNR moyen et de pertes d'images.

Réduction de la complexité du codeur VVC

La nouvelle norme de codage vidéo VVC présente un grand avantage lorsqu'elle est utilisée dans des contraintes de faible latence. Néanmoins, la conception actuelle du codeur VVC est principalement axée sur le contenu haute résolution, et il n'est malheureusement pas adapté à l'encodage à faible résolution et à faible débit. Les nouveaux outils ajoutés entraînent une charge importante en termes de complexité de calcul.

Nous proposons une méthode d'optimisation pour les scénarios de codage à faible résolution et à faible débit. Plus précisément, nous étudions l'utilité de certains des nouveaux outils de codage. Nous montrons expérimentalement qu'une réduction significative de la complexité peut être obtenue en désactivant certains de ces outils tout en préservant l'efficacité du codage. Notre objectif est d'identifier le sous-ensemble d'outils de codage de VTM5.0 qui peuvent être désactivés avec les séquence vidéo de basse résolution et en encodage à faible débit. Dans ce but, nous présentons une méthode de type branch-and-prune pour déterminer l'ensemble d'outils de codage qui fournissent la meilleure réduction de complexité, tout en satisfaisant une contrainte sur la dégradation de BD_{rate}.

Les résultats expérimentaux montrent qu'une réduction significative de la complexité de l'encodage peut être obtenue, avec une perte négligeable de BD_{rate}. Par exemple, une réduction de complexité de 56% a été obtenue pour la séquence vidéo *Johnny* à une résolution de 384×216 en appliquant notre méthode, avec une perte de 1.88% en BD_{taux}.

De plus, nous avons pu proposer une combinaison commune d'outils à désactiver pour chaque résolution. Nos résultats expérimentaux montrent que ces outils engendrent moins de 2% BD_{rate} de perte avec 35% de réduction de complexité d'encodage en moyenne. Ce résultat est particulièrement bénéfique car nous pouvons construire des profils de codage pour chaque résolution en supprimant les outils inutiles. Par conséquent, les désactiver automatiquement et réduire la complexité du traitement pour l'encodage en temps réel.

Conclusion

Dans cette thèse, nous abordons le problème de l'adaptation du débit vidéo pour le streaming vidéo à faible latence. Notre objectif est la conception d'un algorithme d'adaptation du débit pour le streaming vidéo à faible latence à partir d'un émetteur mobile avec des délais de bout à bout entre 100 et 200 ms.

Nous présentons d'abord un nouveau modèle Rate-QP-Distortion, *i.e.* R-(QP, D). Notre modèle décrit la relation entre la taille du flux binaire R_n de l'image n, son paramètre de quantification QP_n et la distorsion MSE D_{n-1} de l'image de référence $n-1$. Le modèle proposé est avantageux lors de l'ajustement du QP de l'image en fonction d'un budget de débit cible en cas de transmission vidéo en direct à faible latence. Ce budget de débit cible est déterminé via un algorithme d'adaptation de débit vidéo.

Notre deuxième contribution est un nouvel algorithme d'adaptation du débit pour le streaming vidéo à faible latence à partir d'un émetteur mobile. L'approche proposée exploite le niveau de tampon de transmission et une estimation du débit de transmission sans fil pour déterminer le débit de codage cible de chaque image. Le choix du paramètre de quantification pour chaque image est effectué via le modèle R- (QP, D) proposé. Nous comparons les

performances de l'approche proposée avec quatre algorithmes d'adaptation de référence, à savoir Festive [10], Panda [11], BOLA [12] et BBA [13], en considérant des scénarios de streaming de latences inférieures à 200 ms. Les résultats de simulation impliquant de vraies traces de bande passante 4G ont montré que notre approche proposée surpasse les algorithmes de la littérature à la fois en termes de PSNR moyen et de nombre des images perdues.

Enfin, la diffusion en direct est sensible au délai et nécessite un encodeur capable de compresser le flux vidéo en temps réel. Par conséquent, nous proposons une méthode pour réduire la complexité du codeur VVC. Notre méthode consiste à identifier un ensemble d'outils de codage qui peuvent être désactivés tout en satisfaisant certaines contraintes sur la dégradation BD_{rate}. Une réduction de complexité allant jusqu'à 56% a été obtenue pour des séquences vidéo de résolution 384×216, 512×288 et 640×360 en appliquant notre méthode, avec une perte inférieure à 2% en BD_{taux}. De plus, nous avons pu identifier un ensemble commun d'outils de codage à désactiver dans chaque résolution. Ces outils communs peuvent être utilisés pour créer des profils de codage pour chaque résolution, réduisant ainsi la complexité du codage lors du codage à la volée.

Perspectives futures

Le travail présenté dans ce document peut être étendu dans de nombreuses directions. Voici quelques-uns d'entre eux:

- **Amélioration du modèle R-(QP, D) :** Le modèle R- (QP, D) peut être amélioré pour être compatible avec le mode de codage très faible latence où l'image n utilise deux images références déjà codées, *e.g.*, les images $n-1$ et $n-3$. Une amélioration possible pourrait être l'utilisation d'une combinaison des distorsions des images références. On doit aussi étudier les performances de l'estimation itérative des paramètres du modèle avec l'intra-refresh activé.

- **Estimation du débit de transmission :** Une direction de recherche importante est de savoir comment estimer efficacement la bande passante du canal et du réseau. L'estimation de la bande passante peut se faire

on construisant une carte de bande passante pour estimer le débit de transmission dans une position géolocalisée de l'émetteur mobile. Dans cette approche, une base de données des débits de transmission avec les positions GPS est d'abord construite hors ligne [14]. Les informations collectées sont utilisées pour estimer les conditions futures du réseau, puis le débit vidéo est ajusté en conséquence. La position de l'émetteur peut être calculée à l'aide d'une prédiction basée sur Kalman.

- **N-steps MPC Algorithm :** S'il est possible d'estimer le débit de transmission N pas à l'avenir, il serait possible d'effectuer un contrôle du débit de codage N pas à l'aide de l'algorithme MPC. Le débit cible des futures trames N peut être déterminé à l'avance. N-step MPC permet une adaptation du débit et un contrôle plus efficaces en anticipant la baisse future du débit de transmission. Il permet également de limiter les oscillations en lissant les débits cibles des images dans une fenêtre de taille N.

Chapter 1

Introduction

1.1 Context

Video streaming has become the dominant type of traffic over the internet, with more than 80% in 2021 [1], while 57% of it was attributable to the top six Over-The-Top (OTT) brands: Google, Netflix, Facebook, Apple, Amazon, and Microsoft [15]. Live streaming represents an important part of video traffic.

Low-latency video streaming has emerged as a key application in the broadcasting of sports events, video-conferencing, telepresence, or remote driving. According to the Cisco Annual Internet Report, live video grew by 93% in 2020 and will account for 17% of internet video traffic in 2022 [1]. Live streaming will become even more popular thanks to the advances in telecommunication networks and video codecs. The new generation of video codecs like the Versatile Video Coding (VVC) saves half of the bandwidth used by the High Efficiency Video Coding (HEVC) for the same quality, and the fifth-generation (5G) of mobile networks enables a new type of latency-sensitive applications that was not possible with the 4G.

The consumption of Ultra-High Definition (UHD) videos is also increasing over time. According to [16], 17% of the content in the Netflix catalog is in 4K, and 30% of Netflix subscribers have the UHD package, making it the largest 4K OTT platform in the market.

Moreover, UHD live streaming is expected to grow significantly in the next few years, especially for sports events and entertainment. Many companies have performed 4K and 8K live trials using the exiting telecommunication infrastructures and Scalable HEVC encoder. The Korean Broadcasting System

(KBS) has carried out live 4K terrestrial broadcasting of major sports events such as the 2014 FIFA World Cup [17]. BT Sport and Samsung recently presented the first live 8K sports broadcast for the Tokyo Olympic Games [18].

Many streaming methods leverage HTTP Adaptive Streaming (HAS) [2]. With HAS, video clips are divided into segments of a few seconds, encoded at several bitrates, and stored on HTTP media servers. Using HAS, a client can request video segments with suitable encoding rates. Nevertheless, HAS does not specify a rate adaptation logic. In downlink streaming applications, adaptation algorithms are usually implemented at the client to select the optimal bitrate of the requested segments. These algorithms exploit instantaneous or averaged measurements of the network and channel characteristics and/or of the client buffer level. They aim to maximize the Quality of Experience (QoE) of the client. This is usually obtained by maximizing the quality of the received video while minimizing the number of freezes and switches of the video quality.

The efficiency of the bitrate adaptation algorithm has a direct impact on the client QoE. If the segment bitrate is incorrectly selected or not optimal following a change in network state, the downloading of the concerned segment may take an additional delay, affecting the end-to-end latency of the session. Hence, the proper functioning of the bitrate adaptation algorithm depends on the accuracy of the available bandwidth and the client buffer level estimation.

In live video streaming, video acquisition, encoding, and transmission are performed in real-time. Minimizing the delay between the acquisition of a frame and its display at the client (glass-to-glass delay [19]) is an additional QoE requirement. Live streaming is significantly more challenging than classical Video-On-Demand (VOD) streaming. To mitigate bandwidth variations between the server and the client, large buffers are usually implemented at the client. These buffers induce a significant delay in the live streaming context, and their size has thus to be minimized. Client-level rate control algorithms based on HAS may entail large delays as control decisions are taken at the same period as the video segment duration. When the bandwidth between the server and the client varies with a smaller time scale, wrong decisions may induce significant segment download delays when the bandwidth is less

than expected. Traditional streaming protocols such as Real-time Messaging Protocols (RTMP) [20] or Real-time Streaming Protocols (RTSP) [21] associated with Real-time Transport Protocol (RTP) [22] store and process a few milliseconds of video in each packet, making it possible to achieve a low-latency transmission of order 300 milliseconds [23].

HTTP-based streaming protocols have been designed to be scalable and fault-tolerant through pull-based bitrate adaption schemes. In case of a failed downloading of the video segment, the client can request it again from the same or different media server. The segment bitrate is selected according to client network conditions. This approach is based on the assumption that the client network conditions are unstable or continuously varying while the media server is located in a safe location with guaranteed access to the network. Whereas this assumption is valid for VOD streaming, where the client usually requests video content already encoded and saved on Content delivery network (CDN) servers, the client-driven approach is not always optimal for low latency live streaming.

In several applications such as live video streaming from a car during a race or remote control of a drone, the camera acquiring the scene transmits its compressed stream over a 4G/5G wireless access network to the client via the wired part of the network. Mobility induces significant and fast variations of the wireless channel characteristics. In such a context, transmitter-side control approaches appear better suited for selecting the video encoding parameters adapted to the wireless channel and network characteristics. This also prevents waiting for delayed reports of the network and buffer states provided by the client using, *e.g.*, the RTCP protocol. transmitter-side control allows a much finer adaptation granularity, which is necessary to reduce the size of reception buffers at the client and achieve low delay.

As previously stated, the new generation of video codecs has a significant advantage in variable network conditions, particularly in low latency streaming. For instance, the Versatile Video Coding (VVC) achieves 50% compression gain compared to its predecessor, The High-Efficiency Video Coding (HEVC), at the same PSNR quality.

VVC is mainly focused on Ultra-High Definition (UHD) video content. A variety of encoding tools were used to achieve high coding efficiency. However,

these new coding tools entail a significant amount of computational complexity. The current version of the VVC encoder (VTM5.0) has 10x the runtime execution of the HEVC encoder (HM16) [3]. Therefore, The VVC encoder is unsuited for real-time application, even for low-bitrate encoding and low-resolution contents that are still used for video streaming today (*e.g.*, 480p and 360p).

In low latency applications, the video stream is encoded on the fly from a source that captures the video in real-time. The compressed video is sent immediately to the packetizer, where a streaming protocol such as MPEG-DASH divides the compressed video into segments before transmitting it. Hence, the video encoder must finish encoding the video frame before the next available frame at the source.

1.2 Objectives

In summary, the objectives of this dissertation are:

- Address the challenges of low latency video streaming from mobile transmitter in variable network conditions.
- Propose a server-driven bitrate adaptation algorithm for low latency video streaming from a mobile transmitter. The proposed algorithm must achieve a small bitrate control granularity and adjusts the video bitrate according to the up-link channel state and transmitter buffer level.
- Improve the overall QoE of the client by maximizing the average PSNR quality of the received video while minimizing the frame loss and the variations of the video quality, and decreasing the initial playback delay of the session, which is the time difference between capturing the video frame and displaying it at the client-side.
- Propose Rate-QP model to adjust the video bitrate at the frame level. The bitrate of the frame is adjusted by the mean of a rate model that determines the quantization parameter for having a bitstream of size at most equal to the allocated bits budget of the frame. The accuracy and the well-tuning of the model parameters for each frame are critical to satisfying the bit budget constraint of the frame.

- Optimize the VVC encoder by reducing the run time execution in the case of low bitrate encoding and for video resolutions lower than HD without sacrificing much of its compression efficiency. This goal is accomplished by removing less efficient coding tools for these use cases; therefore, significant gains in computational complexity can be achieved for a slight decrease in coding gain.

1.3 Contributions and structure of the thesis

Contributions and the structure of this thesis are as follows:

- Chapter 2 introduces the reader to the necessary background on video coding and HTTP adaptive streaming. It reviews the state of the art bitrate adaptation algorithms, the parametric models used for video rate control, and lastly, some optimization methods for VVC encoder.
- Chapter 3 presents a new rate model denoted R-(QP, D), of the relation between the bitstream size R_n of frame n, the quantization parameter QP_n, and the Mean Square Error (MSE) distortion $D_{n-1}$of the reference frame $n-1$. Our proposed model is used to determine the optimal QPs for encoding the frames. We also present a real-time iterative estimation approach of the model parameters. Part of the material in Chapter 3 has been presented in
    - Mourad Aklouf, Marc Leny, Michel Kieffer, and Frédéric Dufaux. "Interframe-Dependent Rate-QP-Distortion Model for Video Coding and Transmission." In 2021 IEEE International Conference on Image Processing (ICIP), pp. 2019-2023. IEEE, 2021.
- Chapter 4 presents a transmitter-side encoding bitrate adaptation algorithm adapted for low-latency streaming applications. It aims to control the client playback margin (*i.e.*, the number of frames in the client buffer). The control is performed at the frame level. Using measurements of the up-link channel and transmitter buffer level, a Model Predictive Control (MPC) framework is employed to infer the bits budget (encoding bitrate) of the frame to be transmitted.

- Chapter 5 presents a branch-and-prune optimization method for VVC encoder that aims to identify a set of coding tools which may be disabled while satisfying a constraint on the coding efficiency. The material in Chapter 5 has been presented in
    - Mourad Aklouf, Marc Leny, Frederic Dufaux, and Michel Kieffer. "Low complexity versatile video coding (VVC) for low bitrate applications." In 2019 8th European Workshop on Visual Information Processing (EUVIP), pp. 22-27. IEEE, 2019.
- Chapter 6 summarizes the results of our research and suggests several possible topics for future research.

Chapter 2

Background and State-of-the-art

This chapter overviews several concepts and technical tools used throughout this dissertation. Section 2.1.1 reviews the technical features of the new Versatile Video Coding standard (VVC), then we run through the state of the art techniques proposed to reduce the complexity of VVC encoder. In Section 2.1.4, we recall several parametric models used for video bitrate control, as well as the difference between regular rate models and inter-frame dependent rate models.

Section 2.2.1 the HTTP Adaptive Streaming (HAS) and MPEG-Dynamic Adaptive Streaming over HTTP (DASH). We also identify the primary delay components of HAS architecture and explain why it is not well suited for low latency video transmission. Section 2.2.2 presents a point-to-point video communication model proposed by Bachhuber *et al.* [19] and discusses the different delay components in point-to-point video streaming. Finally, Section 2.2.3 surveys the different adaptation bitrate algorithms presented in the literature, their different classes and granularity of adaptation.

2.1 Background on video coding

The HEVC video coding standard [24] achieves 50% compression gain at the same quality compared to its predecessor H.264/AVC. Yet, HEVC compression efficiency is no longer sufficient to address the growing demands on ultra-high definition (UHD). A new video coding standard, Versatile Video Coding (VVC) [25], has been developed to bring further compression gain over HEVC. VVC reference encoder (VTM) achieves nearly 50% of compression gain at the

same quality as HEVC reference software (HM) [26]. However, this comes with a cost of an increase in encoder runtime which is ten times that of the HM [3].

The VVC encoder has a significant advantage when used in variable network conditions and low latency constraints, as it saves half of the channel capacity used by the HEVC encoder. However, the VVC encoder is not optimized and thus not well-suited for real-time applications where the encoding of each frame should take no longer than the frame acquisition time.

Video encoder optimization is challenging because the standards do not describe how to build the VVC encoder as a final optimized product. It is up to developers to use different tools to achieve this goal and find a compromise between execution time and compression efficiency. VVC test models are developed only to show the performance in terms of compression gain and do not take into account the complexity constraint.

The following section provides a background VVC encoding process and the state-of-the-art optimization methods proposed for the VVC encoder after finalizing the standard in July 2020 [26].

2.1.1 The Versatile Video Coding (VVC) standard

The Versatile Video Coding (VVC) standard is the most recent video coding standard developed by the Joint Video Experts Team (JVET) of the Moving Picture Experts Group (MPEG/ISO) and the Video Coding Experts Group (VCEG/ITU-T). The project started in 2015 when JVET issued a Call for Proposals (CfP), aiming to achieve 50% encoding efficiency in bitrate for the same quality picture of the HEVC standard [27]. Strong emphasis has been placed on Ultra-High Definition (UHD) video content, including 3840x2160 (4K) and 7680x4320 (8K) formats [28], that became mainstream nowadays. More than 30 companies and institutions from all around the globe have contributed to the design of the new video standard. Initial proposals were grouped in a software known as JVET Exploration Model (JEM) but it was quickly replaced by the enhanced software VVC Test Model (VTM). The final draft of VVC was issued in July 2020 after more than three years of development.

VVC uses the same block-based hybrid video coding scheme of its predecessors, the High-Efficiency Video Coding (HEVC) [24] and the Advanced Video Coding (AVC) [29] standards. VVC was designed starting from the HEVC by

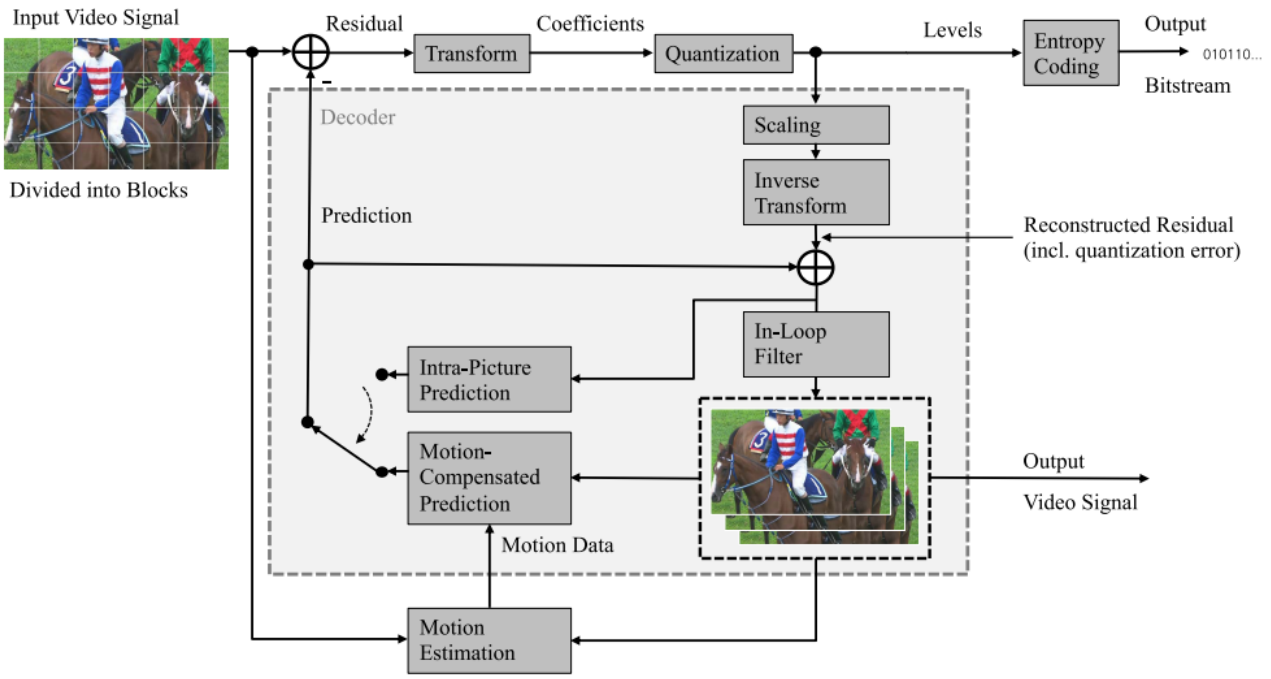

Figure 2.1: Block diagram of a hybrid video encoder, including the modeling of the decoder within the encoder. This image is reused from [26].

enhancing its different coding modules and inserting new coding tools to increase the coding efficiency. The new codec is not destined only for Standard Dynamic Range (SDR) video content, but also includes state-of-the-art coding tools for different video contents, such as High Dynamic Range (HDR), 360° video, and computer-generated content. VVC is thus a versatile video coding codec that can address a variety of use cases.

The high-level functionalities, and a detailed description of the new features can be found in [30]. In what follows, we give a brief overview of the coding process and we focus on the essential coding tools integrated into the VVC.

Figure 2.1 shows the diagram of the classical hybrid (*i.e.*, inter/intra-coding) scheme of the VVC codec. The represented video content consists of either one color plane (*i.e.,* luma Y) or three-color planes (*i.e.,* one luma Y, and two chroma components Cb and Cr) of sample values with represented bit depth. VVC can handle a video bit depth of 8 bits and 10 bits. The chroma sampling of the input video sequences can be 4:2:0, 4:2:2, or 4:4:4 in which the chroma planes have the same width and height as the luma plane.

Video frames are subsequently coded in a specific order according to the initial coding configuration, *e.g.*, Low Delay (LD), Random Access (RA), or All-Intra (AI) [31]. Each frame is first split into large Coding Tree Units (CTUs) of size 128×128 luma samples. These CTUs are considered as the primary processing unit and are iteratively fragmented into Coding Units (CUs). A CU is then used for prediction and transform coding. The new partitioning tool in VVC extends the regular quadtree partitioning of HEVC by enabling a binary and ternary splitting of the CUs. Accordingly, CUs can be of non-square shapes. The new binary and ternary splitting types enable more flexible partitioning and allow a better adaptation to the spatial properties of the frame. The new partitioning module provides the highest coding efficiency among the newly added tools by up to 12% in UHD sequence [32].

The encoder performs an exhaustive search process, known as Rate-Distortion Optimization (RDO), testing all possible combinations of CTU splitting structures, intra-prediction modes, and transforms. The RDO process minimizes the cost J, defined as $J = D + \lambda \times R$, where D is the distortion of the CU, R is its bitrate, and λ the Lagrangian weighting factor that depends on the quantization parameter (QP). At the end of the partitioning process, a CU is either inter or intra-predicted.

In addition to DC, planar, and the 33 angular modes of the HEVC, advanced intra-picture prediction techniques have been adopted in VVC. First, the angular modes are increased to 65 for finer and more accurate spatial prediction, and 28 wide angles are used for non-square blocks. Furthermore, new coding modes have been included to increase the intra-coding efficiency. For instance, the matrix-based prediction is a low-complexity neural-network-based intra-prediction. The cross-component prediction predicts chroma samples from luma samples, and the Position-Dependent Prediction Combination refines the samples of specific prediction modes.

Motion compensation in VVC is enhanced beyond that of the HEVC thanks to various new inter-coding tools. Like HEVC, inter-coded CUs may have one or two Motion Vectors (MVs). Still, VVC introduces advanced methods to code the MVs. *e.g.*, History-Based MV Prediction and Symmetric MV Difference. Besides, both motion compensation and refinement processes can be performed on the subblock level. A detailed review of the adopted techniques

can be found in [26] and [30].

VVC conducts a transform and quantization step to the prediction residuals of the inter/intra-predicted CU. The energy compaction of VVC is further enhanced using large transform blocks of 64×64 samples and new transform methods and matrices are introduced. The Context Adaptive Binary Arithmetic Coding (CABAC) algorithm encodes these samples and inserts them into the output bitstream. For more information about The CABAC engine of the VVC and the binarization process of the transform coefficients, refer to [33].

Finally, many filters are applied to the reconstructed blocks before using them as an output signal (decoded frame) and as references for the subsequent motion-compensated blocks. This step is known as In-loop filtering. VVC uses a new luma mapping tool with chroma scaling, applied before the other filters of the In-loop filtering module. Then, the reconstructed blocks are enhanced, and blocking artifacts are reduced by applying the deblocking filter and the new Adaptive Loop Filter, respectively. Like HEVC, the Sample Adaptive Offset filter is the last filter to use before outputting the reconstructed frame.

2.1.2 Video encoder evaluation metrics

To compare the coding gain of two video encoders or two coding configurations of the same video encoder, the *Bjontegaard Delta Rate* (BD_{rate}) or the *Bjontegaard Delta PSNR* (BD_{PSNR}) are commonly adopted. A coding configuration is usually defined by a set of enabled coding tools. To properly evaluate the performance of a set of coding tool, several target values of the rate (in Kbps) have to be evaluated, which lead to associated values of distortion D (typically measured using the weighted average *PSNR* of the three components *Y*, *U*, and *V* [34]) and complexity C (approximated by the run-time, measured in seconds).

In our work, we use the *Bjontegaard Delta Rate* (BD_{rate}) [35] to evaluate the coding gain of a set $\mathcal{P}_1$ compared to another set $\mathcal{P}_2$. Assume that a video $v \in \mathcal{V}$ is encoded considering the set of tools $\mathcal{P}_1$ and $\mathcal{P}_2$, and with n_{DR} distinct QP values $\text{QP}^{(i)}$, $i = 1, \ldots, n_{\text{DR}}$, with $n_{\text{DR}} \geq 4$. This leads to as many values

$\left(R_j^{(i)}, D_j^{(i)}, C_j^{(i)}\right)$, $i = 1, \ldots, n_{\text{DR}}$, $j = 1, 2$ of the triple (R, D, C). To evaluate the BD_{rate} between decoded videos when the coding tools are are taken from $\mathcal{P}_1$ and $\mathcal{P}_2$, one first fits two polynomial models

$$r_j(D) = a_{j,3}D^3 + a_{j,2}D^2 + a_{j,1}D + a_{j,0}, j = 1, 2, \tag{2.1}$$

using $r_j^{(i)} = \log(R_j^{(i)})$ and $D_j^{(i)}$, $i = 1, \ldots, n_{\text{DR}}$. Then, using these models,

$$\text{BD}_{\text{rate}}\left(v, \mathcal{P}_1, \mathcal{P}_2\right) = 10^{\frac{1}{D_h - D_\ell}\int_{D_\ell}^{D_h}(r_2(D) - r_1(D))dD} - 1, \tag{2.2}$$

where $\text{D}_h = \min_{j=1,2}\max_{i=1,\ldots,n_{\text{DR}}} D_j^{(i)}$ and $\text{D}_\ell = \max_{j=1,2}\min_{i=1,\ldots,n_{\text{DR}}} D_j^{(i)}$.

The relative coding complexity of a set $\mathcal{P}_1$ compared to another set $\mathcal{P}_2$ is evaluated as follows:

$$\Delta C\left(v, \mathcal{P}_1, \mathcal{P}_2\right) = \frac{1}{n_{\text{DR}}}\sum_{i=1}^{n_{\text{DR}}}\left(\frac{C_2^{(i)} - C_1^{(i)}}{C_2^{(i)}}\right), \tag{2.3}$$

2.1.3 VVC Encoder Optimization

A few optimization proposals for the VVC encoder have been already proposed before finalizing the standardization process in the third quarter of 2020. The first initiatives were released in 2019 and are implemented on non-final versions of the VTM. In the literature, we often find methods that target the image partitioning process and the intra-prediction. In this section, we review some propositions to optimize the VVC encoder.

Tissier *et al.* [36] demonstrated that the encoding complexity of VTM3.0 is proportional to video resolution and QP, *i.e.*, encoding high-resolution frames with small QPs, generates the most significant complexity. In addition, they showed that the partitioning tool generates 97% of the complexity in AI configuration mode. The respective complexity contribution for Intra-mode prediction and Transform coding in VTM3.0 are 65% and 55%. However, the transform module of VTM3.0 contains only one tool (the Enhanced Multiple Transform (EMT)). Most intra-coding tools were not added yet in this version.

Pakdaman *et al.* [37] conducted an extensive complexity analysis of the VVC Test Model 6.0 (VTM6.0). The complexity of both the encoder and decoder, as well as their bandwidth consumption, was reported for six video

sequences and tree resolutions including 720p, 1080p, and 2160p. The results are compared with the latest HEVC software (HM16). The complexity of the VVC encoder is 1.5 times that of HEVC in Low Delay (LD), and 31 times in All-Intra (AI) profile. The decoder complexity is 5 times that of HEVC in LD and 1.8 times in AI. The authors observed that a significant portion of the complexity is generated by three coding modules: Intra-coding tools, Motion Estimation tools, and the Transform tools. Decoder complexity is mainly due to Motion Compensation, In-loop Filtering, and Inverse-Transform / Inverse-Quantization. Finally, the analysis reported that VVC encoding and decoding use 30 times and 3 times more memory bandwidth than that of the HEVC, which means VVC needs huge memory access optimization.

An in-depth complexity analysis of the intra-coding tools in the VTM7.0 was conducted by Saldanha *et al.* [38]. The complexity of each intra-tool was reported in the case of AI configuration. The authors found that the Multi-Type Tree (MTT), *i.e.*, the binary and ternary splitting of the CUs, is responsible for about 90% of the complexity. In addition, the luma component generates about 85% of the complexity. It is also reported that the Rough Mode Dictionary, *i.e.*, the process of speeding up the selection of intra-mode in a CU, and the Transform/Quantization represent 70 to 80% of coding complexity.

Tianyi Li *et al.* [39] proposed a deep learning approach to predict CUs partition using a Convolutional Neural Network (CNN) instead of the brute-force search of the RDO process. The proposed multi-stage CNN reduces the VVC encoding time in intra-coded slices by up to 67%, with BD_{rate} loss of less than 3.2%. The proposed CNN was trained with a large-scale database of CU partitions of 8000 frames coded in intra-mode using VTM7.0.

Similar to [39], Tissier *et al*. [40] presented a deep learning approach to reduce the complexity of Quadtree Multi-Type-tree (QTMT) search in VTM6.1. The proposed CNN is used to analyze the texture of 64×64 luma CUs and outputs a probabilities vector for all 4×4 blocks inside the CUs. The probabilities vector contains the splitting probabilities of the right and bottom boundaries in 4×4 blocks. A probability of the binary, ternary, and Quadtree splitting is calculated using the output vector, and a decision is made accordingly. The advantage of this method is the small execution time compared to the approach in [39]. The proposed CNN is tested with multiple video sequences

of high resolutions. The proposed solution reduced the complexity of the encoder in IA configuration by 42.2%, with a slight BD_{rate} increase of 0.75%.

Finally, Brandenburg *et al.* [41] established a new optimized VVC software, the VVC Optimized Encoder (VOE), by enabling only a subset of the VVC coding tools and adding some algorithmic optimizations to it. For instance, in the partitioning module of VTM7.0, the authors redesigned some of the partitioning rules to reduce redundancy of the QTMT search, thus speeding up the RDO process. The other optimized tools are Affine Motion Compensation, Adaptive Motion Vector Resolution (AMVR), Merge with Motion Vector Differences (MMVD), Symmetric Motion Vector Difference (SMVD), and the Motion Estimation. The subset of the enabled tools is chosen after analyzing their complexity and coding efficiency similarly to our proposed method in Chapter 5. Additionally, VOE has a Single Instruction Multiple Data (SIMD) implementation of some coding modules such as forward and inverse transform, in-loop filtering, and block interpolation filtering. The overall optimizations provide a complexity reduction of 39% with a compression efficiency of 30.35% compared to HEVC encoder.

All the previous approaches make algorithmic modifications to the non-normative part of the encoding process of the VVC. Our proposed method, presented in Chapter 5, aims to identify coding tools which can be ignored in low-bitrate use cases, with a greatly reduced complexity and a preserved coding efficiency. This could lead to the definition of application-oriented profiles, where some tools are automatically disabled in the high-level syntax.

2.1.4 Rate model for video coding and transmission

The control of the encoding rate is crucial in low latency streaming. The bitstream size resulting from the encoding process must be regulated according to the channel transmission rate, encoder/decoder buffer sizes, and the constant end-to-end delay. Video rate control can be performed at the macroblock level, the frame level, or on a group of pictures (GOP). It relies on a model of the relation between the size of the bitstream resulting from the encoding process, *i.e*, frame encoding rate, and the video encoding parameters. Nevertheless, the characteristics of the video and the inter-frame dependencies increase the difficulty of constructing the model and decrease it

precision.

Rate models can be classified into two categories. The first category assumes that the coding units (frame or CUs) are independent of each others, whereas the second category considers the temporal dependencies among the coding units.

Ding *et al.* [42] suggest that the rate R of a frame and the quantization step size Q curves may be modeled by the R-Q [1] model

$$R_k = p_1 + \frac{p_2}{Q_k^{p_3}}, \tag{2.4}$$

nwhere p_1, p_2 and p_3 are model parameters, and Q is the quantization step size. The relation between Q_k and the quantization parameter QP_k is defined by

$$Q_k = 2^{\frac{QP_k - 4}{6}}. \tag{2.5}$$

The model in [42] has been refined in [4] to predict the encoding rate of the H.265/HEVC frames and CTUs, accounting for the Mean Absolute Difference (*MAD*) between the original $p_k(i,j)$ and the reconstructed k-th coding unit $\widehat{p}_k(i,j)$ of size $M \times N$,

$$R_k = M \cdot N \cdot MAD_k \cdot \left(\frac{p_1}{Q_k^2} + \frac{p_2}{Q_k} \right), \tag{2.6}$$

with

$$MAD_k = \sum_{i=1}^{M} \sum_{j=1}^{N} |\widehat{p}_k(i,j) - p_k(i,j)|. \tag{2.7}$$

The MAD of CTU k is predicted using that of the CTU $k-1$:

$$MAD_k = p_3 MAD_{k-1} + p_4. \tag{2.8}$$

In the same spirit, [43] and [5] present a model involving the Sum of Absolute Difference (SAD) to describe the rate of the k-th frame

$$R_k = p_1 \frac{SAD_k}{Q_k} + p_2, \tag{2.9}$$

[1] This notation covers both R-Q and R-QP models, as there is a direct relationship between the quantization step size Q and QP

where

$$SAD_k = M \cdot N \cdot MAD_k. \tag{2.10}$$

Additionally, the SAD_k is inferred as

$$SAD_k = SAD^{\text{org}}_{k,k-1} + p_3\sqrt{D_{k-1}} + p_4, \tag{2.11}$$

where $SAD^{\text{org}}_{k,k-1}$ denotes the SAD between the original frame k and the original reference frame $k-1$. D_{k-1} is the MSE distortion of the reference frame after encoding,

$$D_{k-1} = \frac{1}{M \cdot N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\widehat{p}_{k-1}\left(i,j\right) - p_{k-1}\left(i,j\right)\right)^2.$$

Both models 2.6 in [4] and 2.9 in [5] involve thus 4 parameters, p_1, p_2, p_3 and p_4. These rate models only account for the quality of the reference CTU $k-1$ via its SAD or its MAD. This bring us to the second category of R-Q models [6], where the temporal dependency between the H.264/AVC macroblocks is more explicitly taken into account to get the model

$$R_k = p_1 \cdot M \cdot N \frac{\sigma_k^2}{Q_k^2}, \tag{2.12}$$

where σ_k is the standard deviation of the motion-compensated residual of the k-th macroblock. This model has a single parameter. All above models have been designed to adjust the QP on the block level, but they may be extended at the whole frame level.

The precision of these models are not very accurate when considering transmission in band-limited channels. First, the models 2.6 in [4] and 2.9 in [5] do not consider the temporal dependency between the frames. Our preliminary tests have shown that, for a given QP_n, the bitstream size of the encoded frame n is much more significant when its reference frame is encoded at a low bit rate compared to a reference frame encoded at a high bit rate. Moreover, these two models and particularly model 2.12, have a small number of parameters that are not sufficient to describe the frame bitrate variation adequately.

Lin and Ortega [44] have proposed a piecewise cubic model of the relation between R_k of frame k, QP_k and the QP of the reference frame.

$$R_k\left(Q_\text{I}, Q_k\right) = \begin{cases} R_k\left(x_1, Q_k\right) & Q_\text{I} \leq x_1 \\ \frac{R_k(x_1,Q_k)\times(D_I(x_2)-D_I(Q_\text{I}))+R_k(x_2,Q_k)\times(D_I(Q_\text{I})-D_I(x_1))}{D_I(x_1)-D_I(x_2)} & x_1 < Q_\text{I} \leq x_2 \\ R_k\left(x_2, Q_k\right) & Q_\text{I} > x_2, \end{cases}$$

where Q_k is the quantization scale of the current P-frame, Q_I and D_I are respectively the quantization scale and the MSE distortion of the last coded I-frame. The two pairs $(x_2, R_k\left(x_2, Q_k\right))$ and $(x_1, R_k\left(x_1, Q_k\right))$, must be determined for each P frame and for given Q_I. Zhang *et al.* [45] experimentally show that the piecewise cubic model is inaccurate in layered coded video sequences.

Two other types of rate models have been proposed in the literature: the R-ρ model [46] and the R-λ model [47]. These rate models use encoding parameters that are evaluated within the encoder during the encoding process.

The R-ρ model [46] predicts the rate R of the H.264/AVC macroblocks using the percentage ρ_k of zero-valued transformed coefficients

$$R_k = p_1 \cdot \left(1 - \rho_k\right), \tag{2.13}$$

where p_1 is the model parameter. The R-ρ model accurate enough for H.264/AVC. However, the introduction of the skip-transform mode in the H.265/HEVC standard and the new entropy coding techniques at the level of the CABAC make the relation between ρ and the rate of the coding block nonlinear. Accordingly, this kind of model is unsuited for the HEVC encoder.

Finally, the R-λ model [47] predicts the rate R of the H.265/HEVC coding units using the Lagrangian multiplier λ for the HEVC rate-distortion optimization (RDO). The relation between the rate R and λ is defined as

$$\lambda_k = p_1 \cdot \left(\frac{R_k}{M \cdot N}\right)^{p_2}, \tag{2.14}$$

with

$$QP_k = p_3 \cdot \ln\left(\lambda_k\right) + p_4. \tag{2.15}$$

Consequently, p_1, p_2, p_3, and p_4 are model parameters that need to be estimated for each coding block. The Lagrangian multiplier λ_k defines the following RDO cost function

$$J = \min\left(D_k + \lambda_k \cdot R_k\right). \tag{2.16}$$

λ is determined from 2.15 using QP_k the quantization parameter of the k-th CU. Then, the splitting type and the prediction mode that minimize 2.16 are determined for the k-th CU. The R-λ model improves the coding efficiency of the HEVC reference software HM10 by 15.9 in Low Delay configuration and 24.6 in Random Access [48]. The efficiency of the R-λ model decreases when it is applied at the frame level as the temporal dependence between the frames is not considered.

Since R-ρ and R-λ models describe the rate using low-level encoding parameters, and they do not provide a straightforward mean to control the encoder behavior as the R-Q models do, when using quantization parameters QP_n of the frame, and D_{n-1} the distortion of the reference image. The R-Q models allow easy control of the encoder without putting an additional delay to the transmission chain.

In Chapter 3, we propose a model of the relation between R_n and QP_n depending on the Mean Square Error (MSE) distortion D_{n-1} for the reference frame $n-1$. Our model, denoted R-(QP, D), is used to determine the optimal QP for encoding a frame considering some target bit budget.

2.2 Background on video streaming

Adaptive video streaming over HTTP is a widely adopted mechanism for video delivery. It offers significant advantages in variable network conditions and a bitrate adaptation that maximize the client QoE. The problem of video bitrate adaptation with HAS has been well-discussed in the literature. Many algorithms have been proposed with different adaptation logics. However, most of the work focused on the Video On Demand (VOD) service, which has less stringent latency constraints compare to live streaming. In the latter case, minimizing the end-to-end delay is an additional QoE requirement.

As a first step towards understanding and designing a reliable low latency video delivery mechanism, we recall the general scheme of the HTTP Adaptive

Streaming (HAS), and we identify the primary delay causes that make HAS unsuited for low latency video transmission. Section 2.2.2 discusses the point-to-point (P2P) video communication model proposed by Bachhuber *et al.* [19] and its delay components. Identifying the various transmission modules and their contribution to the glass-to-glass delay allows the formulation of our proposed bitrate adaptation method in Chapter 4 and so as to minimize the glass-to-glass latency.

Lastly, Section 2.1.4 surveys the different adaptation bitrate algorithms presented in the literature, including throughput-based, buffer-based, and hybrid schemes. We also explain the advantage of the server-based approach and reducing the adaptation granularity for live streaming.

2.2.1 HTTP Adaptive Streaming overview

HTTP Adaptive Streaming (HAS) [2] is a set of protocols that enables the transmission of multimedia content over the Internet. Unlike in traditional transmission protocols, *e.g.,* RTP and RTSP, multimedia content is retrieved via HTTP protocol from conventional HTTP servers. Hence, HTTP Adaptive Streaming is more efficient in large-scale networks as HTTP protocol is largely deployed over the Internet.

Apple HTTP Live Streaming (HLS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH) are the most popular HAS protocols. Despite the standardization and application differences of the two protocols, their operating scheme remains similar [2]. In what follows, we only review MPEG-DASH as an example of HAS scheme.

Figure 2.2 shows the typical HTTP adaptive streaming architecture. The media content is generally generated offline [49] and divided into multiple segments of 2 to 10s; each contains one or more media components, *i.e.*, audio, video, subtitles. The media segments are available in various representations, which are defined by a bitrate, a frame rate and/or resolution. The segments are then stored in HTTP media servers or caches servers along with the Media Presentation Description files (MPD).

HTTP-based Content Distribution Networks (CDNs) are usually deployed to handle the large number of connections from the HTTP clients. Thus, reducing the load on the origin media server and reducing the downloading

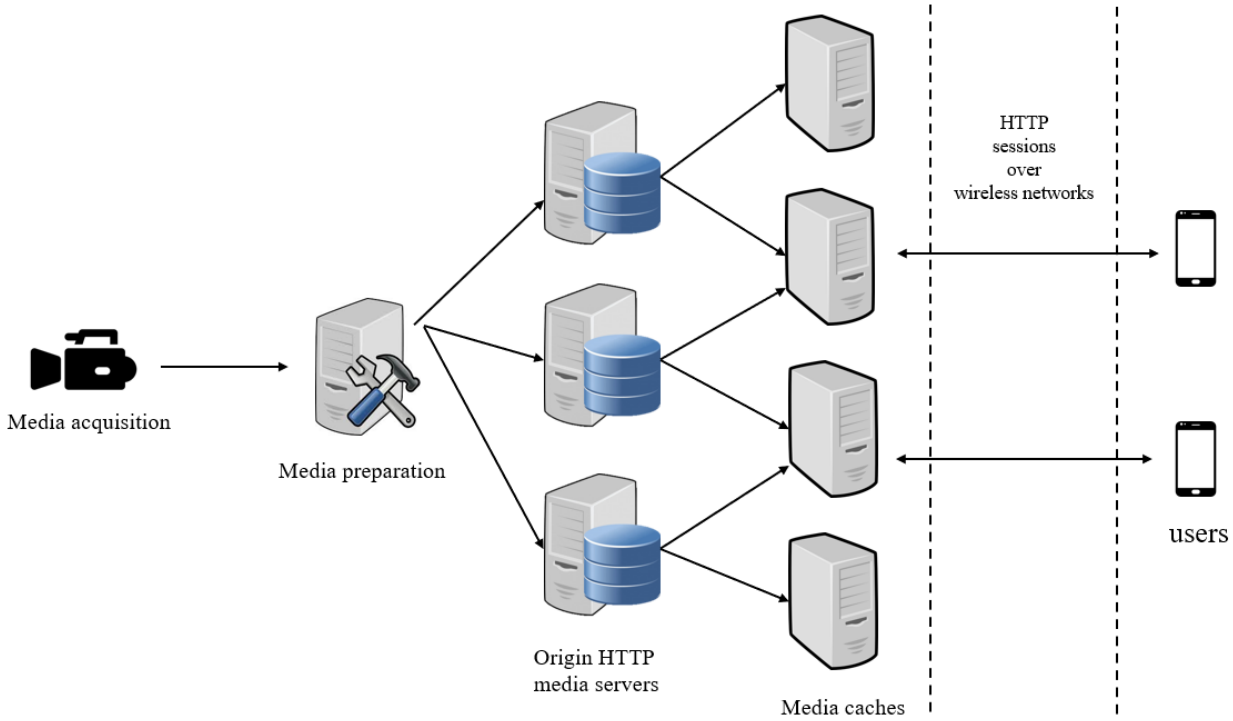

Figure 2.2: HTTP adaptive streaming architecture [51].

latency [50, 2].

The MDP is an XML file that contains the URLs and the timing information used by the client to request a media segments of particular media content [51, 50]. The MPD file is subdivided into three components :

1. Periods are large sequential pieces of the media content.

2. Inside each Period, there are representations, which are different encoding and/or sub-sampling of the same media period.

3. The representations, in their turn, contain a series of segments that can be requested by a unique HTTP URLs.

A representation consists of one initialization segment and one or more media segments. The initialization segment provides the necessary metadata to decode and play the media content.

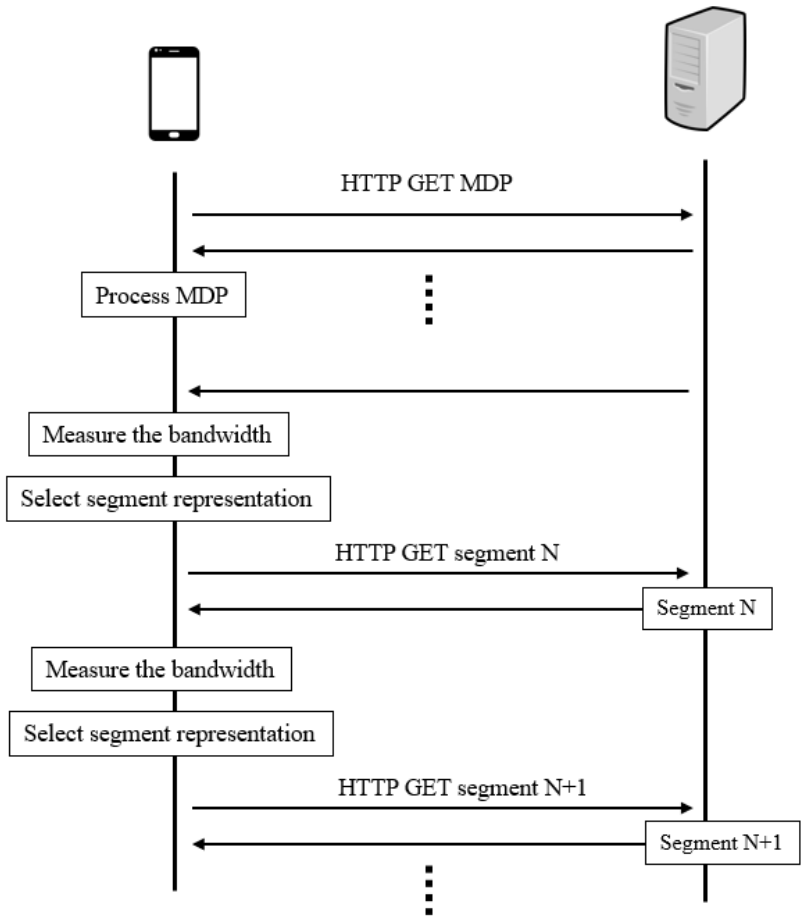

Figure 2.3: Connection establishment and media segments retrieval in HTTP adaptive streaming [52].

Figure 2.3 illustrates the segment retrieval procedure in a given HAS instance [53]. The client sends an HTTP GET to request a media segment using the information from the MDP file. The appropriate representation is selected based on the available network resources of the client using a rate adaptation algorithm. This procedure takes place before requesting a new media segment. Section 2.2.3 reviews the various bitrate adaptation algorithms presented in the literature.

HAS can also be used to stream live contents, *e.g.,* when broadcasting live sports events. In that case, the media segments are generated on the fly from a continuous video stream. Unlike in Video on demand (VOD), minimizing the end-to-end latency is crucial in live streaming. Yet, a few seconds of delay is

unavoidable in HAS due to the media segments preparation and transmission.

Lohmar *et al.* [52] introduce the main components contributing in the end-to-end delay of a DASH session:

1. Content acquisition delay T_{acq} is usually constant and depends on the acquisition device.

2. Segmentation delay T_{seg}: once the acquisition device generates enough media data, the server builds the segments by putting the media into packets and adding the metadata that describes the segment. Hence, the server must buffer an amount of data equivalent to at least one segment, which leads to a minimal delay of the segment duration Δ.

3. Asynchronous fetch of media segments T_{af}: the server does not notify the client when a new media segment is ready to be transmitted. Thus, the client asynchronously makes an HTTP GET request when it is needed. However, to avoid unsuccessful fetch of the segment, the client needs to request it T_{af} after the availability time of the segment. This leads to a delay of one segment duration Δ in the worst case.

4. Download Time T_{ch}: the download time of the segment initially depends on the channel state and the available downloading rate. In the worse case, the download time may be higher than the segment duration $T_{\text{ch}} = \Delta + T_{\text{link}}$, with T_{link} is the propagation time of the packets in the wireless and the physical links.

5. Buffering at client-side T_{b}: the client uses a reception buffer to mitigate the bandwidth fluctuations. The reception buffer holds a few seconds of the video to provide a smooth video playout. However, small receiving buffers must be used in live streaming to minimize the latency. The size of the client buffer must not exceed a duration of two segments Δ according to [52].

6. Decoding time of the segment T_{d} also depends on the used codec and the computational power of the device

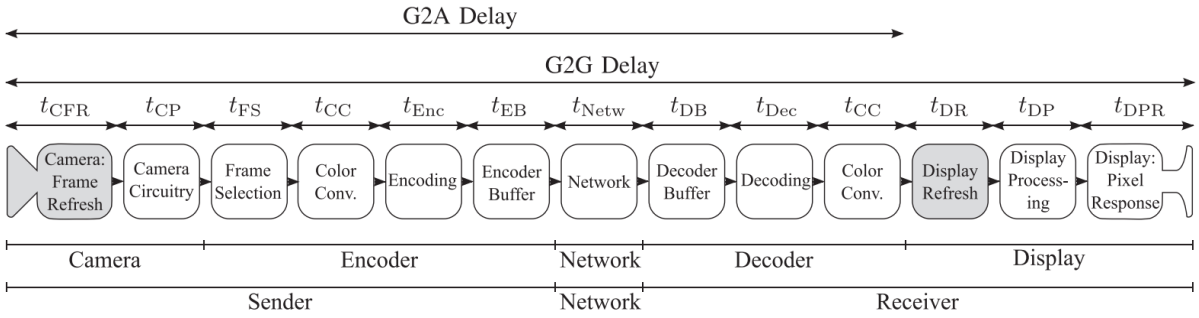

Figure 2.4: Point-to-point video transmission model proposed by Bachhuber *et al.* This image is reused from [19]

Accordingly, the total delay for HTTP streaming is

$$T_{\text{tot}} = T_{\text{acq}} + T_{\text{seg}} + T_{\text{af}} + T_{\text{ch}} + T_{\text{b}} + T_{\text{d}}, \tag{2.17}$$

in the worst case, the total end- to-end delay is

$$T_{\text{tot}} = T_{\text{acq}} + 5\Delta + T_{\text{link}} + T_{\text{d}}. \tag{2.18}$$

Given (2.18), the total latency of the DASH session can be minimized by setting the segment duration Δ to the smallest possible value, *e.g.,* 1 s.

2.2.2 Delay components of point-to-point video transmission systems

Bachhuber *et al.* [19] present a model for point-to-point video communication, as well as the main components of the end-to-end latency, including the camera, coding/decoding, network, and display delays. The authors also introduce the definitions of the Glass-to-Algorithm (G2A) and the Glass-to-Glass (G2G) delays.

The G2A delay characterizes the time difference between a visible event and the reception of the first image of this event by the image processing algorithm in the client side. Hence, the minimization of G2A is critical in delay-sensitive control applications. The Glass-to-Glass (G2G) delay is considered in applications such as live streaming where the video sequence is presented to a human observer. G2A is computed for machine vision systems that employ image processing algorithms to produce autonomous actions. If the video is then displayed to a human observer, the G2A is determined by omitting the display procedure delay from the G2G delay.

Figure 2.4 represents the video transmission model presented in [19]. The main components of the G2G latency they identified are as follows:

1. Camera Frame Refresh delay t_{CFR} : The frame is completely captured after a period t_{Cam}, in which the camera sensor is exposed to light.

2. Camera Circuitry delay t_{CP}: t_{CP} is the time necessary for the camera processing unit to read the pixel values then apply elementary processing operations, such as offset, white balance, and analog to digital conversion. As these operations are usually hardware-implemented, the resulting delay t_{CP} is typically in the order of a few milliseconds, *e.g.*, t_{CP} = $710\ \mu$s ± $62.5\ \mu$s in Guppy Pro1 cameras [19].

3. Frame Selection delay t_{FS} : The captured frames are usually all forwarded to the encoder in standard video transmission applications. Occasionally, some video frames can be skipped and forward fewer frames to reduce the average bitrate of the encoded video sequence and thus the G2G delay. Bachhuber *et al.* [19] propose a frame selection algorithm of $t_{FS} = 246\ \mu$s to $810\ \mu$s for frames at resolution 640×480 pixels.

4. Color conversion t_{CC} and Encoding time t_{Enc} : t_{Enc} delay depends on the computational power of the device, the video compression standard (*e.g.*, VVC, HEVC, AVC), and the used software (*e.g.*, HM [8] and x265 [7] are both HEVC encoders). In addition, color space conversion with a delay t_{CC} can be applied before encoding when the camera captures the frames in RGB format, and encoders compress the frames in YUV format. According to the authors, the segmentation of the compressed frame and data packetization in RTP packets takes a negligible amount of time.

5. Encoder Buffering delay t_{EB} and Decoder Buffering delay t_{DB} : Large encoder buffering delay t_{EB} and decoder buffering delay t_{DB} are observed when channel transmission rate is low compared to the video encoding bitrate. In general, the video bitrate must be quickly and accurately adapted to the channel transmission rate, and the buffers load must be kept small to minimize the buffering delays. The match between video encoding rate and the available transmission rate is ensured using bitrate adaptation algorithms. Section 2.2.3 reviews some of them.

6. Network delay t_{Netw}: t_{Netw} includes the processing, queuing, and propagation time of the packets in the wireless links and the intermediate routers of the IP network.The frame transmission time is R_n/C_n with R_n is the size of frame n (in bits) and C_n is the estimated transmission rate (in bps). Additionally, the signal propagation delay on a wireless channel and the physical links are calculated relative to the speed of light and the refractive index of the optical fiber, and the speed of electrical signals in the physical cables respectively.

7. Decoding time t_{Dec} and Color Conversion t_{CC} : Like t_{Enc} , t_{Dec} depends on the used hardware, the video compression standard, and the software decoder. The authors used the libavcodec/h.264 decoder with an average decoding delay of $t_{Dec} = 272\,\mu$s. Color space conversion can also be applied with an additional delay of t_{CC}.

8. Display Refresh t_{Dis} : t_{Dis} is the time difference between reading out of the frame data from the graphics buffer and forwarding it to the display electronics. It is a constant duration of $1/f_{Dis}$, with f_{Dis} being the rate at which the display panel is refreshed.

9. Display Processing t_{DP}: t_{DP} is the time difference between the instant when the new pixel values are sent to the display and when the display electronics are ready to change the corresponding pixel values. This processing delay varies widely for different monitors. The authors recorded a delay of 1 ms in a Samsung 2233BW monitor and 23 ms in a DELL U2412M.

10. Display Pixel Response t_{DPR} : It is the time for the pixels in LCD monitors to respond to a change in voltage. For example, the Acer XB270H monitor has a t_{DPR} response time of less than 1 ms.

2.2.3 Bitrate Adaptation schemes

The video rate adaptation algorithms available in the literature can be classified according to the input information used for their decision into bandwidth-based, buffer-based, or mixed approaches. Depending on the location the control algorithm is implemented, one may also identify server-based and

client-based approaches. Rate control algorithms adopt also different control granularity, ranging from segment-level to frame-level control. Finally, in push-based approaches, the client does not have to regularly request video segments from the server, while in pull-based methods, the client must request each video segment from the server.

Paper Reference	Method	Control at client	Control at server	Based on buffer	Based on bandwidth	Startup delay	Buffer size	Granularity of control
[11]	PANDA	X			X	NA	30 s	Segments
[10]	Festive	X			X	NA	30 s	Segments
[13]	BBA	X		X		NA	240 s	Segments
[12]	BOLA	X		X		NA	5 - 100 s	Segments
[54]	BOLA-E	X		X		3-50s	10-25s	Segments
[54]	DYNAMIC	X		X	X	3-50s	10-25s	Segments
[55]	PI controller	X		X	X	NA	30 s	Segments
[56]	MPC	X		X	X	1-10s	25 s	Segments
[57][58]	MDP	X		X	X	NA	30 s	Segments
[59]	Q-learning	X		X	X	NA	20 s	Segments
[60]	Deep Learning		X		X	NA	NA	Segments
[61]	K-Push HTTP/2 streaming	X		X		6 s	12 s	Segments
[62]	Frame Discarding	X			X	1s	NA	Segments
[63]	Deep RL		X	X	X	0.5 s	NA	Frames

Table 2.1: Comparison of some streaming methods (NA indicates absence of available information)

Bandwidth-based schemes, such as Festive [10] and PANDA [11], select, for each video segment, a video bitrate inferior or at most equivalent to the observed or inferred capacity of the network. These relatively conservative schemes may lead to a suboptimal exploitation of the available resources.

Buffer-based schemes, such as BBA [13] and BOLA [12], evaluate the target video rate as a function of the playback buffer level of the client.These schemes aim at stabilizing the level of the client buffer to ensure continuous video playback and avoid freezes when the buffer is empty. Such control may result in frequent bitrate switching which may affect the QoE. A comparative study of these different schemes has been proposed in [64] considering mobile network traces. Buffer-based schemes have been observed to outperform the other schemes in terms of adaptability. Nevertheless, they lack in stability, especially when small buffers are considered.

BOLA works best in permanent regime rather than during transients, *e.g.*, during startup when the client buffer is empty. In such situations, many low-

bitrate segments are transmitted before the client buffer reaches a sufficient level to download higher bitrate segments. In addition, since the selected coding bitrates are directly proportional to the client buffer level and the buffer size in the case of live streaming is small, the thresholds between different bitrate choices may be too close in live streaming. Thus, small variations of the size of video segments due to variable bitrate (VBR) coding of the video sequences may cause oscillation between the encoding bitrates. Spiteri *et al.* [54] improve the responsiveness of BOLA to these issues using a placeholder algorithm that changes the client buffer level using virtual video segments containing no data. The proposed algorithm inserts enough virtual segments in the client buffer to obtain the optimal bitrate that maximizes the client QoE. DYNAMIC, introduced in [54], is another improvement to BOLA that selects the target bitrate based on the available bandwidth at the startup phase or when the buffer level is low. DYNAMIC then switches to baseline BOLA when the buffer level is high enough. This approach provides better performance than regular BOLA because bandwidth-based algorithms perform better at low buffer levels.

Buffer-based and bandwidth based approaches have been combined to address their respective drawbacks, aiming at fully exploiting the available bandwidth and stabilizing the buffer level. In [56, 55], control theory is employed to design a predictive control algorithm that combines throughput and buffer occupancy information. De Cicco *et al.* [55] propose a proportional-integral (PI) controller to maximize the client QoE by selecting the optimal bitrate of the video segments given the available bandwidth and the client buffer level. Yin *et al.* [56] propose an MPC algorithm for video rate adaptation at segment level combined with HAS. The algorithm chooses the bitrate of the segments by solving a specific QoE maximization problem at each time step. Moreover, it uses N steps ahead throughput predictions, implying the need to use an efficient throughput predictor. The work does not discuss the design of effective throughput predictors and assumes that predictors are given. This MPC algorithm has a significant computational complexity, which is problematic for real-time video transmission. Thus, the authors simplify the algorithm to a lookup table, built offline and indexed by the client buffer and channel states. The output of the lookup table is the optimal bitrate of

the video segments. Yet, this MPC algorithm is not suitable for low-latency, as it operates at the segment level, assuming that the rate of each segment can be chosen independently of the rate of previous segments.

Rate adaptation for streaming over mobile networks is cast in the framework of Markov Decision Processes (MDP) by Bao and Valentin [57]. Buffer and channel states are employed and the use of a channel state predictor is shown to further improve performance. In the same spirit, Zhou *et al*. [58] propose an MDP-based adaptation scheme for Dynamic Adaptive Streaming over HTTP (DASH) aiming at maximizing the QoE. Their method takes into account additional key factors that impact directly the client QoE, including rate switching frequency and amplitude, buffer overflow/underflow, and the number of stalls. A Q-learning-based adaptation scheme is proposed in [59]. This scheme dynamically learns the optimal policy corresponding to the channel and client buffer states. In addition, the reward function used in the learning process can be tuned to emphasize different aspects of the QoE of the client.

The previous approaches are pull-based schemes: video rate adaptation is performed at the client. This type of approaches is well suited to streaming of video contents to mobile devices through a wireless access network. The available transmission rate mainly depends on the position of the client and of the number of clients sharing the same wireless resource. Moreover, the client has a delay-free access to the level of its reception buffer, and is able to estimate the capacity of its wireless channel, which facilitates control. Conversely, in live transmission of sports events such as car races or sailing contests with on-board video cameras, or in remote car driving or drone operation, the acquisition device (camera and encoder) becomes the server delivering content to remote clients. The server is moving and this motion may lead to fast and significant fluctuations of the wireless channel characteristics between the acquisition device and the eNodeB (4G) or gNodeB (5G) [65] to which it is connected. Figure 2.5 shows the difference between the pull-based and the push-based streaming method.

In addition, the segmentation of the video content in HAS streaming introduces at least one segment duration into the total latency of the streaming session. In live streaming, the video bitrate must be adapted to the available instantaneous transmission rate to avoid accumulation of encoded video frames

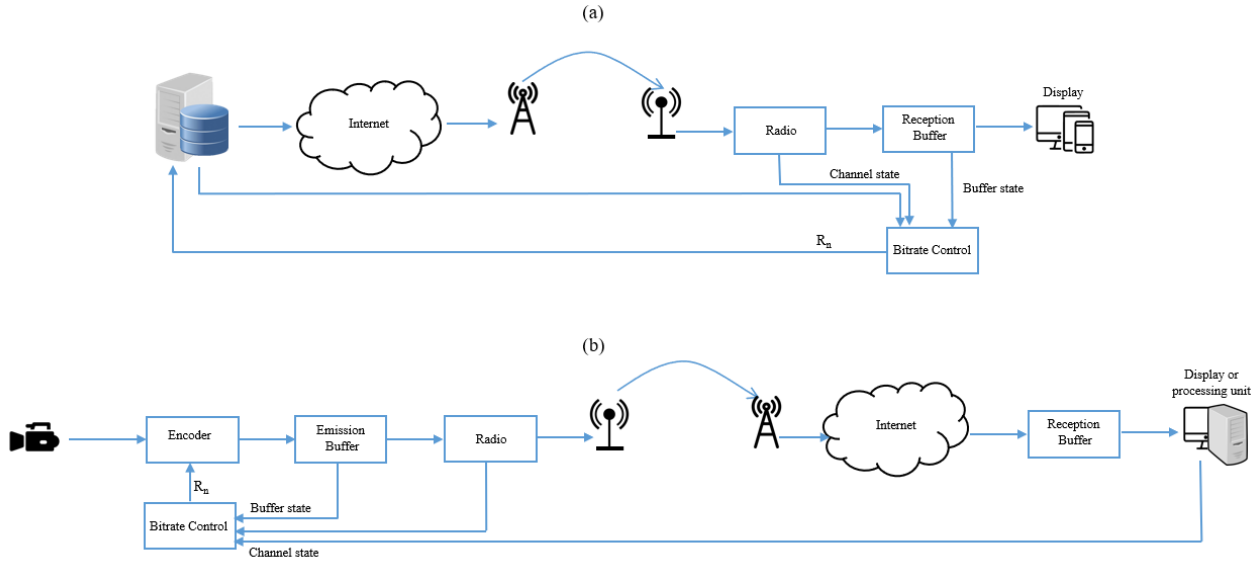

Figure 2.5: (a) Pull-based streaming architecture, (b) push-based streaming architecture.

at the server. Accordingly, the period at which the rate control has to be performed has to be significantly smaller than what is considered in classical segment-based streaming applications, where segments may last from 2 to 10 seconds.

Ben Yahia *et al.* [62], propose an approach to deal with network variations occurring at timescales smaller then the video segment duration. They propose a bitrate adaptation scheme in which the client can discard a set of video frames from the downloaded segment when the available bandwidth is not sufficient to transmit the whole video segment. The proposed scheme is client-based and uses both HTTP/2 and DASH standards, which makes it compliant with the general architecture of HAS.

Feng *et al.* [63] present Vabis, a server-side bitrate adaptation for low-latency applications based on Reinforcement Learning (RL). Vabis adjusts the video bitrate and minimizes the latency between a mobile client player and the HTTP media server located in the core network. It selects the encoding bitrate of a small piece of the video segment. Vabis is based on the HTTP adaptive streaming framework and MPEG Common Media Application Format (CMAF) [66] standard for video delivery. Contrary to HAS, in which the video segment can not be transmitted before it is encoded entirely. CMAF

allows the video encoder to gradually output small pieces of the video segment chunk for delivery immediately after encoding it. This allows to achieve small control granularity and minimize the latency between the client player and the streaming media server. The server receives the measurement of the channel and the client buffer state every 0.5s and selects the optimal encoding bitrate accordingly.

In [61], a push-based method using the HTTP/2 protocol has been proposed. The client sends one request to the server, to push K segments of bitrate V. These K segments are sent in a batch. A probabilistic buffer model is used to optimize the two parameters K and V at the client side. The HTTP/2 based approach significantly reduces the latency of segment delivery in high-RTT networks, and it decreases significantly the startup delay because of using shorter segments.

Du *et al*. [60] propose a Deep Neural Network-Driven Streaming method. The encoder sends the video segment encoded in low quality to the server. The server runs a Deep Neural Network (DNN) to determine the regions of the frames that need to be re-encoded at higher bitrates, and request it back. To determine the bitrates of the low- and high-bitrate regions, a feedback control system is implemented at the server side. The control system uses the total bandwidth of the previous encoded segment, in addition to the currently estimated bandwidth, to tune the resolution and quantization parameters of both the low- and high-bitrate regions.

Table 2.1 compares the above streaming methods. This thesis addresses the bitrate adaptation problem in low latency video streaming and presents a novel method for adjusting the video bitrate at the frame level. The proposed algorithm is based on Model Predictive Control and, unlike most state-of-the-art techniques, operates at the server-side (transmitter). These properties make it suitable for low latency streaming from mobile terminals and over variable channel characteristics.

Chapter 3

Rate-QP-Distortion model for video streaming and compression

3.1 Introduction

Rate control plays a vital role in ultra-low latency streaming. The bitstream size resulting from the encoding process must not violate the constraints imposed by the channel transmission rate, encoder/decoder buffer sizes, and the constant glass-to-glass delay.

In applications such as remote driving [67] or remote surgery [68], only small buffers are allowed at the transmitter and the receiver to limit the latency to a minimum. Accordingly, any mismatch between the encoding rate and available transmission rate may lead to an unacceptable delay increase. The small buffers at the transmitter and the receiver cannot mitigate transmission jitter.

Video encoding rate control can be performed at the macroblock level, the frame level, or on a group of pictures (GOP). It relies on a model of the relation between the size of the bitstream resulting from the encoding process and the video encoding parameters. The rate model is used to determine the optimal encoding parameter for having a bitstream of size at most equal to the allocated bits budget in the constraints mentioned above. The quantization parameter QP is usually considered used in rate control as it directly impacts the size of the resulting bitstream.

Our goal is to adjust the video bitrate at the frame level for low latency streaming. We must use a rate model to determine the optimal QP to encode

the frames in a given bit budget. The bit budget is determined by the bitrate adaptation algorithm that will be introduced in Chapter 4.

Parametric models between the rate of the frame and its QP have been proposed in the literature. However, the precision of these models cannot be reliable for transmission in band-limited channels as they do not consider the temporal dependency between the frames.

We propose a new model of the relation between the encoding bitrate R_n of the frame n and its quantization parameter QP_n depending on the Mean Square Error (MSE) distortion D_{n-1} for the reference frame $n-1$. Our proposed model, denoted as R-(QP, D), is part of our low-latency streaming scheme proposed in chapter 2. We use this model to determine the QP for encoding the frame and having a bitstream size at most equal to its allocated bit budget.

In this chapter, we compare the performance of the proposed approach with state-of-the-art models in terms of accuracy of the encoding rate prediction, especially when the encoding parameters change significantly in time, as required by ultra low-latency streaming applications.

The contributions of this chapter are the following :

- In Section 3.2, we propose a novel Inter-dependent R-(QP, D) model. We give insights into how the proposed model structure has been obtained, and we give detailed instructions for estimating its parameters.

- In Section 3.4, we evaluate the performance of our proposed model in two coding scenarios: encoding at constant QP and encoding with time-variant QP. The performance of our model is compared to those of models in Eqs. (2.6) [4], (2.9) [5] and (2.12) [6].

- In Section 3.3, we present a method that allows finding the model parameters iteratively using the parameters of the first frame and a small number of encoding trails.

The results show that our proposed model outperforms the models in the literature, especially in a low-bitrates encoding which demonstrates the benefits of accounting for the distortion of the reference frame.

3.2 Inter-dependent R-(QP, D) model

This section introduces the proposed R-(QP, D) model for rate control at the frame level. The proposed model has been derived based on experiments with H.265/HEVC [69] (using the x265 encoder [7]). Nevertheless, the methodology is generic and can be extended to VVC [70] or AV1 [71].

In order to motivate our model, we show experiments considering two typical frames, with indexes $n = 79$ and $n = 131$ of the video sequence *ParkScene* encoded with *x265* encoder. Figure 3.1 shows the rate R_n of frame n as a function of the distortion of the reference frame D_{n-1}, for different values of QP_n. These results have been obtained by encoding frame n with six different $QP_n \in \{QP^{(1)} = 20, QP^{(2)} = 24, \ldots, QP^{(6)} = 40\}$.

All previous frames, including frame $n-1$ have been encoded with seven $QP_{n-1} = QP_n + \Delta QP$, where $\Delta QP \in \{\Delta QP^{(1)} = -7, \ldots, \Delta QP^{(7)} = 5\}$. The resulting rates are denoted as $R_{n,i,j}$, where $QP_n = QP^{(i)}, i = 1, \ldots, 6$ and $\Delta QP = \Delta QP^{(j)}, j = 1, \ldots, 7$.

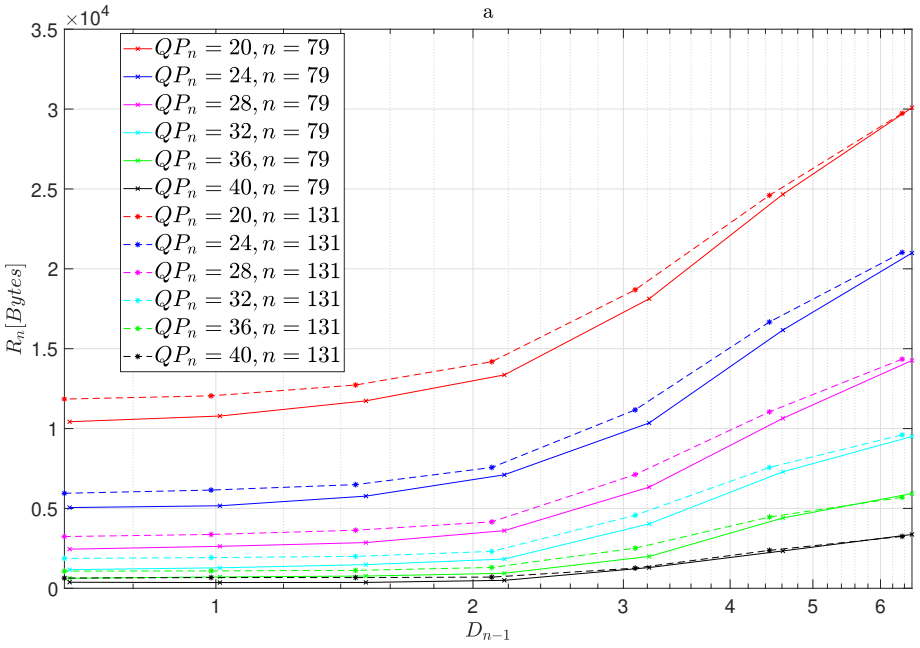

Figure 3.1: The bitrate R_n for the frames $n = 79$ and $n = 131$ of *ParkScene* as a function of the distortion D_{n-1} of the reference frame for different values of QP_n.

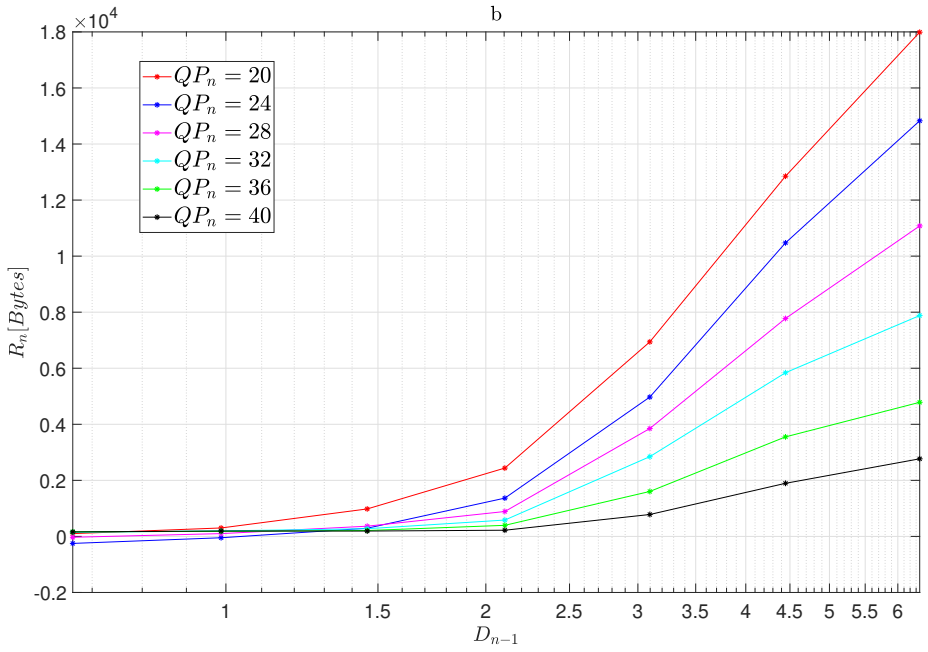

Figure 3.2: R_n^0 for frame $n = 79$ of *ParkScene* as a function of D_{n-1} for different values of QP_n.

Each curve for a given value of QP_n consists of two almost linear parts as a function of $\log(D_{n-1})$. Conversely, for small values of D_{n-1}, R_n increases slowly with D_{n-1}. For larger values of D_{n-1}, the increase is steeper. We observe that the dependance of $R_{n,i,j}$ can be described by a family of sigmoids depending on $\log(D_{n-1})$ and QP_n.

We propose the following R-(QP, D) model

$$\begin{aligned} R_n(QP_n, D_{n-1}) &= g_1(QP_n) \\ &+ g_2(QP_n)\left(\tanh\left(g_3(QP_n)\log(D_{n-1}) - g_4(QP_n)\right) + 1\right), \end{aligned} \tag{3.1}$$

where $g_1(QP_n)$ describes the bitrate R_n for small values of D_{n-1}, and

$$\begin{aligned} R_n^0(QP_n, D_{n-1}) &= R_n(QP_n, D_{n-1}) - g_1(QP_n) \\ &= g_2(QP_n)\left(\tanh\left(g_3(QP_n)\log(D_{n-1}) - g_4(QP_n)\right) + 1\right), \end{aligned} \tag{3.2}$$

describes the bitrate R_n for large values of D_{n-1}.

Figure 3.3-a represents R_n as function of QP_n for the smallest values of D_{n-1} obtained for frame $n = 79$. In this regime, R_n decreases exponentially

with QP_n according to

$$g_1\left(QP_n\right) = p_1 \exp\left(-p_2 QP_n\right). \tag{3.3}$$

Figure 3.2 represents $R^0_{n,i,j} = R_{n,i,j} - g_1(QP^{(i)})$ as a function of D_{n-1} for different values of $QP_n = QP^{(i)}$. For each value of $QP_n = QP^{(i)}, i = 1, \ldots, 6$, a least-squares estimation of g_2, g_3, and g_4 is performed using $R^0_{n,i,j}$, $j = 1, \ldots, 7$ to get $\widehat{g}_2(QP^{(i)})$, $\widehat{g}_3(QP^{(i)})$, and $\widehat{g}_4(QP^{(i)})$. Figure 3.3-b shows that $\widehat{g}_2$ as a function of $\log(QP^{(i)})$ is adequately described by an affine model with two parameters p_3 and p_4

$$g_2\left(QP_n\right) = p_3\left(-p_4 \log\left(QP_n\right) + 1\right). \tag{3.4}$$

Figure 3.3-c illustrates $\widehat{g}_3$ as function of $QP^{(i)}$, which is adequately represented by the linear model depending on p_5

$$g_3\left(QP_n\right) = p_5 QP_n. \tag{3.5}$$

Finally, Figure 3.3-d illustrates the relation between the square root of $\widehat{g}_4$ and $QP^{(i)}$, which justifies the following quadratic model for the dependency of g_4 in QP_n, depending on the parameters p_6 and p_7

$$g_4\left(QP_n\right) = \left(p_6 QP_n - p_7\right)^2. \tag{3.6}$$

Consolidating the previous results, the proposed model (3.1) involves a vector of 7 parameters $\mathbf{p} = (p_1, \ldots, p_7)$, which value is frame-dependent and needs to be determined to accurately predict R_n as a function of QP_n and D_{n-1}.

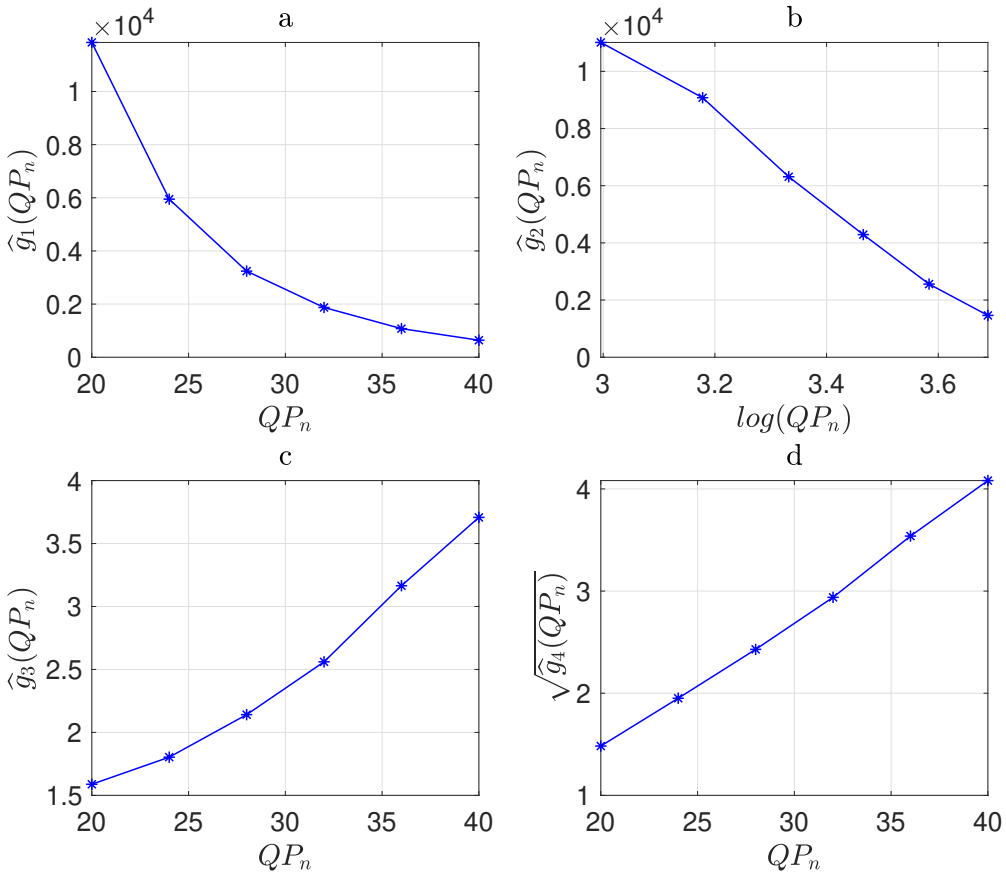

Figure 3.3: $\widehat{g}_1$, $\widehat{g}_2$, $\widehat{g}_3$, and $\widehat{g}_4$ as a function of $QP^{(i)}$ or $\log\left(QP^{(i)}\right)$ for frame 79 of *ParkScene*

3.3 Recursive estimation of the R-(QP, D) model parameters

In the considered low-latency streaming scheme, the control is performed at the frame level and not at the GOP or chunk level, as in most of the streaming solutions [2]. Consequently, the value of $\mathbf{p}_n$ has to be estimated online and for each encoded frame to accurately predict R_n of frame n as a function of QP_n and D_{n-1}. Coding the frame with different QP_n and QP_{n-1} to build the model is not possible in this case because a large number of coding trails generates significant delay.

Our aim in what follows is to evaluate $\mathbf{p}_n$ from $\mathbf{p}_{n-1}$ in an iterative way. The

vector $\mathbf{p}_n$ can be expressed as

$$\mathbf{p}_n = \mathbf{p}_{n-1} + \boldsymbol{\delta}_n, \tag{3.7}$$

where $\boldsymbol{\delta}_n$ represents the difference between $\mathbf{p}_n$ and $\mathbf{p}_{n-1}$. Assuming that the rate-distortion characteristics of consecutive video frames of the same type change smoothly when there is no scene change, $\boldsymbol{\delta}_n$ is usually small.

To estimate $\mathbf{p}_n$ from $\mathbf{p}_{n-1}$, consider M video encoders running in parallel. Each of the $M-1$ first encoders processes frame n with a different $QP_{n,m}$, $m = 1, \ldots, M-1$. The M-th encoder uses the value $\widehat{QP}_n$ for frame n provided by the encoding rate controller. Let $D_{n-1,m}$, $m = 1, \ldots, M-1$, be the distortion obtained for frame $n-1$ when encoded by the m-th encoder and $R_{n,m}$ be the rate of frame n at the output of the m-th encoder.

Assume that an estimate $\widehat{\mathbf{p}}_{n-1}$ of $\mathbf{p}_{n-1}$ is available. Using $D_{n-1,m}$, $QP_{n,m}$, and $R_{n,m}$, one considers the estimate $\widehat{\boldsymbol{\delta}}_n$ of $\boldsymbol{\delta}_n$ that minimize the following regularized weighted least-squares cost function

$$\widehat{\boldsymbol{\delta}}_n = \arg\min_{\boldsymbol{\delta}} \sum_{m=1}^{M} w_{n,m} \left(R_{n,m} - R\left(QP_{n,m}, D_{n-1,m}, \widehat{\mathbf{p}}_{n-1} + \boldsymbol{\delta}\right)\right)^2 + \alpha \boldsymbol{\delta}^T \boldsymbol{\delta}, \tag{3.8}$$

where $w_{n,m} \geqslant 0$, $m = 1, \ldots, M$ are some weights and $\alpha \geqslant 0$ is a regularizing coefficient, which aim is to favor small values of $\boldsymbol{\delta}$.

Considering the first-order Taylor expansion of $R_n\left(QP_{n,m}, D_{n-1,m}, \widehat{\mathbf{p}}_{n-1} + \boldsymbol{\delta}\right)$, one gets

$$\widetilde{R}\left(QP_{n,m}, D_{n-1,m}, \widehat{\mathbf{p}}_{n-1} + \boldsymbol{\delta}\right) = R\left(QP_{n,m}, D_{n-1,m}, \widehat{\mathbf{p}}_{n-1}\right) + \frac{\partial R\left(QP_{n,m}, D_{n-1,m}, \widehat{\mathbf{p}}_{n-1}\right)}{\partial \widehat{\mathbf{p}}_{n-1}^T} \boldsymbol{\delta}. \tag{3.9}$$

Introducing

$$\mathbf{y}_n = \left(R_{n,1} - R\left(QP_{n,1}, D_{n-1,1}, \widehat{\boldsymbol{p}}_{n-1}\right), \ldots, R_{n,M} - R\left(QP_{n,M}, D_{n-1,M}, \widehat{\boldsymbol{p}}_{n-1}\right)\right)^T,$$

$$\mathbf{W}_n = \mathsf{diag}\left(w_{n,1}, \ldots, w_{n,M}\right),$$

$$\mathbf{X}_n = \begin{pmatrix} \frac{\partial R(Q_{n,1}, D_{n-1,1}, \widehat{\mathbf{p}}_{n-1})}{\partial \widehat{\mathbf{p}}_{n-1}^T} \\ \vdots \\ \frac{\partial R(Q_{n,M}, D_{n-1,M}, \widehat{\mathbf{p}}_{n-1})}{\partial \widehat{\mathbf{p}}_{n-1}^T} \end{pmatrix},$$

and using (3.9), one may rewrite (3.8) as

$$\widehat{\boldsymbol{\delta}}_n = \arg\min_{\boldsymbol{\delta}} \mathbf{W}_n \left(\mathbf{y}_n - \mathbf{X}_n\boldsymbol{\delta}\right)^T \left(\mathbf{y}_n - \mathbf{X}_n\boldsymbol{\delta}\right) + \alpha\boldsymbol{\delta}^T\boldsymbol{\delta}.$$

Consequently,

$$\widehat{\boldsymbol{\delta}}_n = \left(\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n + \alpha\mathbf{I}\right)^{-1}\mathbf{W}_n\mathbf{X}_n^T\mathbf{y}_n. \tag{3.10}$$

Then, $\widehat{\mathbf{p}}_n$ is determined from $\widehat{\mathbf{p}}_{n-1}$ and $\widehat{\boldsymbol{\delta}}_n$ as

$$\widehat{\mathbf{p}}_n = \widehat{\mathbf{p}}_{n-1} + \widehat{\boldsymbol{\delta}}_n.$$

The choice of $\mathbf{W}_n$ and of the different $Q_{n,m}, m = 1, \ldots, M-1$ is discussed in Section 3.4.1.

3.4 Evaluation of the proposed model

The performance of the proposed model to predict R_n as a function of QP_n is compared to the reference models in Eqs. (2.6) [4], (2.9) [5], and (2.12) [6], used at a frame level.

3.4.1 Experimental setup

Three *JCT-VC* test sequences, namely *Tango*, *Racehorses*, and *ParkScene* [72], are selected for the experiments, as well as a recording from the inside of a racing car (*Magnycours* [73]). The encoding is performed with the *x265* software [7], configured in low delay mode and with an intra-refresh cycle of one second.

The parameters for the three reference models and the proposed models are only estimated for every four frames of the video sequences. The parameters are then assumed to remain constant for the successive three frames. For that purpose, 42 coding trials are done with $QP_n \in \{QP^{(1)}, \ldots, QP^{(6)}\}$ and $QP_{n-1} = QP_n + \Delta QP$ with $\Delta QP \in \{\Delta QP^{(1)}, \ldots, \Delta QP^{(7)}\}$ to get $R_{n,i,j}$, $i = 1, \ldots, 6$, and $j = 1, \ldots, 7$. A weighted least-square estimation of the model parameters is then performed considering the following cost function

$$C_n\left(\mathbf{p}\right) = \sum_{i=1}^{6}\sum_{j=1}^{7} \frac{1}{R_{n,i,j}} \left(R_{n,i,j} - R_n(QP^{(i)}, D_{n-1}^{(j)})\right)^2, \tag{3.11}$$

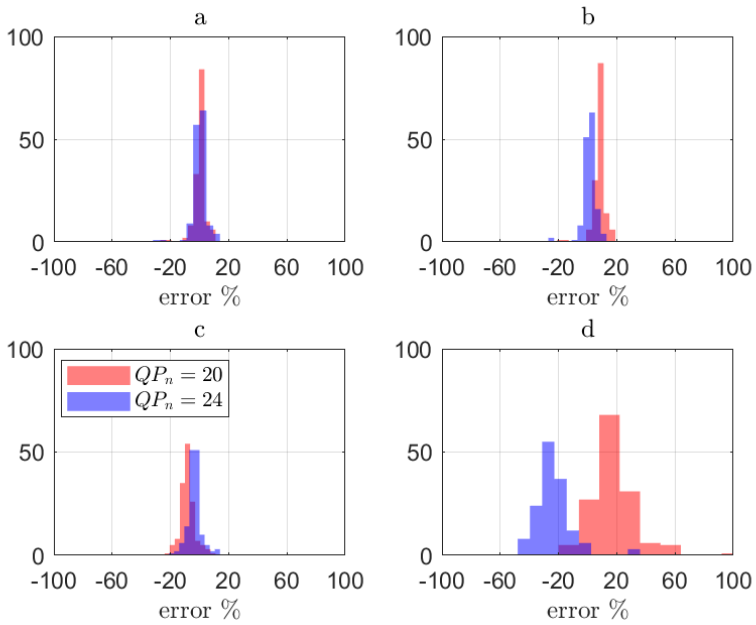

Figure 3.4: Histogram of prediction errors for *Tango* at high bitrates, (a) proposed model (3.1), (b) (2.6) from [4], (c) (2.9) from [5] and (d) (2.12) from [6].

where $R_n(QP^{(i)}, D_{n-1}^{(j)})$ is given by the proposed model in (3.1), or by the models (2.6), (2.9) or (2.12), and $D_{n-1}^{(j)}$ is the distortion for frame $n-1$.

To compare the performance of the four models, in a first set of experiments all frames of the video sequences are encoded at constant $QP \in \{QP^{(1)}, \ldots, QP^{(6)}\}$. In a second set of experiments, QP may vary from frame to frame as the realization of a first-order Markov process such that with a probability $P = 0.6$, $QP_n = QP_{n-1}$, and with a probability $1 - P$, QP_n is uniformly distributed in the set $\mathcal{Q}_n = \{QP_{n-1} - 5, \ldots, QP_{n-1} + 5\} \cap \{20, \ldots, 40\}$.

The ability to predict the actual encoding rate is evaluated using the relative rate error

$$E_n = 100 \left(R_{\text{pred}} - R_{\text{actual}}\right) / R_{\text{actual}}, \tag{3.12}$$

where R_{actual} is the actual size of the encoded frame n and R_{pred} is the predicted one obtained from the rate model.

3.4.2 Results when coding with constant QP

We first compare the performance of the four models in a set of experiments when encoding with constant QP.

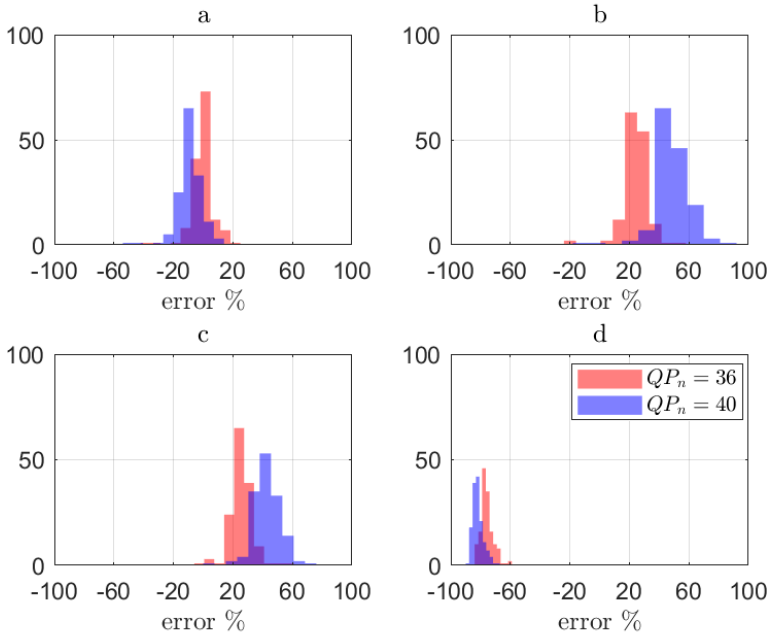

Figure 3.5: Histogram of prediction errors for *Tango* at low bitrates, (a) proposed model (3.1), (b) (2.6) from [4], (c) (2.9) from [5] and (d) (2.12) from [6].

Figure 3.4 shows the histogram of the prediction errors obtained with the proposed model in Eq. (3.1), and by the models in Eqs. (2.6), (2.9) and (2.12), used on the *Tango* sequence coded at high bitrates, *i.e.*, $QP_n = 20, 24$. We notice that our proposed model provides the best performance in this case. The errors of model in Eq. (3.1) are mainly between -13.6% to 14%, with a peak at 3.4%. The prediction errors with the models in Eqs. (2.6) and (2.9) are between -11% and 19%, with a peak near 11% and -6.5% respectively. Model in Eq. (2.12) has the worst performance, with prediction errors spreading between -50% and 64%.

Figure 3.5 shows the error histograms of the four models on the *Tango* sequence coded at low bitrates, *i.e.*, $QP_n = 36, 40$. The performance of our proposed model slightly decreases but significantly outperforms the three other ones. Prediction errors are between -28.6% and 18.3%, with a peak around 4.9%. Both models in Eq. (2.6) and Eq. (2.9) lead to errors between 0% and 81% with a peak near 48% and 27.5% respectively. The model in Eq. (2.12) largely underestimates the rate, with prediction errors distributed between -90% and -58%.

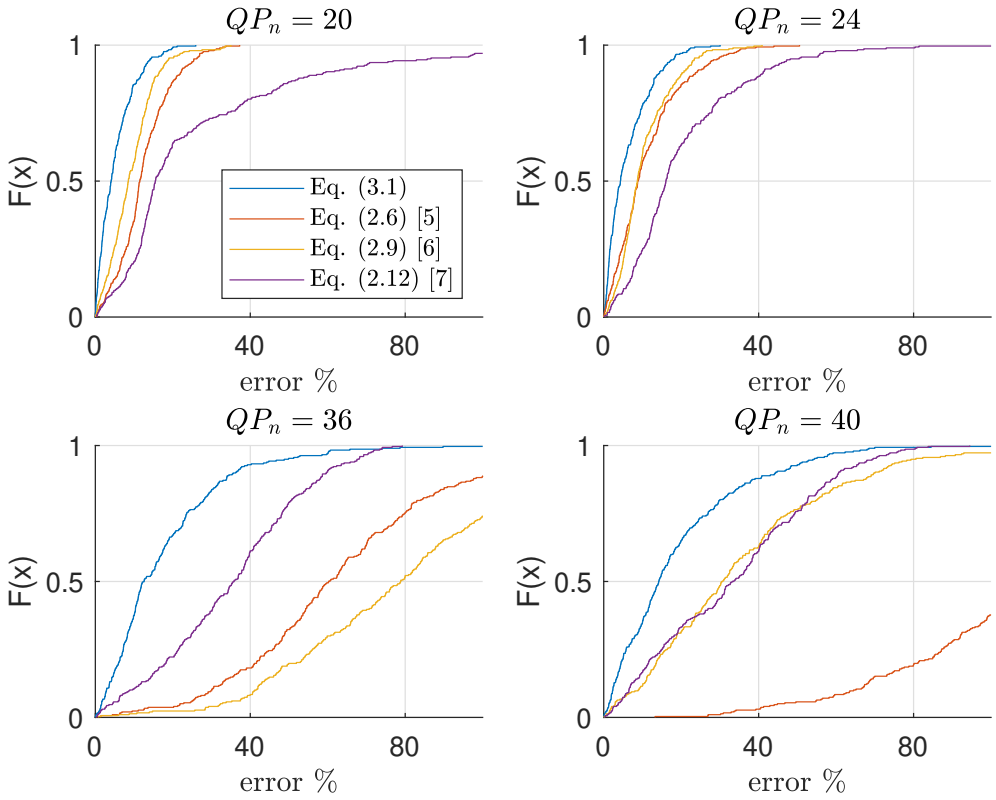

Figure 3.6: CDF of prediction errors for *Magnycours* sequence.

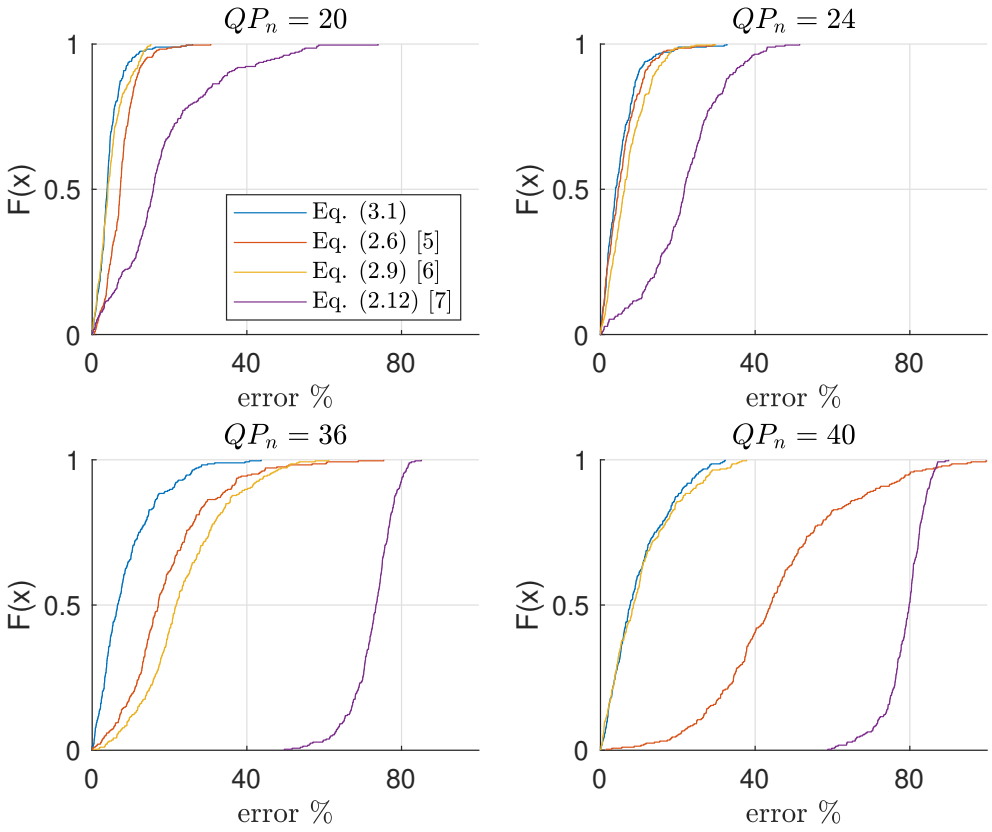

Figure 3.7: CDF of prediction errors for *RaceHorses* sequence.

Figure 3.6 shows the error cumulative distribution functions (*CDF*) when using the four $QP_n = 20, 24, 36$ and 40 for the *Magnycours* sequence. The proposed model achieves the lowest prediction error compared to the other models. For instance, with $QP_n = 20$, 90% of the prediction errors are less than 12.2% with the proposed model, compared to 16.7%, 22.6%, and 51.8% for the models in Eqs. (2.6), (2.9), and (2.12), respectively.

Figure 3.7 shows the error CDF for *RaceHorses* sequence. Here, we see close performance of models in Eqs. (3.1), (2.6), and (2.9) with $QP_n = 20$ and 24, and close performance between model in Eq. (3.1) and model in Eq. (2.6) with $QP_n = 40$. The proposed model shows a significant advantage in the other cases.

Figures 3.6 and 3.7 both show that the gains with our model tend to be more significant at low bitrates (*i.e.*, high QPs). This can be explained by the fact that we incorporate the distortion of the reference frame in our model,

which is especially important at low bitrates.

Figure 3.8 illustrates the average error CDF when coding the four sequences with a constant *QP*. Our proposed model achieves the best performance for all test sequences.

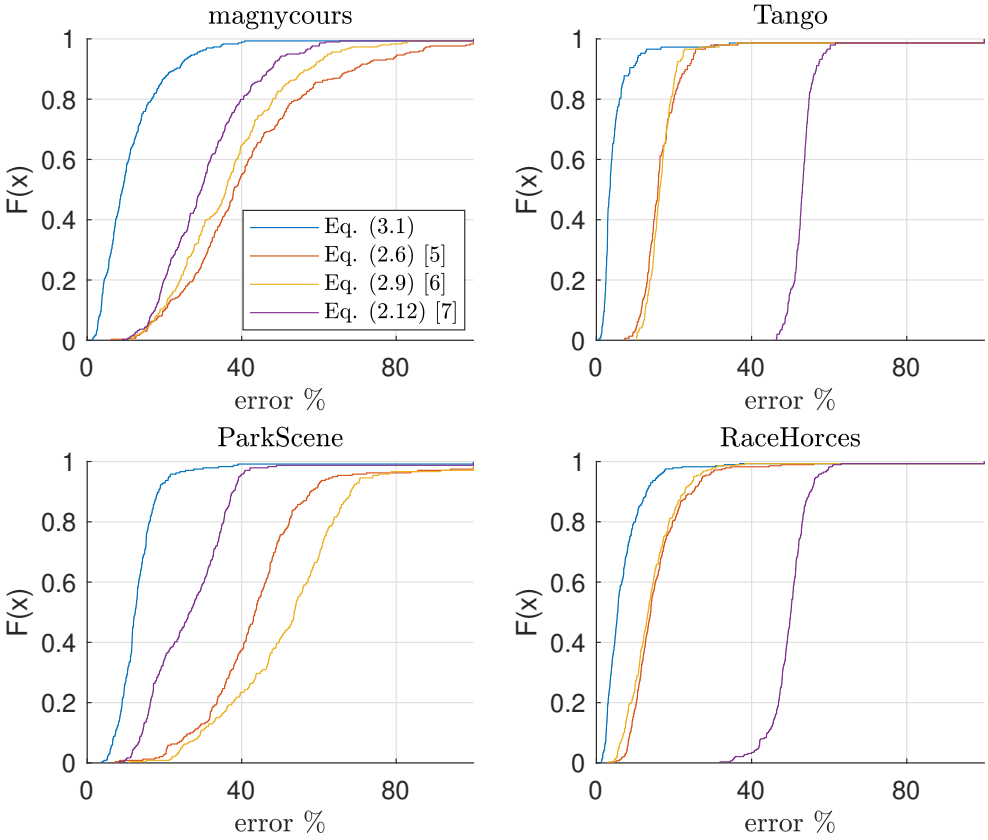

Figure 3.8: Average error CDF with constant *QP* for each sequence.

3.4.3 Results when coding with time-varying QP

We present results of the second set of experiments using time-varying QPs based on a first-order Markov process.

Figures 3.9 shows the histogram of the prediction errors obtained with the proposed model in Eq. (3.1), and by the models in Eqs. (2.6), (2.9) and (2.12), used on the *Tango* sequence coded with QP that variate as a first-order Markov process realization with a probability $P = 0.6$. Our proposed model provides the best performances with prediction error between -32% and 19.1%.

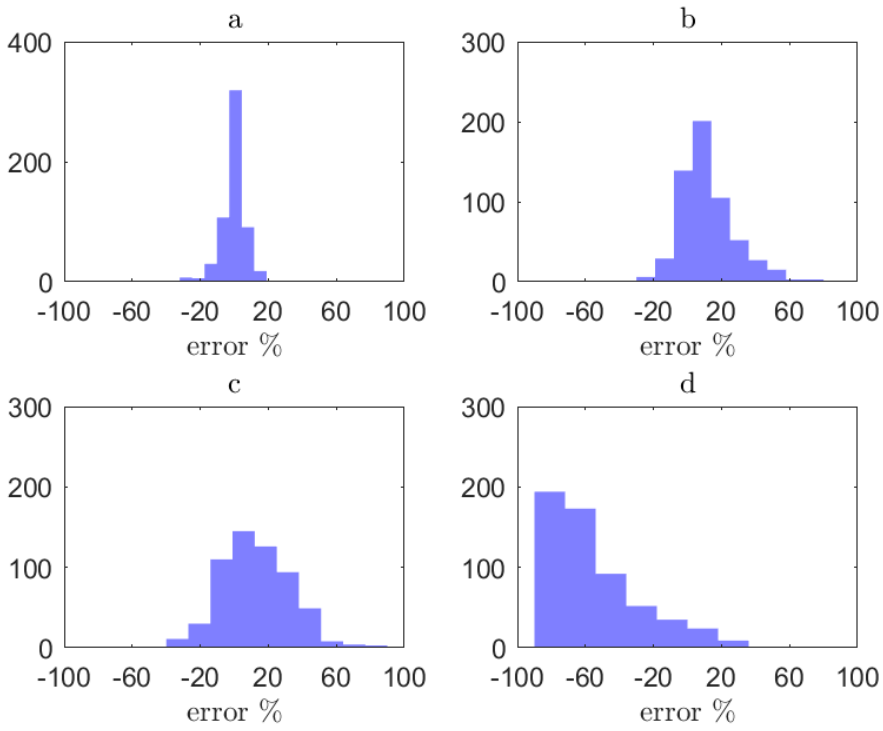

Figure 3.9: Histogram of errors for *Tango* sequences with first-order Markov process variations of QP, (a) proposed model (3.1), (b) (2.6) from [4], (c) (2.9) from [5], and (d) (2.12) from [6]

The prediction errors with the model in Eq. (2.6) leads to prediction errors between -30% and 69%, and the errors with Eq. (2.9) are between -40% and 90%. Model in Eq. (2.12) underestimate the frame sizes. Its prediction errors are between -90% and 36%.

Figure 3.10 shows the average error CDF with time-varying QPs for the four test sequences. The proposed model outperforms the other ones for all sequences. For *Magnycours*, *ParkScene*, and *Tango*, the improvements are significant, whereas for *RaceHorses*, the models in Eqs. (3.1), (2.6), and (2.9) reach close performances. This difference in performance is due to the spatial and temporal properties of the video sequence that make it difficult to determine the model parameters. Thus, the prediction error increases in some frames.

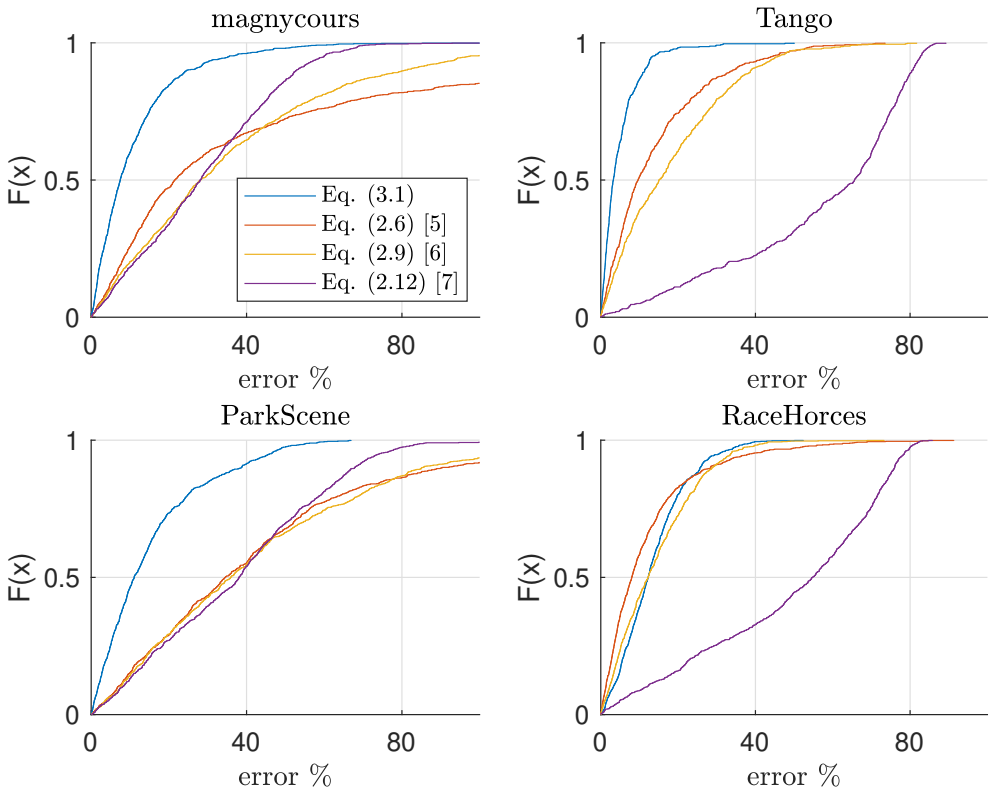

Figure 3.10: Error CDF with first-order Markov process variations of *QP* for each sequences, (a) proposed model (3.1), (b) (2.6) from [4], (c) (2.9) from [5], and (d) (2.12) from [6]

Figures 3.11-(a) illustrate the temporal variations of the frames bitstream size in *Magnycours* video sequence, as well as the predictions obtained by our proposed model in Eq. (3.1) and model in Eq. (2.9) [5]. We note that our proposed model manages to predict the size of the images after the encoding process with relatively low errors. In contrast, the model in Eq. (2.9) overestimates the size of the resulting bitstream.

Figures 3.11-(b) shows the difference between the perdition of the two models, in Eq. (3.1) and Eq. (2.9) [5], and the sizes of the frames of Magnycours video sequence in bytes. We see that our proposed model has an error of 21 bytes on average, while the model in Eq. (2.9) has an error of average of 447 bytes.

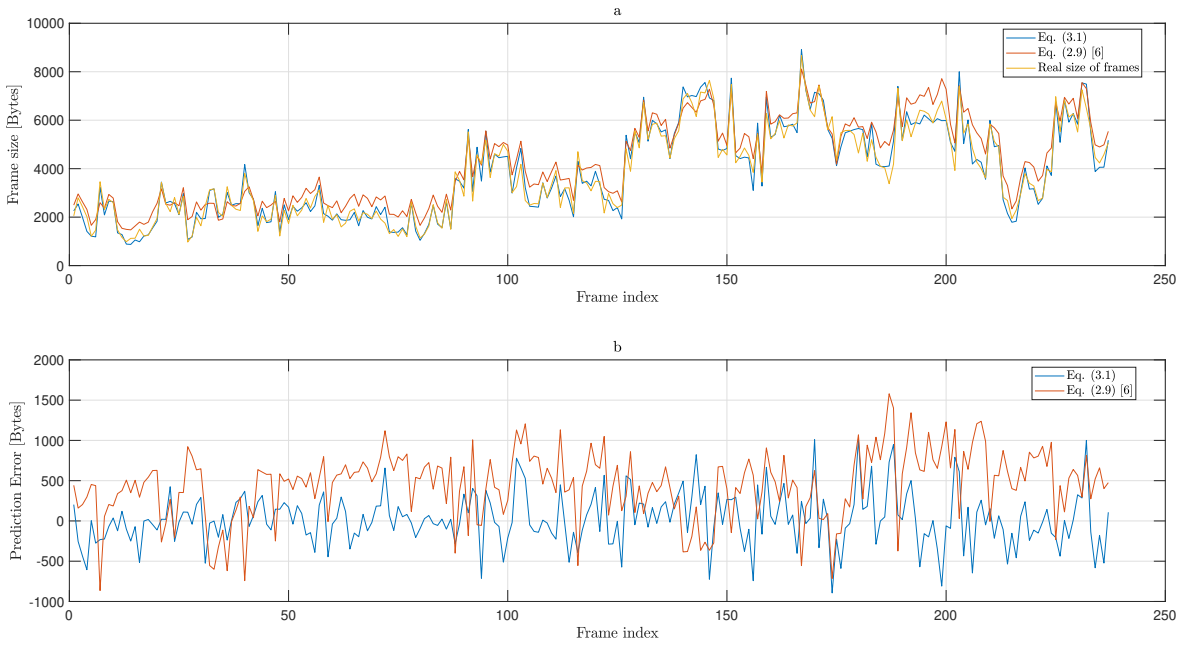

Figure 3.11: (a) The temporal variation of the sizes of the frames, predictions of the model in Eq. (3.1) and predictions of the model in Eq. (2.9) [5] with *Magnycours* sequence, (b) The temporal variation of the error between predictions of the model in Eqs. (3.1) and .(2.9) [5], and the sizes of the frames in bytes, with *Magnycours* sequence.

3.5 Evaluation of the proposed model when its parameters are iteratively estimated

In this section, we evaluate the performance of the proposed model to predict R_n when its parameters are estimated iteratively using the method presented in section 3.3.

3.5.1 Experimental setup

We consider ten transmission episodes of the video sequence *Magnycours* at resolutions 640×360 and frame rate 25 fps, in a simple network consisting of a server and a client. The server encodes the video frames using the x265 encoder [7] configured in low delay and transmits the encoded packets to the client that contains a video player. The server manages the rate control algorithm BBA [13], which selects the target encoding bitrate of the frames and

the R-(QP, D) model to determine the frames QP based on its target encoding bitrate.

The access and core networks are simulated using 4G bandwidth traces taken from [9]. More details about simulation setup and the rate control algorithm can be found in chapter 4.

The value of vector of parameters $\mathbf{p}_n$ for frame n of the R-(QP, D) model is estimated iteratively as described in Section 3.3. Apart from the encoder generating the transmitted packets, three additional encoders are used to provide data to the estimator. Accurate estimates require these encoders operate with time-varying QPs. The choice of QP for frame n and for encoder $i = 1, \ldots, 3$ is performed as follows

$$QP_{n,i} = \begin{cases} QP_{0,i} & \text{if } n = 0 \\ QP_{n-1,i} + \Delta QP_i & \text{if } n\%4 = 1, 2 \\ QP_{n-1,i} - \Delta QP_i & \text{if } n\%4 = 3, 0, \end{cases} \tag{3.13}$$

where $n\%4$ is the remainder of the division of n by 4. The vector $QP_0 = (24, 36, 40)$ contains the QPs of the first frame, and $\Delta QP = (4, 4, -4)$ is the variation of QP. This choice of QP_0 provides a better model accuracy at low rate (large values of QP). An optimization of the values of $QP_{n,i}$ for $i = 1, \ldots, 3$ to get the best parameter estimate is an important direction for future research. In (3.8), we have chosen $w_{n,m} = 1/R_{n,m}$ to better balance the importance of measurements obtained at low and high rates. This provides a fit where the relative rate error is approximately constant, whatever the encoding rate. Moreover, we set $\alpha = \lambda_1(\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n)/100$, where $\lambda_1(\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n)$ is the largest eigenvalue of $\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n$. This ensures that $\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n + \alpha\mathbf{I}$ is invertible and is sufficient to smooth out the variations of $\widehat{\mathbf{p}}_n$ with n. The ability to predict the actual encoding rate is evaluated using the same metric in Eq.3.12.

3.5.2 Results of the proposed model built with the recursive estimation

Figure 3.12 shows the variations of the QP in three out of ten transmission episodes conducted in this experiment. The QP variations here are more stable than the variations used in the second experiment of section 3.4.1.

The QPs of the frames are determined using the R-(QP, D) model. Each QP generates a bitrate R_n close to the target bitrate R_n^* selected by the algo-

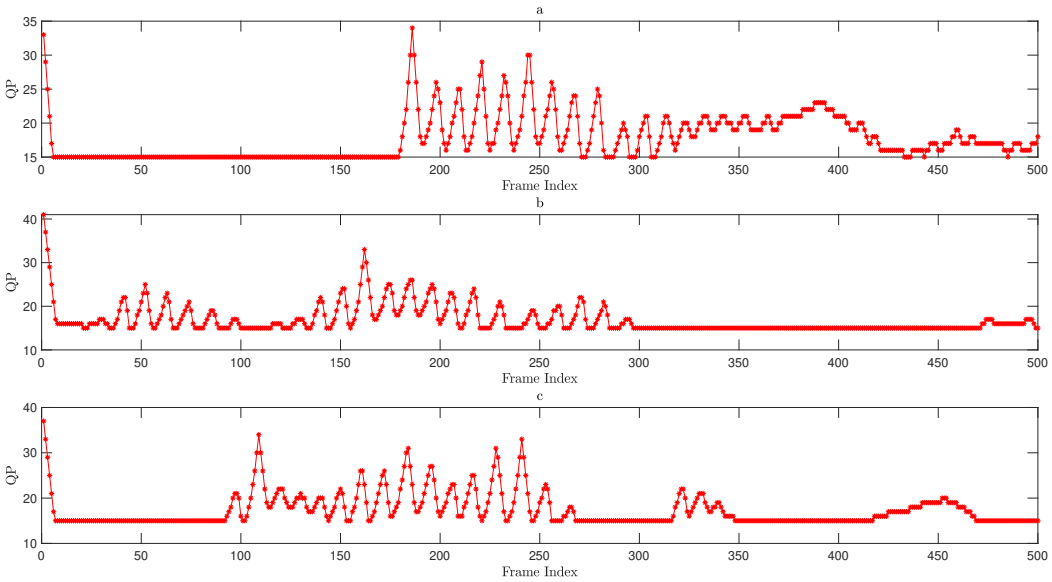

Figure 3.12: The variation of quantification parameter QP in the 4th (a), 6th (b) and 8th (c) transmission episode of the video sequence Magnycours at resolution 640×360 and 25 fps using the BBA [13] algorithm.

rithm BBA. To evaluate the prediction performance of the R-(QP, D) model, we compute the error between the predicted bitrate of the model and the actual bitrate of the frame R_n.

Figure 3.13 - a shows the Error Cumulative distribution function (CDF) of the R-(QP, D) model in the ten transmission episodes. We notice that our rate model R-(QP, D) provides a good performance when estimating its parameters iteratively. 99% of all prediction errors in absolute value are less than 35%.

Figure 3.13 - b illustrates the histogram of prediction errors of the same video sequence. Most prediction errors are between -40% and +40%. Our model performance is still very reliable when its parameters are recursively estimated, with the advantage of reducing the computational complexity of the estimation.

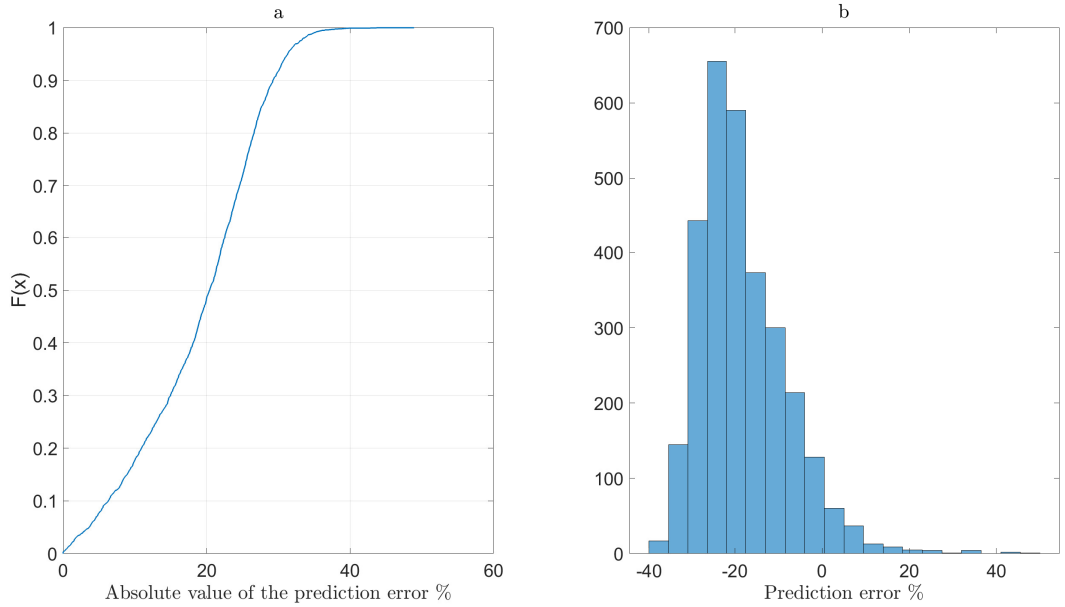

Figure 3.13: (a) Error Cumulative distribution function (CDF) and (b) Histogram of model R-(QP, D) prediction errors when $\mathbf{p}_n$ is determined iteratively, in ten transmission tests of the video sequence Magnycours at resolution 640×360 and 25 fps using the BBA [13] algorithm.

3.6 Conclusion

In this chapter, we present a novel inter-dependent Rate-QP model. Our model describes the relation between the quantization parameter QP_n used to encode the frame n, its bitstream size R_n, and the MSE distortion D_{n-1} of the reference frame. The R-(QP, D) is beneficial when adjusting the QP of the frame according to the allocated bitrate budget in case of low latency live streaming. The optimal quantification parameters QP_n for encoding the n-th frames can be determined based on the distortion D_{n-1} of the reference frame and the target size of the frame n to be encoded.

We evaluate the performance of the proposed model in two coding scenarios: encoding at constant QP and encoding with time-varying QP. In the second scenario, the QP variation is chosen to simulate the case of video coding for transmission on an unstable transmission channel, where the QP changes after every few frames following a drop or increase in transmission rate.

In both coding scenarios, the proposed model outperforms other models

from the literature. For instance, in the video sequence *Tango*, up to 90% of all prediction errors are inferior to 8.6% when using constant QP encoding, and 90% of all prediction errors are inferior to 12% when using variable QP encoding.

The gains are especially significant at low bitrates (*i.e.*, high QPs), showing the benefits of accounting for the distortion of the reference frame in our proposed model. Besides, this attribute shows that our model is exceptionally reliable in the case of coding for live steaming at low transmission rates or when sudden drops occur.

The next chapter will focus on integrating the proposed model in a bitrate adaptation scheme for low-latency video streaming. We will first propose a bitrate adaptation algorithm that determines the maximum rate budget for the frame to be transmitted, according to the available network resources. Then, using our model, we determine the optimal QP to encode the frame such that the resulting bitstream size will not surpass the allocated bitrate budget of the frame.

Chapter 4

Model Predictive Video Bitrate Control for Low-Delay Live Streaming

4.1 Introduction

In applications such as low latency live streaming, such as remote control of a drone or broadcasting sports events, the camera acquiring the scene transmits its compressed stream over a wireless network to the client, *e.g.*, a processing unit. Mobility induces significant and fast variations of the wireless channel characteristics. In such a context, encoder-side control approaches appear better suited for adapting the video encoding bitrate to the wireless channel and network characteristics. In addition, a finer adaptation granularity and a small reception buffer size at the client is necessary to achieve low delay.

This chapter proposes an encoding rate control algorithm adapted for low-latency live streaming applications with glass-to-glass delay targets of 100 to 200 ms. The control is performed at the frame level, which requires the use of the R-(QP, D) model introduced in Chapter 3, to describe the size of the encoded frame R_n as a function of its quantization parameter QP_n and of the distortion of the previous frame D_{n-1}. Using measurements of the channel and network characteristics, a Model Predictive Control (MPC) approach is employed to infer the future client playback margin for different choices of target encoding rates for the next frame to compress and transmit. The controller can then select the appropriate value of QP for that frame.

The contributions of this chapter are the following:

- In Section 4.2 we present a server-driven live streaming architecture for ultra-low latency video delivery.

- In Section 4.3 we propose a Model Predictive Control (MPC) approach to determine the encoding rate of each video frame to be transmitted using information related to the server (transmitter) buffer level and the wireless channel characteristics. The R-(QP, D) model presented in Chapter 3 is used to determine the best QP once the target encoding rate of the frame is determined.

- In Section 4.4 the proposed algorithm is compared to four reference algorithms, namely Festive [10], Panda [11], BOLA [12], and BBA [13]. The reference algorithms have been adapted to operate on the server side and at the frame level.

Simulation results, involving real 4G bandwidth traces show that the proposed algorithm outperforms the other algorithms in terms of average PSNR and number of lost frames, and it is able to provide video with a glass-to-glass latency of 120 ms.

4.2 Overview of the proposed low-latency adaptive streaming approach

Figure 4.1 illustrates the components of the proposed server-driven live streaming architecture. The server consists of the following main components: a camera, a video encoder, an encoding rate controller, a transmission buffer, and a transmitter.

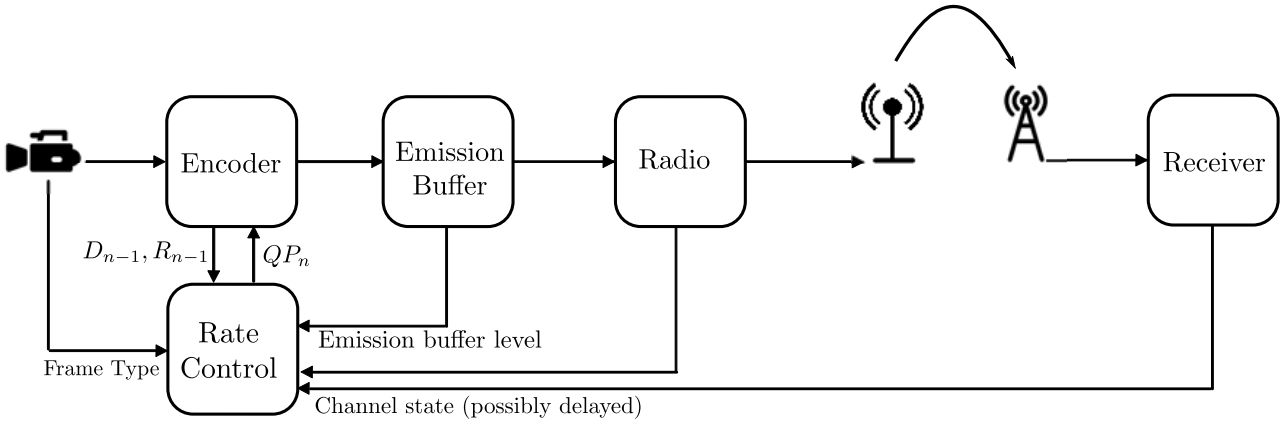

Figure 4.1: Model of the server-driven live streaming architecture.

We assume that video frames are acquired with a period T_f and that the acquisition by the video camera of frame n starts at time $t_n = nT_\mathsf{f}$. The frame acquisition delay depends of the camera aperture and scene illumination. Here, it is assumed constant and equal to T_a. Once frame n is acquired, it is transmitted to the encoder and compressed. For frame n, the trade-off between encoding rate and quality is controlled by the quantization parameter QP_n provided by the rate control module. The encoding delay T_e is assumed constant. The resulting bitstream is segmented into RTP or MPEG2 TS packets and put into the transmission buffer. The packets are drained from the buffer and transmitted via Wifi, 4G, or 5G to some router, eNodeB, or gNodeB [65]. Packets are then carried out through the access and core network to the receiver. This introduces a delay $T_{\mathsf{c},n}$ which depends on the congestion of routers along the path between the eNodeB or gNodeB and the receiver, as well as on the length of this path. At the client side, once all packets related to frame n have been received, decoding starts and introduces a decoding delay T_d. The resulting frames are then temporarily buffered before being displayed at time $t_n + \Delta_\mathsf{p}$, where Δ_p is the acquisition-to-playback delay (*glass-to-glass* delay) [19].

The control is performed so as to prevent the buffer containing decoded frames at receiver from starving. This ensures that frames are displayed on time. If a frame is not decodable, *e.g.*, due to corruption or loss of one of its packets, a frame concealment process is performed [74]. Outdated packets in the transmission buffer, *i.e.*, packets which have no chance to reach the

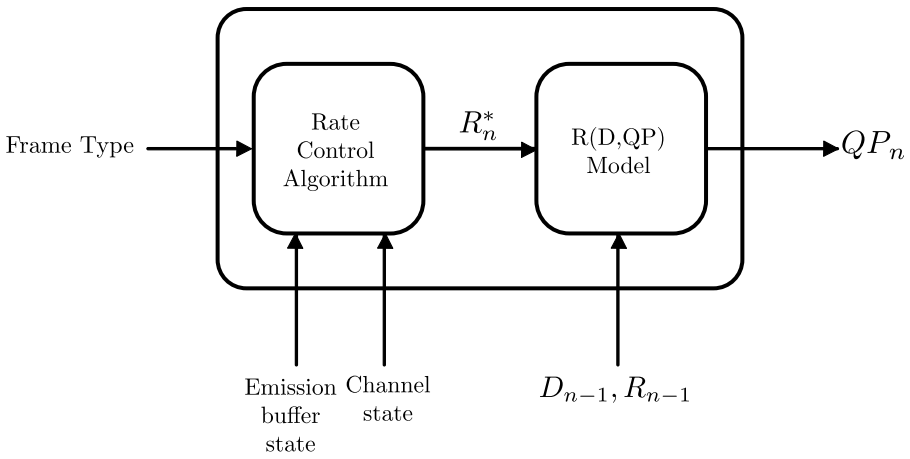

Figure 4.2: The components of the rate control block.

receiver on time are purged, as suggested in [12].

To determine the value QP_n of the quantization parameter for frame n, the rate control module takes as input the amount of bits B_n stored in the transmission buffer as well as some (possibly delayed) channel quality indicator (CQI) provided by the wireless transmitter, see Figure 4.2. The CQI is useful to infer the rate $C(t)$ at which the packets will be transmitted over the wireless channel in the time interval $\left[t_n, t_n + \Delta_{\mathsf{p}}\right]$. In the proposed approach, the rate control block manages an R-(QP, D) model to estimate the encoding rate R_n of the frame n as a function of the distortion D_{n-1} of the previous frame and of QP_n. Since the temporal and spatial characteristics of the frames evolve with time, an online update of the parameters of the R-(QP, D) model is performed using the encoding rates R_{n-i} and distortion D_{n-i} obtained from previously encoded frames, as well as additional encoding trials. Additional information related to frame characteristics (frame type, complexity), which may impact the parameters of the R-(QP, D) model may also be taken into account. Feedback from the client, *e.g.*, in RTCP packets [75], may also be used by the rate controller.

T_f	frame period
$t_n = nT_\mathsf{f}$	start of acquisition of frame n
T_a	frame acquisition duration
B_n	buffer level in bits at transmitter
$T_\mathsf{e} < T_\mathsf{f}$	frame encoding duration
$T_\mathsf{d} < T_\mathsf{f}$	frame decoding duration
R_n	encoding rate of frame n
$C(t)$	channel rate at time t
Δ_p	acquisition-to-playback delay
τ_n	receiver playback time margin for frame n

Table 4.1: Notations used in Section 4.3.

4.3 Model-predictive encoding rate control

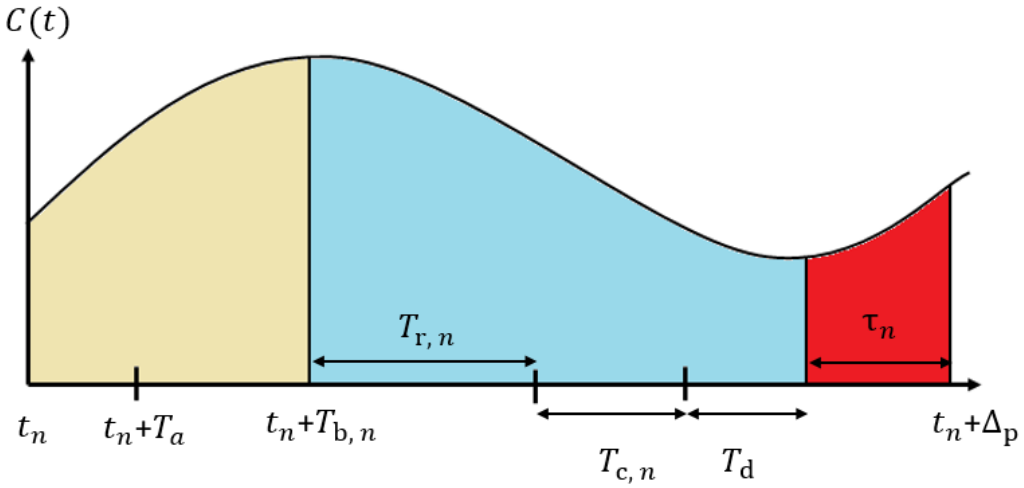

Figure 4.3: The key time instants related to the transmission of frame n.

Figure 4.3 illustrates the different time instants related to the acquisition, processing, transmission, and display of frame n. Frame n starts to be acquired at time t_n. We consider that its encoding starts at $t_n + T_\mathsf{a}$ and ends at $t_n + T_\mathsf{a} + T_\mathsf{e}$. Assume that the encoding rate R_n of frame n has been chosen in the time interval $[t_{n-1} + T_\mathsf{a}, t_n + T_\mathsf{a}[$. Our aim in what follows is to determine the encoding rate R_{n+1} of frame $n+1$ before encoding starts at time $t_{n+1} + T_\mathsf{a}$. For that purpose, we consider an MPC approach in which a model of the evolution of the channel capacity is used to predict the evolution of the playback margins

τ_n and τ_{n+1} of frames n and $n+1$. The rate R_{n+1} will be chosen so as to meet a target playback margin τ^*.

4.3.1 Playback margin of frame n

At time t_n, the transmission buffer contains B_n bits from packets related to previously encoded frames. Assuming that the wireless link capacity is $C(t)$ at time t, the time $T_{\mathsf{b},n}$ required to flush these B_n bits satisfies

$$B_n = \int_{t_n}^{t_n+T_{\mathsf{b},n}} C(t)\, dt. \tag{4.1}$$

With an encoding rate R_n for frame n, $R_n T_{\mathsf{f}}$ bits are generated during the encoding process. We assume that the transmission buffer starts to be fed at the beginning of the encoding process at time $t_n + T_{\mathsf{a}}$ instead of being fed once encoding is finished. Moreover, we assume that the transmission buffer is fed at a rate larger than the channel capacity over the encoding time interval $[t_n + T_{\mathsf{a}}, t_n + T_{\mathsf{a}} + T_{\mathsf{e}}]$, *i.e.*, that $R_n T_{\mathsf{f}}/T_{\mathsf{e}} > C(t)$. Consequently, the transmission buffer does not get empty over the encoding time interval. Consequently, the time $T_{\mathsf{r},n}$ required to drain the $R_n T_{\mathsf{f}}$ bits of the encoded frame n from the transmission buffer is such that

$$R_n T_{\mathsf{f}} = \begin{cases} \int_{t_n+T_{\mathsf{b},n}}^{t_n+T_{\mathsf{b},n}+T_{\mathsf{r},n}} C(t)\, dt & \text{if } T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}, \\ \int_{t_n+T_{\mathsf{a}}}^{t_n+T_{\mathsf{a}}+T_{\mathsf{r},n}} C(t)\, dt & \text{if } T_{\mathsf{b},n} < T_{\mathsf{a}}. \end{cases} \tag{4.2}$$

The first case corresponds to a transmission buffer still containing bits when the video encoder starts to feed bits from frame n, contrary to the second case where all bits from previous frames have been drained before time $t_n + T_{\mathsf{a}}$. The first case better exploits the available channel capacity.

The time $T_{\mathsf{c},n}$ spent by packets in the core network is assumed to evolve slowly with time compared to the evolution of $C(t)$. Consequently, $T_{\mathsf{c},n}$ is assumed constant and known in what follows. When frame n is decoded, it is ready to be displayed by the receiver at time $t_n + \max\{T_{\mathsf{a}}, T_{\mathsf{b},n}\} + T_{\mathsf{r},n} + T_{\mathsf{c},n} + T_{\mathsf{d}}$. Consequently, the playback margin for the n-th frame is

$$\begin{aligned} \tau_n &= t_n + \Delta_{\mathsf{p}} - (t_n + \max\{T_{\mathsf{a}}, T_{\mathsf{b},n}\} + T_{\mathsf{r},n} + T_{\mathsf{c},n} + T_{\mathsf{d}}) \\ &= \Delta_{\mathsf{p}} - (\max\{T_{\mathsf{a}}, T_{\mathsf{b},n}\} + T_{\mathsf{r},n} + T_{\mathsf{c},n} + T_{\mathsf{d}}). \end{aligned} \tag{4.3}$$

4.3.2 Prediction of the playback margin of frame $n+1$

At time $t_n + T_{\mathsf{a}}$, we know that the rate R_n has been chosen to encode frame n. We also know that the acquisition of frame $n+1$ starts at time t_{n+1}. The transmission buffer level B_{n+1} at time t_{n+1} will satisfy

$$B_{n+1} = \begin{cases} \max\left\{0, B_n + R_n T_{\mathsf{f}} - \int_{t_n}^{t_n+T_{\mathsf{f}}} C\left(t\right) dt\right\} & \text{if } T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}, \\ \max\left\{0, R_n T_{\mathsf{f}} - \int_{t_n+T_{\mathsf{a}}}^{t_n+T_{\mathsf{f}}} C\left(t\right) dt\right\} & \text{if } T_{\mathsf{b},n} < T_{\mathsf{a}}. \end{cases} \tag{4.4}$$

At time $t_{n+1} + T_{\mathsf{a}}$, encoding of frame $n+1$ starts. Assuming that a rate R_{n+1} has been chosen for frame $n+1$, one obtains equations satisfied by $T_{\mathsf{b},n+1}$ and $T_{\mathsf{r},n+1}$ from (4.1) and (4.2) as

$$B_{n+1} = \int_{t_{n+1}}^{t_{n+1}+T_{\mathsf{b},n+1}} C\left(t\right) dt \tag{4.5}$$

and

$$R_{n+1} T_{\mathsf{f}} = \begin{cases} \int_{t_{n+1}+T_{\mathsf{b},n+1}}^{t_{n+1}+T_{\mathsf{b},n+1}+T_{\mathsf{r},n+1}} C\left(t\right) dt & \text{if } T_{\mathsf{a}} \leqslant T_{\mathsf{b},n+1}, \\ \int_{t_{n+1}+T_{\mathsf{a}}}^{t_{n+1}+T_{\mathsf{a}}+T_{\mathsf{r},n+1}} C\left(t\right) dt & \text{if } T_{\mathsf{a}} > T_{\mathsf{b},n+1}. \end{cases} \tag{4.6}$$

The playback margin for frame $n+1$ has the same expression as (4.3) and is

$$\begin{aligned} \tau_{n+1} &= t_{n+1} + \Delta_{\mathsf{p}} - \left(t_{n+1} + \max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n+1}\right\} + T_{\mathsf{r},n+1} + T_{\mathsf{c},n+1} + T_{\mathsf{d}}\right) \\ &= \Delta_{\mathsf{p}} - \left(\max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n+1}\right\} + T_{\mathsf{r},n+1} + T_{\mathsf{c},n+1} + T_{\mathsf{d}}\right). \end{aligned} \tag{4.7}$$

We evaluate now $\tau_{n+1} - \tau_n$, the evolution of the playback margin for frames n and $n+1$, assuming that $T_{\mathsf{c},n+1} = T_{\mathsf{c},n}$

$$\begin{aligned} \tau_{n+1} - \tau_n &= \Delta_{\mathsf{p}} - \left(\max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n+1}\right\} + T_{\mathsf{r},n+1} + T_{\mathsf{d}}\right) - \Delta_{\mathsf{p}} - \left(\max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n}\right\} + T_{\mathsf{r},n} + T_{\mathsf{d}}\right) \\ &= \max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n}\right\} - \max\left\{T_{\mathsf{a}}, T_{\mathsf{b},n+1}\right\} + T_{\mathsf{r},n} - T_{\mathsf{r},n+1}. \end{aligned} \tag{4.8}$$

4.3.3 Evaluation of the rate R_{n+1}

To determine the value R_{n+1}^* of R_{n+1} allowing the receiver to observe a playback margin τ_{n+1} equal to τ^*, some additional hypotheses are considered related to the channel capacity $C\left(t\right)$. In what follows, we assume that $C\left(t\right)$ is piecewise constant over time intervals of the form $[t_n, t_n + T_{\mathsf{f}}[$ and equal to C_n.

With this hypothesis, (4.1) and (4.5) become

$$T_{\mathsf{b},n} = \frac{B_n}{C_n} \tag{4.9}$$

and

$$T_{\mathsf{b},n+1} = \frac{B_{n+1}}{C_{n+1}}. \tag{4.10}$$

The expression (4.4) of the buffer level at time t_{n+1} boils down to

$$B_{n+1} = \begin{cases} \max\{0, B_n + (R_n - C_n)\, T_{\mathsf{f}}\} & \text{if } T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}, \\ \max\{0, C_n T_{\mathsf{a}} + (R_n - C_n)\, T_{\mathsf{f}}\} & \text{if } T_{\mathsf{b},n} < T_{\mathsf{a}}. \end{cases} \tag{4.11}$$

Moreover, assuming that even if $T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}$ in (4.2), $t_n + T_{\mathsf{b},n} \in [t_n, t_n + T_{\mathsf{f}}]$, (4.2) becomes

$$T_{\mathsf{r},n} = \frac{R_n}{C_n} T_{\mathsf{f}}. \tag{4.12}$$

This expression is clearly an approximation since the transmission of the bits of encoded frame n may last over several time interval of duration T_{f}. Assuming similarly that $t_{n+1} + T_{\mathsf{b},n+1} \in [t_{n+1}, t_{n+1} + T_{\mathsf{f}}]$, one gets

$$T_{\mathsf{r},n+1} = \frac{R_{n+1}}{C_{n+1}} T_{\mathsf{f}}. \tag{4.13}$$

The evaluation of R_{n+1} starts at $t_n + T_{\mathsf{a}}$. At that time instant, the server is able to determine whether $T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}$ or $T_{\mathsf{b},n} < T_{\mathsf{a}}$ by observing the level of the transmission buffer. In what follows, we assume that $T_{\mathsf{b},n} \geqslant T_{\mathsf{a}}$, which corresponds to the case of a non-empty transmission buffer at $t_n + T_{\mathsf{a}}$, a situation where the channel capacity is fully exploited. Introducing (4.9-4.13) in (4.8), one gets

$$\tau_{n+1} = \tau_n + \frac{B_n}{C_n} - \max\left\{T_{\mathsf{a}}, \frac{\max\{0, B_n + (R_n - C_n)\, T_{\mathsf{f}}\}}{C_{n+1}}\right\} + \left(\frac{R_n}{C_n} - \frac{R_{n+1}}{C_{n+1}}\right) T_{\mathsf{f}}. \tag{4.14}$$

Imposing a target playback margin τ^* for frame $n+1$, from 4.14, we get after some simple derivations the target encoding rate R^*_{n+1} of frame $n+1$

as follows

$$R^*_{n+1} = \frac{\tau_n - \tau^*}{T_\mathsf{f}} C_{n+1} + \frac{B_n}{T_\mathsf{f}} \frac{C_{n+1}}{C_n} - \max\left\{ \frac{T_\mathsf{a}}{T_\mathsf{f}} C_{n+1}, \frac{\max\left\{0, B_n + (R_n - C_n)\, T_\mathsf{f}\right\}}{T_\mathsf{f}} \right\} + \frac{C_{n+1}}{C_n} R_n. \tag{4.15}$$

Assuming further that the transmission buffer will not be empty over the time interval
$[t_n + T_\mathsf{a}, t_{n+1} + T_\mathsf{a}]$, (4.15) boils down to

$$\begin{aligned} R^*_{n+1} &= \frac{\tau_n - \tau^*}{T_\mathsf{f}} C_{n+1} + \frac{B_n}{T_\mathsf{f}} \frac{C_{n+1}}{C_n} - \frac{B_n + (R_n - C_n)\, T_\mathsf{f}}{T_\mathsf{f}} + \frac{C_{n+1}}{C_n} R_n \\ &= \frac{\tau_n - \tau^*}{T_\mathsf{f}} C_{n+1} + \left(\frac{C_{n+1}}{C_n} - 1 \right) \left(\frac{B_n}{T_\mathsf{f}} + R_n \right) + C_n. \end{aligned} \tag{4.16}$$

The evaluation of R^*_{n+1} using (4.16) is performed at the server side. The playback margin τ_n for frame n is observed at client side at time $t_n + \Delta_\mathsf{p} - \tau_n$. Even if R_n has been chosen to get a playback margin τ^* for frame n, due to the discrepancies between R^*_n and the obtained encoding rate R_n and between the channel capacity estimate $\widehat{C}_n$ used at time $t_{n-1} + T_\mathsf{a}$ to evaluate R^*_n and that experienced over the time interval $[t_n, t_n + T_\mathsf{f}]$, the actual playback margin τ_n is likely to differ from τ^*. Consequently, the server needs an estimate $\widehat{\tau}_n$ of τ_n to be able to calculate R^*_{n+1}. It also use estimates $\widehat{C}_n$ and $\widehat{C}_{n+1}$ of the channel rates C_n and C_{n+1}. Designing an effective channel rate estimator is essential, but we focus only on the bitrate adaptation algorithm in this work and assume that estimators are given to us, *e.g.*, using tools such as those described in [14, 76]. Then, (4.16) becomes

$$R^*_{n+1} = \frac{\widehat{\tau}_n - \tau^*}{T_\mathsf{f}} \widehat{C}_{n+1} + \left(\frac{\widehat{C}_{n+1}}{\widehat{C}_n} - 1 \right) \left(\frac{B_n}{T_\mathsf{f}} + R_n \right) + \widehat{C}_n \tag{4.17}$$

with $\widehat{\tau}_n$ obtained introducing (4.9) and (4.12) in (4.3) to get

$$\widehat{\tau}_n = \Delta_\mathsf{p} - \left(\frac{R_n T_\mathsf{f} + B_n}{\widehat{C}_n} + T_{\mathsf{c},n} + T_\mathsf{d} \right). \tag{4.18}$$

Some insights on (4.17) may be obtained considering the target number of bits

$$R^*_{n+1} T_\mathsf{f} = (\widehat{\tau}_n - \tau^*)\, \widehat{C}_{n+1} + \left(\widehat{C}_n T_\mathsf{f} - R_n T_\mathsf{f} \right) + \left(\frac{\widehat{C}_{n+1}}{\widehat{C}_n} - 1 \right) B_n + \frac{\widehat{C}_{n+1}}{\widehat{C}_n} R_n T_\mathsf{f} \tag{4.19}$$

allocated to frame $n+1$ to reach a playback margin equal to τ^*. If $\widehat{\tau}_n > \tau^*$, the estimated playback margin for frame n is larger than the target, and the first term in (4.19) shows that more bits may be used to represent frame $n+1$. The second terms of (4.19) indicates that more bits may be used to represent frame $n+1$ when $\widehat{C}_n T_\mathsf{f} > R_n T_\mathsf{f}$, *i.e.*, when more bits are drained from the transmission buffer than those fed by the encoding of frame n. The third term of (4.19) translates the allowed rate increase or decrease due to a more or less efficient drain of the bits present in the transmission buffer at time t_n. The last term of (4.19) corresponds to the number of bits to be transmitted in steady-state. If $\widehat{C}_{n+1} > \widehat{C}_n$, the target rate can increase and else has to decrease.

4.4 Performance Evaluation

This section compares the proposed model predictive encoding rate control algorithm with reference rate-based and buffer-based schemes.

4.4.1 Simulation setup

We consider a simulation setup consisting of a server and a client as described in Section 4.2. The server receives the video frames to encode, runs the x265 encoder [7], and feeds the transmission buffer with encoded packets. It also manages the R-(QP, D) model introduced in Chapter 3 and the rate control algorithm, described in Section 4.3. The client contains a reception buffer, an HEVCdecoder (the HEVC Test Model HM16 [8]), and a decoded frame buffer. The access and core networks are simulated using 4G bandwidth traces taken from [9].

Six video sequences are used. Five of them belong to the JVET test sequences: *CrowdRun*, *ParkJoy*, *TouchDownPass*, *DaylightRoad2*, and *KristenandSara* [72], An additional sequence, *Magnycours* [73], acquired from a camera on a race car is also considered. The video sequences are sub-sampled using FFmpeg [77] to two spatial resolutions 640×360 and 1280×720, and to one temporal resolution of 25 fps, *i.e.*, $T_\mathsf{f} = 40$ ms. The acquisition delay is taken as $T_\mathsf{a} = 2$ ms.

The x265 encoder [7] is configured in low delay mode and with an intra-

refresh cycle of one second. Encoded data packets are embedded in RTP/UDP/IP packets and temporarily stored in the transmission buffer. Wireless transmission of packets is simulated with a period of one millisecond. For that purpose, the 4G trace *A_2018.01.27_10.58.49.csv*, taken from the set of traces described in [9], has been considered. This 15 mn long trace has been acquired using GNetTrack Pro [78] around Cork within a moving car. Downlink (DL) transmission rates are available with a measurement period of one second. We assume that similar rates are available in the Uplink (UL) direction, even if the allocation between UL and DL is not symmetric. Moreover, the transmission rates have been spline interpolated to one millisecond for the transmission simulation. Transmission is assumed to be loss-free thanks to HARQ mechanisms between the transmitter and the base station. The time $T_{c,n}$ spent by packets in the core network is neglected.

Received packets are temporarily stored in the client reception buffer. Decoding starts upon reception of the last packet related to the considered frame. The decoding time is taken as $T_d = 20$ ms. Decoded frames are stored in a decoded frame buffer before their display, at time $t_n + \Delta_p$ for frame n, where Δ_p is the playback delay. When a frame is not available in the display buffer, a simple concealment process is realized. Lost frames are replaced by the last correctly decoded frame, which leads however to a significant loss in PSNR. More sophisticated concealment mechanisms could be considered with a larger complexity, see, *e.g.*, [74].

For each sequence, the value of vector of parameters $\mathbf{p}_n$ for frame n of the R-(QP, D) model is estimated iteratively using the method in Section 3.3 of Chapter 3. We use three encoders operating with time-varying QPs to provide data to the estimator. The choice of QP for frame n and for encoder $i = 1, \ldots, 3$ is performed as follows

$$QP_{n,i} = \begin{cases} QP_{0,i} & \text{if } n = 0 \\ QP_{n-1,i} + \Delta QP_i & \text{if } n\%4 = 1, 2 \\ QP_{n-1,i} - \Delta QP_i & \text{if } n\%4 = 3, 0, \end{cases} \tag{4.20}$$

where $n\%4$ is the remainder of the division of n by 4. The vector $QP_0 = (24, 36, 40)$ contains the QPs of the first frame, and $\Delta QP = (4, 4, -4)$ is the variation of QP. We also set $w_{n,m} = 1/R_{n,m}$ and $\alpha = \lambda_1(\mathbf{W}_n\mathbf{X}_n^T\mathbf{X}_n)/100$ in Eq.

3.10.

The interpolated bandwidth trace is used to simulate the transmission of video packets in the network with a time period of 1 millisecond, and by the rate adaptation algorithms to get a measurement of the current transmission rate before selecting the encoding trate of each frame. All five rate adaptation algorithms use the same bandwidth estimator for their decisions.

4.4.2 Reference Algorithms

The proposed encoding rate control algorithm is compared to four rate adaptation algorithms from the literature: Festive [10], Panda [11], BOLA [12], and BBA [13]. To ensure fair performance comparison, all these algorithms have been adapted to operate at the server side and to adjust the target encoding rate at a frame level. All algorithms share the same R-(QP, D) model.

Festive and Panda, in their client-side implementation use a bandwidth estimator to predict the transmission rate from past downloads and adjust the target encoding rate of the frame/chunk accordingly. Festive decreases the target encoding rate as soon as a reduction of the transmission rate is detected, and increases the encoding rate slowly when the transmission rate improves. Contrarily, Panda increases the encoding rate more aggressively when an improvement in the transmission rate is detected. Panda and Festive have been implemented as specified in [10] and [11] respectively. Both algorithms have been implemented on the server side. The bitrate of the video is adjusted at the frame level by selecting the target frame encoding rate considering the last available measurement of the transmission rate from the bandwidth trace.

BOLA, in its client-side implementation [12], selects the target encoding rate of each frame according to the level of the client reception buffer. As the throughput of the network varies, BOLA uses Lyapunov optimization to maximize video quality and minimize rebuffering events at the client-side. In the proposed server-side variant of BOLA, the server estimates the level of the reception buffer of the client. For that purpose, the playback delay Δ_{p} is assumed constant during the streaming session. Neglecting packets in the access and core networks, the number of frames $Q_{\mathrm{c},n}$ in the client buffer is

estimated from the number of frames $Q_{\mathsf{t},n}$ in the transmission buffer as

$$\begin{cases} \widehat{Q}_{\mathsf{c},n} = \frac{\Delta_{\mathsf{p}}}{T_{\mathsf{f}}} - Q_{\mathsf{t},n} & \text{if } nT_{\mathsf{f}} > \Delta_{\mathsf{p}} \\ \widehat{Q}_{\mathsf{c},n} = n\text{–}Q_{\mathsf{t},n} & \text{otherwise.} \end{cases} \tag{4.21}$$

$\widehat{Q}_{\mathsf{c},n}$ is then used by BOLA to adjust the encoding rate of each frame at server side. The same logarithmic utility and cost functions proposed in [12] are used for the considered implementation of BOLA.

Festive, Panda, and BOLA were originally designed to select a target encoding rate in a discrete set of rates. Since the proposed algorithm selects a target rate from a continuous interval, a large number of target bitrates has been considered for the other algorithms to mimic a continuous target bitrate selection. Consequently, 30 target bitrates uniformly spaced in a logarithmic scale between 145 kbps and 75 Mbps have been considered. These rates have been chosen considering typical minimum and maximum available transmission rates observed in the rate traces. Figure 4.4-a summarizes the target encoding rates as a function of $\widehat{Q}_{\mathsf{c},n}$ obtained for BOLA.

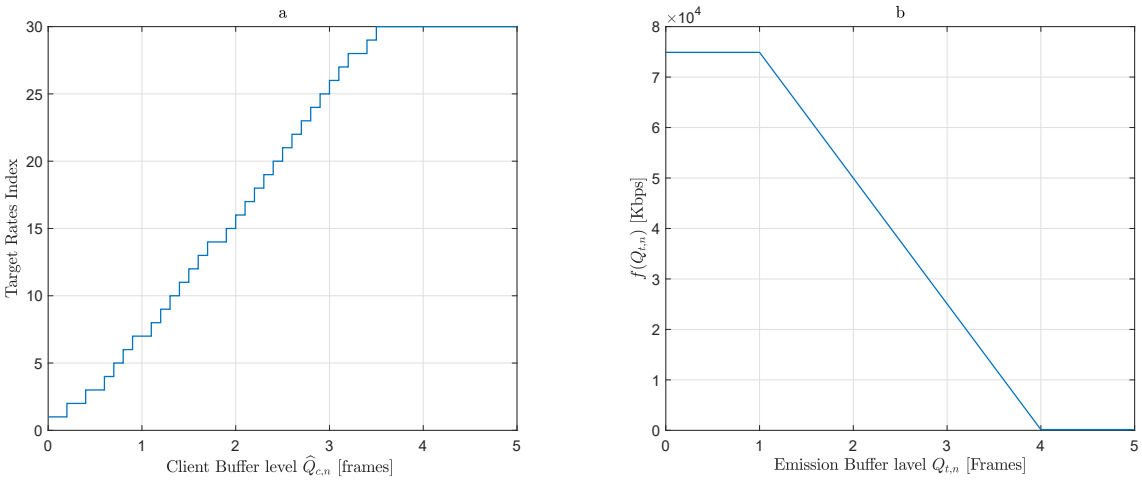

Figure 4.4: (a) Encoding rate index provided by BOLA as a function of $\widehat{Q}_{\mathsf{c},n}$, the estimated number of frames in the client buffer; index 1 corresponds to 145 kbps, while index 30 corresponds to 75 Mbps. (b) Encoding rate provided by BBA as a function of number of frames $Q_{\mathsf{t},n}$ of the transmission buffer, when $\Delta_{\mathsf{p}}/T_{\mathsf{f}} = 5$, $Q_{\min} = 1$, $Q_{\max} = 4$, $R_{\min} = 145$ kbps and $R_{\max} = 75$ Mbps.

The encoding rate provided by BBA is selected based on the level $Q_{\mathsf{t},n}$ of the transmission buffer. A maximum encoding rate $R_{\max}$ is chosen by BBA when

$Q_{\mathsf{t},n} \in [0, Q_{\min}]$ (*i.e.* the client buffer contains many frames) and a minimum encoding rate $R_{\min}$ is chosen when $Q_{\mathsf{t},n} \in \left[Q_{\max}, \Delta_{\mathsf{p}}/T_{\mathsf{f}}\right]$ (*i.e.* the client buffer is almost empty). In the interval $[Q_{\min}, Q_{\max}]$, the encoding rate decreases linearly with $Q_{\mathsf{t},n}$, see Figure 4.4-b. The choice of the values of $Q_{\min}$, $Q_{\max}$, $R_{\min}$, and $R_{\max}$ depends on the application and impact directly the QoE. In our experiments, we set $Q_{\min} = 0.2\Delta_{\mathsf{p}}/T_{\mathsf{f}}$ and $Q_{\max} = 0.8\Delta_{\mathsf{p}}/T_{\mathsf{f}}$. $R_{\min}$ and $R_{\max}$ are chosen as 145 kbps and 75 Mbps, respectively. For instance, with $\Delta_{\mathsf{p}} = 200$ ms and $T_{\mathsf{f}} = 40$ ms, $Q_{\min} = 1$ frame, $Q_{\max} = 4$ frames and BBA selects a conservative encoding rate when the client buffer contains less than one frame.

The R-(QP, D) model is used by all rate adaptation algorithms to obtain the target QP of each frame from the selected target encoding rates.

4.4.3 Evaluation Metrics

All encoding rate control algorithms have been compared based on the following metrics, also used in [56].

One considers the average PSNR over all frames

$$\overline{P} = \frac{1}{N} \sum_{n=0}^{N-1} P_n, \tag{4.22}$$

where N is the number of frames in the sequence, and P_n is the PSNR in dB of the displayed frame n compared to the original frame n. The average PSNR variation is evaluated using the mean absolute value of the PSNR difference between two consecutive frames

$$\overline{|\Delta P|} = \frac{1}{N-1} \sum_{n=1}^{N-1} |P_n - P_{n-1}| . \tag{4.23}$$

Finally, one also considers the number of lost frames L during the streaming session.

4.4.4 Performance analysis of the proposed MPC algorithm

First, the performance of the proposed model predictive rate control algorithm is evaluated for different values of the initial playback delay Δ_{p} ranging

from 120 ms to 240 ms, and different target playback margins τ^*. Results are described for the *Park Joy* sequence. Similar results are obtained for the other sequences.

Tables 4.2 and 4.3 summarize the obtained results at resolutions 640×360 and 1280×720 respectively.

First, as expected, a larger playback delay ($\Delta_p = 200$ ms or $\Delta_p = 240$ ms) leads to fewer losses, since transmission rate variations are better handled. Similarly, large values of τ^* provide also better performance. For example, when $\Delta_p = 200$ ms and $\tau^* = 160$ ms, the control is performed so as to provide four frames in the client buffer. In such a regime, the MPC algorithm manages to adjust the bitrate of the frames without causing any frame loss. Nevertheless, large values of τ^* lead to conservative encoding rate selections, which decreases the average PSNR of the decoded video.

When τ^* is too small, frames may be lost. The MPC algorithm becomes less conservative and tries to better exploit the available transmission rate by encoding frames with a higher bitrate. Nevertheless, due to the inaccuracy of the R-(QP, D) model, the bitrate of the encoded frame may be higher than the target bitrate. Similarly, the transmission rate considered for the encoding rate selection may be larger than the actual transmission rate, which induces a larger transmission delay than expected. The playback margin τ^* aims at compensating these discrepancies. For a video at 1280×720, $\tau^* = 80$ ms provides good results even with a very small end-to-end playback delay of $\Delta_p = 120$ ms.

Δ_p (ms)	120	120	120	160	160	160	160	160	200	200	200	200	200	240	240	240
τ^* (ms)	20	40	80	10	20	40	60	80	20	40	80	120	160	20	40	80
$\overline{P}$	33.39	33.01	29.89	25.21	34.26	34.25	34.15	33.37	34.17	34.27	34.25	33.37	29.92	31.32	34.29	34.27
$\overline{\lvert\Delta P\rvert}$	0.95	1.25	1.20	1.05	0.92	0.92	0.95	1.24	0.93	0.92	0.93	1.24	1.23	0.93	0.92	0.91
L	3	2	0	63	1	0	0	0	1	0	0	0	0	16	0	0

Table 4.2: Performance of the MPC algorithm for *Parkjoy* at resolution 640×360 considering $\overline{P}$, the average PSNR in dB, $\overline{|\Delta P|}$, the average of the absolute value of the PSNR variation of consecutive frames in dB, and L the number of lost frames.

Δ_p (ms)	120	120	120	160	160	160	160	160	200	200	200	200	200	240	240	240
τ^* (ms)	20	40	80	10	20	40	60	80	20	40	80	120	160	20	40	80
$\overline{P}$	22.32	25.37	24.06	20.77	23.11	25.18	26.85	26.29	24.56	26.83	27.02	26.29	24.07	23.43	26.69	27.15
$\overline{\lvert\Delta P\rvert}$	0.88	0.82	0.53	0.85	0.73	0.69	0.73	0.77	0.70	0.63	0.66	0.77	0.54	0.73	0.64	0.61
L	65	20	0	98	44	14	3	0	38	6	1	0	0	54	8	0

Table 4.3: Performance of the MPC algorithm for *Park Joy* at resolution 1280×720 considering $\overline{P}$, the average PSNR in dB, $\overline{|\Delta P|}$, the average of the absolute value of the PSNR variation of consecutive frames in dB, and L the number of lost frames.

Figures 4.5-a , 4.5-b, show the evolution of the target and actual encoding rates for the *Parkjoy* sequence at resolution 640×360 when $\tau^* = 40$ ms. Figures 4.6-a and 4.6-b show similar results for the same video sequence when $\tau^* = 160$ ms.

Imposing a large τ^* implies a conservative use of the channel capacity: When $\tau^* = 160$ ms, the selected target rate is always less than the transmission rate. Conversely, when $\tau^* = 40$ ms, see Figure 4.5, the target encoding rates selected by the MPC approach are close to the available transmission rates. Smaller values of τ^* lead to a better exploitation of the available channel capacity.

Figure 4.6-d shows that the client buffer contains about 4 frames all the time when $\tau^* = 160$ ms, thus setting large τ^* provides large margin to react to sudden drops of the transmission rate. Conversely, as shown in Figure 4.5-d, the client buffer contains only 1.5 frames when $\tau^* = 40$ ms. Accordingly, τ^* should be chosen as a trade-off between bandwidth exploitation and a protection against sudden drops of the transmission rate.

Figure 4.5-e shows the evolution of the estimated and actual values of τ , as well as the estimated and actual values of the flushing delay of the transmission buffer $T_{\mathrm{b},n}$, when $\Delta_p = 200$ ms and $\tau^* = 40$ ms. Figure 4.6-e shows similar results when $\tau^* = 160$ ms. The estimates of τ and $T_{\mathrm{b},n}$ are quite accurate when using the last measure of the transmission rate as described in 4.5-f and 4.6-f. The proposed algorithm has difficulties to maintain the actual value of τ at the level of the target playback margin τ^* when τ^* is too small or too close to Δ_p due to coded packets stored at the transmission buffer and packets passing through the network. The time required to transmit the packets varies due to the estimation error of $\widehat{C}_n$ used in the determination of the target frame encoding rate R_n^*.

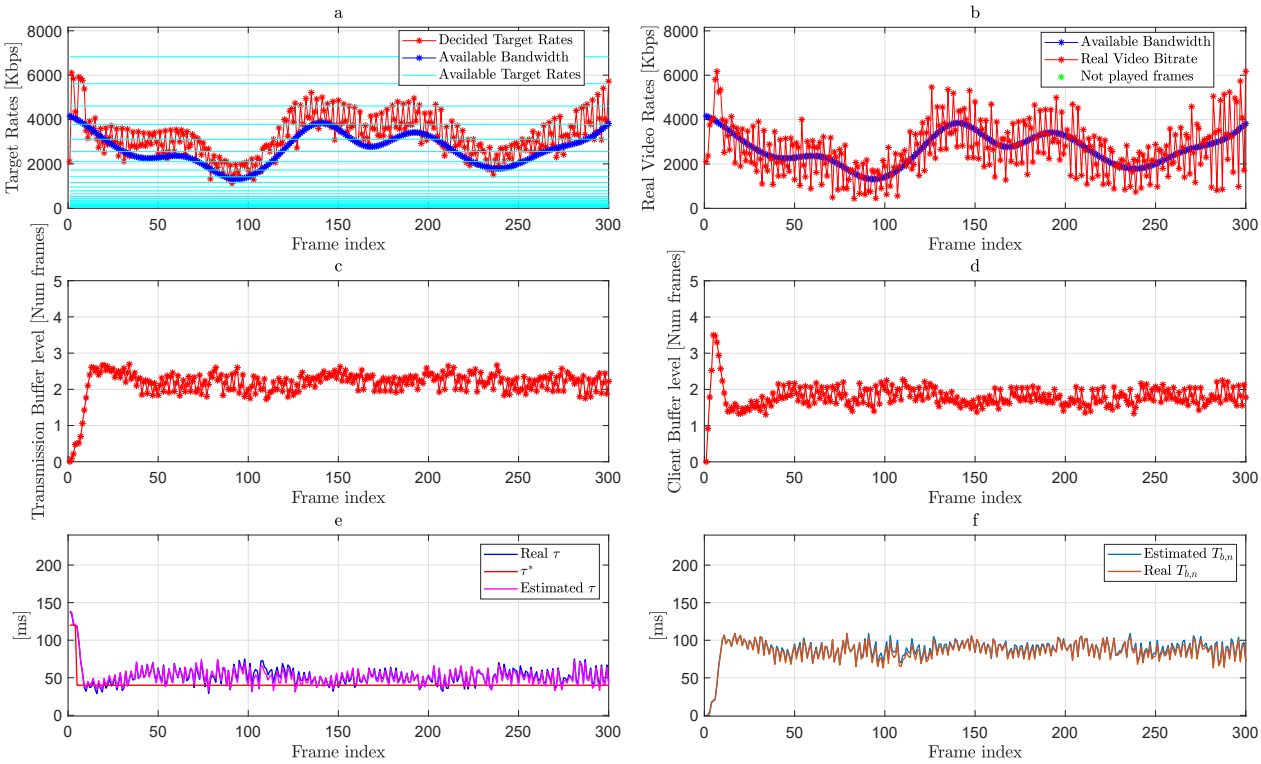

Figure 4.5: Proposed approach: Evolution of the target rates and transmission rates (a), actual frame encoding rates (b), transmission buffer level (c), client buffer level (d), actual and estimated value of τ (e), actual and estimated values of $T_{\mathrm{b},n}$ (f) for the *Park Joy* sequence at 640×360 when $\Delta_{\mathrm{p}} = 200$ ms and $\tau^* = 40$ ms.

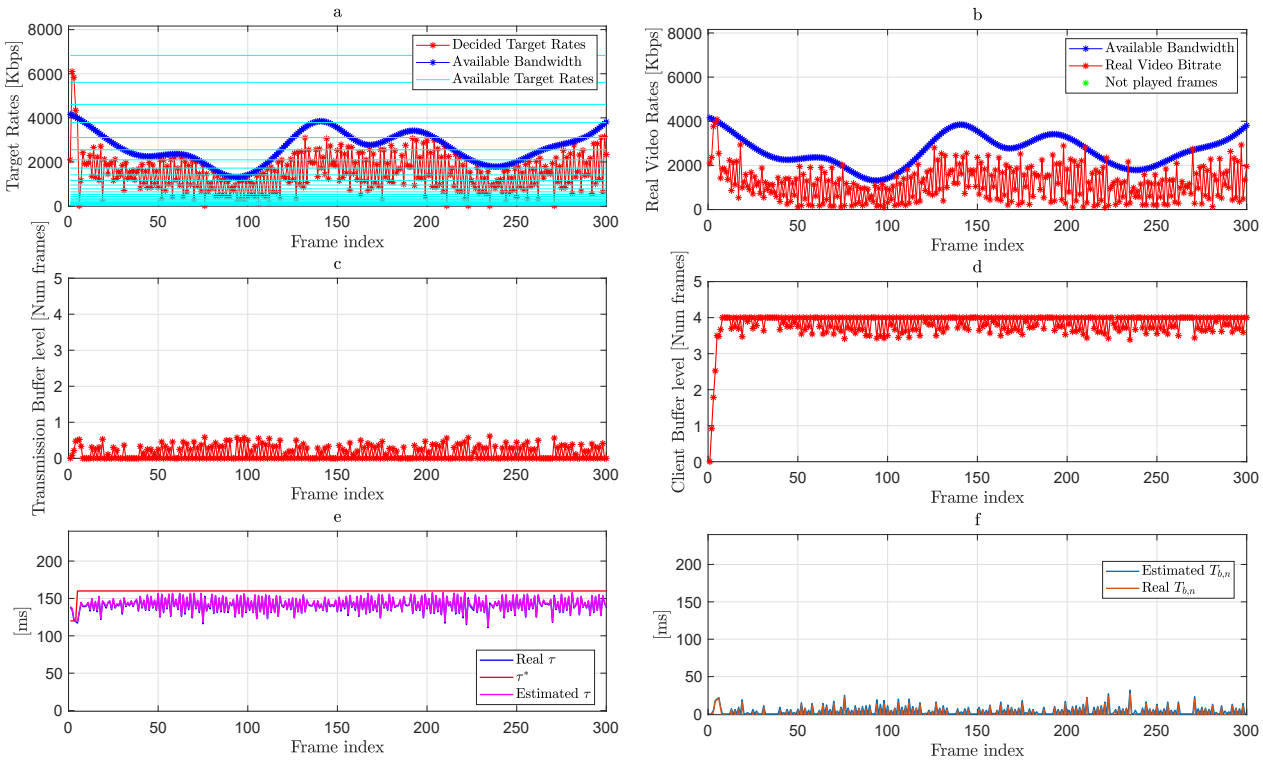

Figure 4.6: Proposed approach : Evolution of : the target rates and transmission rates (a), actual frame encoding rates (b), transmission buffer level (c), client buffer level (d), actual and estimated value of τ (e), actual and estimated values of $T_{\mathrm{b},n}$ (f) for the *Park Joy* sequence at 640×360 when $\Delta_{\mathrm{p}} = 200$ ms and $\tau^* = 160$ ms.

4.4.5 Performance comparison with state-of-the-art reference algorithms

The proposed rate control algorithm is compared to Festive [10], Panda [11], BOLA [12], and BBA [13]. Results are average over ten transmission episodes for each video sequence. Each episode considers a different initial time instant in the considered bandwidth trace. The initial playback delay is set to $\Delta_p = 200$ ms for all algorithms. For the proposed algorithm, the target playback margin is set to $\tau^* = 50$ ms when the frame resolution is 640×360 and to $\tau^* = 80$ ms when it is 1280×720.

Table 4.4 summarizes the results obtained with each rate control algorithm and when the frame size is 640×360 and 1280×720.

The proposed algorithm provides the best performance in terms of average PSNR and lost frames for all tested video sequences in both resolutions. The largest frame loss for the proposed algorithm is obtained with the *Park Joy* sequence at resolution 1280×720, where 5 frames are lost among 3000 transmitted frames. The cause of a larger number of lost frames is due to a reduced accuracy of the R-(QP, D) model for some frames. The model accuracy decreases when time variations in the video sequence are high. Due to the large activity in the video sequence, it is challenging to iteratively estimate the parameters of the R-(QP, D) model.

The price to be paid is a relatively large variability with time of the PSNR of encoded frames. The smallest variability in PSNR is obtained by Panda and Festive, which are both bandwidth-based algorithms. They select the encoding rate based on available transmission rate only. On the contrary, the other algorithms try to stabilize buffer levels. This creates oscillations of the encoding rates and then of the PSNR. The proposed algorithm has an average PSNR variation less than that of BOLA. This variation is for most sequences less than 1 dB, which is usually unnoticeable by observers.

Compared to the proposed algorithm, BOLA achieves a slightly lower average PSNR quality and more lost frames. When BOLA achieves 0 lost frames, it comes at the cost of a lower average PSNR and larger PSNR variations.

The BBA algorithm has the worst performance as it tends to be too aggressive by selecting a high encoding rate when the buffer level allows it. This causes a large number of frame losses, especially when the R-(QP, D) model is less accurate. The variations of the PSNR when using BBA is usually smaller

		640×360			1280×720		
Sequence	Method	L	$\overline{P}$	$\overline{\lvert\Delta P\rvert}$	L	$\overline{P}$	$\overline{\lvert\Delta P\rvert}$
CrowdRun	MPC	**0**	**33.35**	0.55	**0**	**28.26**	0.30
	BBA	1	33.25	0.35	**0**	28.25	0.18
	Festive	6	29.92	0.17	**0**	26.90	**0.13**
	Panda	**0**	30.80	**0.16**	**0**	26.86	**0.13**
	BOLA	**0**	32.99	1.25	**0**	28.15	0.46
ParkJoy	MPC	**0**	**34.27**	0.92	5	**27.03**	0.65
	BBA	5	32.97	0.59	20	26.81	0.62
	Festive	**0**	31.86	**0.27**	7	24.93	**0.42**
	Panda	**0**	31.74	**0.27**	5	24.90	**0.42**
	BOLA	**0**	33.84	1.27	**1**	26.28	0.71
Magnycours	MPC	**0**	**46.42**	0.35	**1**	**40.26**	0.42
	BBA	131	34.77	0.26	232	24.54	0.18
	Festive	**0**	45.96	**0.12**	70	36.50	0.27
	Panda	**0**	46.02	**0.12**	69	35.20	0.25
	BOLA	**0**	46.13	0.85	9	33.00	0.34
TouchDownPass	MPC	**1**	**44.58**	0.61	**0**	**40.14**	0.46
	BBA	95	27.98	0.64	34	37.26	0.40
	Festive	36	37.99	0.32	6	38.73	0.33
	Panda	24	40.20	**0.27**	**0**	38.74	**0.32**
	BOLA	0	44.38	0.92	**0**	39.74	0.50
DaylightRoad2	MPC	**0**	**44.32**	0.33	**0**	**40.10**	0.26
	BBA	**0**	44.31	0.19	**0**	40.09	0.14
	Festive	**0**	42.87	**0.11**	**0**	38.87	**0.08**
	Panda	**0**	42.72	**0.11**	**0**	38.79	**0.08**
	BOLA	**0**	44.10	0.88	**0**	39.93	0.55
KristenandSara	MPC	**0**	**48.28**	0.08	**0**	**44.18**	0.24
	BBA	**0**	48.27	0.07	**0**	44.17	0.13
	Festive	**0**	47.64	**0.09**	**0**	43.43	**0.09**
	Panda	**0**	47.58	**0.09**	**0**	43.38	**0.09**
	BOLA	**0**	48.20	0.10	**0**	44.05	0.30

Table 4.4: Average performance of the proposed algorithm compared to Festive [10], Panda [11], BOLA [12], and BBA [13] when the videos have a resolution of 640×360 and 1280×720; L is the number of lost frames, $\overline{P}$ is the average PSNR in dB, and $\overline{\lvert\Delta P\rvert}$ is the average of the absolute value of the PSNR variation of consecutive frames in dB

than those of observed with BOLA and the proposed approach.

Table 4.5 summarizes the performances of the tested bitrate adaptation algorithms in terms of the average SSIM of the received sequences, the SSIM variations, and the VMAF score for the tested video sequences. We notice that our proposed algorithm provides the best performance in terms of average SSIM and VMAF scores for the most of the tested video sequences in both resolutions. The proposed MPC algorithm has an SSIM variance with time slightly larger than Panda and Festive. The buffer-based algorithms Bola and BBA have the highest SSIM variance in most of the cases. This large variability of the SSIM is due to frequent oscillation of the selected frame target encoding rate to stabilize buffer levels. Finally, our algorithm provides the highest VMAF score, indicating the best video quality compared to that provided by the other tested bitrate adaptation algorithms. BBA algorithm has the worst recorded VMAF scores. This is because of the frequent oscillations in the target encoding rate and a large number of frame losses.

Figure 4.7 illustrates the evolution with time of the transmission rate, the target encoding rate, and the actual encoding rate for the considered encoding rate adaptation algorithms in the second transmission episode of the *DaylightRoad2* video sequence. Festive and Panda have close behavior in the selection of the target encoding rate. These two rate-based approaches are conservative and select a target encoding rate lower than channel transmission rate. BOLA leads to large rate oscillations. This explains the fact that BOLA has the largest average PSNR variations. The proposed approach and BBA have an overall similar behavior, even if the proposed approach follows better the variations of the channel capacity compared to BBA. In addition, BBA leads to slightly larger oscillations of the PSNR. This has been verified with the other transmission episodes and the other video sequences.

Figure 4.7 illustrates also the accuracy of the R-(QP, D) model: in most of the cases, the actual encoding rate is relatively close to the target encoding rate. This shows that the value of QP determined from the model provides an actual encoding rate close to the encoding rate predicted by the model.

Figure 4.8 shows the evolution of the transmission and client buffer levels for the considered rate adaptation algorithms in the same transmission episodes. Festive and Panda, the two rate-based algorithms, keep the client

		640×360				1280×720			
Sequence	Method	L	average SSIM	SSIM variation	VMAF score	L	average SSIM	SSIM variation	VMAF score
CrowdRun	MPC	0	0.9276	0.0040	96.85	0	0.8213	0.0057	79.08
	BBA	1	0.9276	0.0029	96.80	0	0.8209	0.0038	79.04
	Festive	6	0.8931	0.0019	91.83	0	0.7806	0.0031	69.97
	Panda	0	0.8911	0.0020	91.45	0	0.7799	0.0032	69.74
	BOLA	0	0.9264	0.0091	95.87	0	0.8195	0.0087	78.24
ParkJoy	MPC	0	0.9311	0.0053	96.54	5	0.8061	0.0113	76.25
	BBA	5	0.8913	0.0119	88.62	20	0.7995	0.0114	74.19
	Festive	0	0.9002	0.0026	92.58	7	0.7366	0.0111	60.10
	Panda	0	0.8983	0.0026	91.49	5	0.7351	0.0112	60.08
	BOLA	0	0.9296	0.0075	95.97	1	0.7873	0.0141	71.49
Magnycours	MPC	0	0.9881	0.0007	99.26	1	0.9658	0.0017	95.80
	BBA	131	0.8522	0.0034	31.68	232	0.7800	0.0070	61.59
	Festive	0	0.9870	0.0004	99.12	70	0.9229	0.0028	91.20
	Panda	0	0.9874	0.0004	99.16	69	0.9183	0.0027	91.05
	BOLA	0	0.9876	0.0016	99.11	9	0.9234	0.0023	95.19
TouchDownPass	MPC	1	0.9834	0.0017	99.40	0	0.9528	0.0036	95.33
	BBA	95	0.7821	0.0192	23.60	34	0.9152	0.0047	16.45
	Festive	36	0.9029	0.0049	69.72	6	0.9364	0.0038	77.34
	Panda	24	0.9286	0.0028	79.08	0	0.9368	0.0037	67.69
	BOLA	0	0.9822	0.0026	99.27	0	0.9491	0.0040	64.55
DaylightRoad2	MPC	0	0.9859	0.0007	99.75	0	0.9671	0.0011	76.25
	BBA	0	0.9859	0.0005	99.74	0	0.9671	0.0007	74.19
	Festive	0	0.9824	0.0003	99.70	0	0.9605	0.0005	60.10
	Panda	0	0.9819	0.0003	99.67	0	0.9602	0.0006	60.08
	BOLA	0	0.9856	0.0017	99.69	0	0.9635	0.0025	71.49
KristenandSara	MPC	0	0.9918	0.0001	98.38	0	0.9806	0.0007	97.75
	BBA	0	0.9918	0.0001	98.38	0	0.9806	0.0004	97.75
	Festive	0	0.9911	0.0001	98.30	0	0.9787	0.0003	97.42
	Panda	0	0.9911	0.0001	98.30	0	0.9786	0.0003	97.39
	BOLA	0	0.9917	0.0001	98.36	0	0.9802	0.0008	97.64

Table 4.5: Average performance of the proposed algorithm compared to Festive [10], Panda [11], BOLA [12], and BBA [13] in terms of the average SSIM of the received sequences, the SSIM variability with time, and the VMAF score.

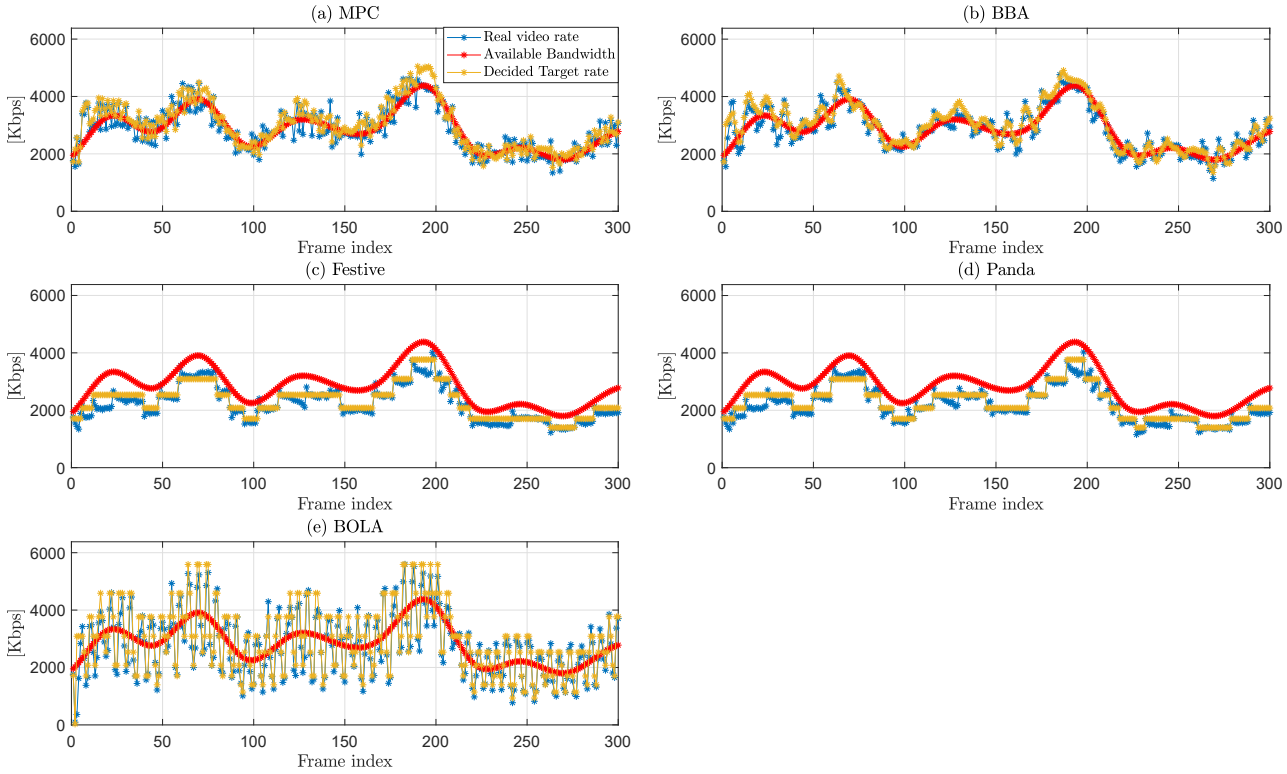

Figure 4.7: Evolution of the transmission rate, the selected target rate, and the actual encoding rate for the proposed algorithm, Festive [10], Panda [11], BOLA [12], and BBA [13] in the second transmission episode of the *DaylightRoad2* sequence at resolution 640×360, when $\Delta_p = 200$ ms and $\tau^* = 50$ ms.

buffer level high. This is due to their conservative behavior, where the encoding rate is selected to avoid an empty client reception buffer. Conversely, the buffer level with BOLA oscillates as the selected target rate is continuously changing. In addition, as BOLA is less conservative, the buffer level has a lower value than with Festive or Panda. The buffer level of BBA oscillates around the same value than that obtained by the proposed approach, but the latter is much more stable.

4.5 Conclusions

This chapter presents a new rate adaptation algorithm for low-latency video streaming applications. The proposed approach employs the MPC framework. It exploits the transmission buffer level and an estimate of the wire-

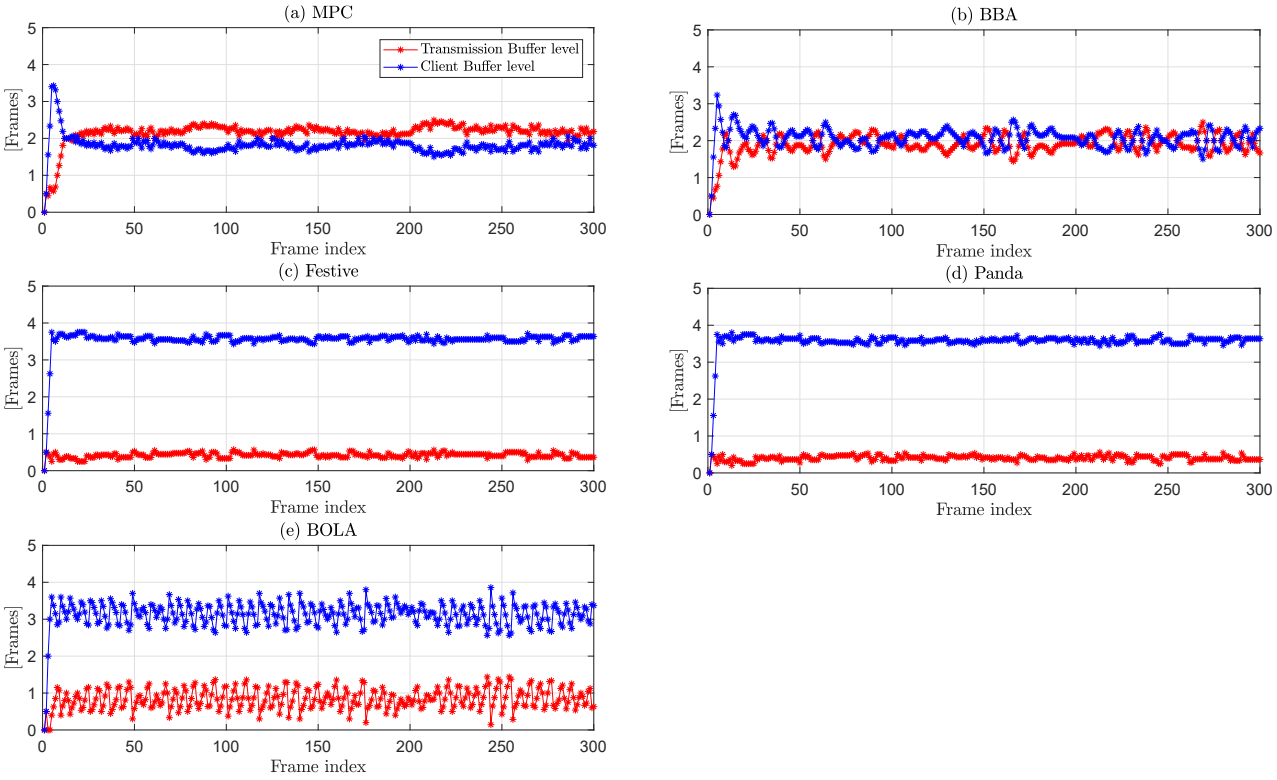

Figure 4.8: Temporal variations of the client buffer level and the transmission buffer level for the the proposed MPC algorithm, Festive [10], Panda [11], BOLA [12], and BBA [13] in the 2nd transmission episode of *DaylightRoad2* sequence at resolution 640×360, when $\Delta_{\mathrm{p}} = 200$ ms and $\tau^* = 50$ ms.

less transmission rate to determine the target encoding rate of each frame. The choice of the quantization parameter for each frame is performed via an R-(QP, D) model, able to predict the size of the current encoded frame as a function of its quantization parameter and the distortion of the previous frame.

The performance of the proposed rate adaptation approach is compared to four reference algorithms considering a streaming application with an end-to-end latency less than 200 ms. The proposed approach outperforms these algorithms in both average PSNR and frame losses. The price to be paid is a slightly larger variability with time of the PSNR of each frame.

The performance of the proposed model depends on the estimation quality of the transmitter buffer level and of the transmission rate. Using measurements acquired using tools such as GNetTrack Pro, this information is only available with a period of about one second. Tools such as QXDM [79] or MobileInsight [80], give access to messages excahnged at the PHY and MAC layer of the proptocol stack and are able to provides such information with a higher frequency.

In the next chapter, we approach another component of the transmission chain that significantly affects the transmission latency. The video encoder used in live streaming must be able to encode the video frames on the fly. *i.e.*, the encoding time of the frame must not exceed the frame acquisition time. However, the newly Versatile Video Coding (VVC) encoder entails significant computational complexity as it contains various encoding tools designed for high-resolution content. Hence, we propose an optimization framework to tune the VVC encoder for low-resolution and low-bitrate scenarios by identifying a set of coding tools which may be disabled without harming the coding efficiency.

Chapter 5

Reducing the complexity of VVC for low bitrate applications

5.1 Introduction

The new Versatile Video Coding (VVC) standard [27] further reduces the size of the video files beyond the capabilities of the High Efficiency Video Coding (HEVC) standard [24] without compromising their quality after compression. Compared to HEVC, VVC generates half the amount of data for the same PSNR quality of the video. This feature is decisive, especially for low-latency streaming, as it can reduce the transmission delay of a video coded in the same HEVC quality.

Nevertheless, while VVC is mainly focused on high resolution content, efficient video compression solutions are also requested for streaming videos of lower resolutions (less than HD) over unstable bandwidth-limited networks. Indeed, whereas HD video is now the standard on the Internet, low-resolution contents are still used in both live and Video on demand (VOD) streaming, especially 480p and 360p.

We specifically consider use cases such as the acquisition and live streaming of low-resolution (less than HD) sport events (*e.g.*, car races and sailing races) over unstable wireless networks including LTE and satellite transmissions. For such use cases, transmission bandwidth is typically in the range of 50Kbps to 1Mbps. Moreover, low computational complexity and low power consumption solutions are highly desired for acquisition devices and embedded systems used to capture video content in sports events.

In the low latency streaming scheme introduced in Chapter 4, the acquisition device includes a camera and a computer that hosts both the encoder and the bitrate control algorithm. The acquisition device is generally onboard a vehicle or carried by a moving agent. It is usually equipped with an external battery of limited capacity. Thus, we have to minimize energy consumption. In addition, the encoder must code the frame in real-time and add a limited delay to the transmission chain. This is ensured by software or/and hardware optimization as it is present in x265. We also assume that the onboard computer is backed with sufficient calculation capacity to run the rate adaptation algorithm in real-time.

The current design of the VVC encoder is unfortunately unsuited with our use case. More precisely, a variety of encoding tools designed for HD and UHD contents are not optimal for lower resolutions and low bitrates, and they may entail a significant burden in terms of computational complexity.

In this chapter, we propose an optimization framework to tune VVC for low-resolution and low-bitrate scenarios. More specifically, we investigate the usefulness of some of the new coding tools in VVC. We experimentally show that significant complexity reduction can be achieved by disabling some of these tools while preserving coding efficiency.

To the best of our knowledge, this was the first study to investigate complexity reduction of the VVC at low resolutions and low-bitrates before the end of standardization work by the end of 2020. The experimental part of this work was performed using the VVC test model 5 (VTM5.0) as it was the latest VVC coding software available at the time of this work. VTM5.0 has been superseded by VTM10.0, which kept most of the coding tools investigated in this work. Accordingly, similar results may be observed considering VTM10.

The contributions of this chapter are the following:

- In Section (5.2), we provide a brief overview of the technical features and the coding tools of VVC Test model 5 (VTM5.0).

- In Section (5.4.2) we present a methodology used to identify the subset of coding tools that may be disabled in low-resolution and low-bitrate use cases. Our method provides a significant reduction in terms of coding complexity, while preserving compression efficiency

- In Section (5.5) we use our method to identify the best coding tools to disable in 7 video sequence and three resolutions. Then, we identify a subset of the common tools to disable in each resolution.

The experiments show that significant complexity reduction can be achieved by disabling some coding tools, while preserving coding efficiency, e.g., up to 56.06% reduction for *Johnny* sequence at 384×216 resolution with less than 1.88% increase in BD_{rate}. In addition, a set of coding tools that can be disable in each resolution for all video sequences was identified. These tools achieves 35% of complexity reduction in general with less than 2% increase in BD_{rate}.

5.2 Overview of Versatile Video Coding Test Model 5 (VTM 5.0)

The new video coding standard VVC [25] provides significant improvement in compression performance over the existing HEVC standard, with a up to 40% bitrate saving for High Definition (HD) and Ultra-High Definition (UHD) video content. The requirements for VVC include capabilities of encoding 4K and 8K sequences at up to 120 fps [28]. In what follows, the main coding tools of VTM5.0 in each module are overviewed [81].

5.2.1 Partitioning

Each frame is divided into a sequence of coding tree units (CTUs) just as in the HEVC standard, although the maximum size of the Luma CTU is up to 128×128. For each CTU, a Quad-Tree with nested Multi-Type Tree (MTT) using Binary and Ternary splitting structures is used (QTBT-TT). The CTU is first partitioned recursively using a quad-tree structure into square shapes. Then, the quad-tree leaf nodes can be further partitioned horizontally or vertically by a binary or ternary splitting structure. The final nodes are called Coding Units (CUs). They have either a square or rectangular shape and are used directly for prediction and residual coding without any further partitioning unless the CU is too large for the maximum transform length. Lastly, I-slices can have separate block tree structures for Luma and Chroma (DualTree).

5.2.2 Intra-Picture Prediction

VTM5.0 supports 65 angular intra-prediction modes, in addition to the planar and DC modes. Some conventional angular modes are replaced with wide-angle intra-prediction modes for the non-square blocks. A Multiple Reference Line (MRL) intra prediction is also proposed to use two additional lines (reference line 1 and reference line 3) in angular prediction. VTM5.0 also extends the Most Probable Modes (MPM) list to 6 candidates. For interpolating the luma samples, two sets of 4-tap filters are used. The first set of filters corresponds to the DCT-based filters applied in chroma motion compensation, while the others are reference smoothing filters. For chroma components, VVC uses only 2-tap linear interpolation.

VTM 5.0 introduces three new ways of Intra predicting a block:

1. The Cross-Component Linear Model (CCLM) prediction mode, in which the Chroma samples are predicted based on the reconstructed Luma samples of the same CU, using a linear model,

2. The Intra Sub-Partitions (ISP) where the Luma coding block is vertically or horizontally divided into 2 or 4 sub-partitions. All sub-partitions share the same intra mode, however the processing is performed gradually sub-partition by sub-partition downwards (horizontal split) or rightwards (vertical split), so each sub-partitions uses the previous reconstructed samples to generate the prediction of the current sub-partition,

3. The Matrix-based Intra Prediction (MIP) takes one line of reconstructed neighboring samples, from the left and above blocks as input vectors, and performs a matrix-vector multiplication between this vector constructed from the reference samples and a matrix selected from a set of pre-defined matrices. Finally, a linear interpolation in the vertical and horizontal directions is executed to get the predicted samples. MIP is applied only for luma blocks, but it can also be applied to chroma blocks in the case of 4:4:4 chroma sampling frames.

5.2.3 Inter-Picture Prediction

Motion prediction is performed at a sub-CU level to improve the precision. VTM 5.0 supports currently a Sub-Prediction Unit Temporal Motion Vector Prediction (SbTMVP) that includes a subblock merge mode applied to CUs with both width and height larger or equal to 8 luma samples. The MVs in this case are determined from a particular reference picture called the collocated picture. Furthermore, an AFFine motion compensation prediction (AFF) can be applied to cope with irregular motions like zoom in/out and rotation, where a sub-block is described by two or three motion vectors.

The bi-prediction mode is extended beyond simple weighted averaging, by using up to five predefined weights (Generalized Bi-prediction (GBI)). A pixel level motion refinement (Bi-Directional Optical Flow (BDOF)) may be performed on top of the merge mode or AMVP mode to improve bi-prediction at the decoder side. In order to increase the accuracy of the MVs of the merge mode, a refined operation may be performed around the initial MVs in both reference picture lists L0 and L1 using the Decoder side Motion Vector Refinement (DMVR). Motion vectors are stored at 1/16th-Luma-sample precision for Luma. In addition, the Adaptive Motion Vector Resolution (AMVR) allows the Motion Vector Difference (MVD) of the CU to be coded in one of the three resolutions: Quarter-luma-sample, Integer-luma-sample, and Four-luma-sample (or 1/16 luma-sample in AFF).

Finally, the VTM 5.0 inter coder introduces these new concepts:

1. The Triangular prediction (Triang) in which a CU may be further split into two triangular units, in either diagonal or inverse diagonal direction. Each of the two units is predicted using its own Uni-directional MV,

2. Combined Inter and Intra Prediction (CIIP) is proposed to improve the Intra mode in inter pictures, by combining the decided Intra mode with an extra merge indexed prediction,

3. Merge with MVD scheme (MMVD) is used for skip and merge modes with a new motion vector expression method with simplified signaling: The expression method includes starting point, motion magnitude, and motion direction,

4. Symmetric MVD (SMVD), which derives the MVD of reference list 1 from reference list 0, based on the assumption of linear motion in bi-prediction mode.

5.2.4 Quantization

The maximum Quantization Parameter (QP) is extended from 51 to 63, and a new concept of quantization is introduced: The Dependent Quantization (DepQuant), in which the reconstruction value for a transform coefficient depends on the value of the transform coefficient that precedes it in the reconstruction order.

5.2.5 Transform

Large block-size transforms of up to 64x64 pixels are used. High-frequency transform coefficients are zeroed out, so that only the lower-frequency coefficients (top-left 32×32 block) are retained. VTM 5.0 uses Enhanced Multiple Transform (EMT), where two new transform matrices are added in addition to DCT-II, namely the DST-VII and the DCT-VIII. Moreover, to reduce the size of the matrices of transformed coefficients, a Low-Frequency Non-Separable Transform (LFNST) is applied between transform and quantization at encoder and between de-quantization and inverse transform at decoder side. For an inter-predicted CU, the Sub-Block Transform for inter blocks (SBT) may be used instead of EMT to code only a part of the residual block with inferred adaptive transform and the other part of the residual block is zeroed out.

5.2.6 In-loop Filtering

Besides deblocking filter and Sample Adaptive Offset (SAO) used in HEVC, the Adaptive Loop Filter (ALF) is applied at the decoder side directly on the reconstructed samples of the SAO process, where one filter among 25 filters is selected for each 4×4 luma block and another filter among 8 filters for each 4×4 chroma block, based on the direction and activity of local gradients. ALF also enhances the reconstructed video at the encoder side using 7×7 diamond-shaped filters for luma and a similar 5×5 filter for chroma.

Finally, Luma Mapping with Chroma Scaling (LMCS) is performed before the in-loop filtering. This tool adjusts the input luma signal by redistributing it across the dynamic range using a piecewise linear mapping function and scales the chroma residuals according to the average value of the corresponding luma samples.

5.2.7 Entropy Coder

VVC still uses the same entropy coding method used in HEVC (Context-adaptive binary arithmetic coding (CABAC)), but with some changes: The CABAC engine uses a 2-state model with variable probability updating window sizes, instead of the pre-computed LUT of the HEVC. The transform coefficients within a Coefficient Group (CG) are coded according to pre-defined scan orders in five passes. And finally, the selected probability model and binarization models depend on the local neighborhood, where the template used to specify the local neighborhood is defined by the 5 nearby samples in the left-bottom of the current coefficient. For more information about VVC CABAC, refer to [81].

DualTree	Separate Partitioning for Luma & Chroma in I-slice
CCLM	Chroma prediction based on linear model
MRL	Multiple Reference Line intra prediction
MIP	Matrix-based Intra prediction
ISP	Intra Sub-Partitions
CIIP	Combined Inter and Intra prediction
SbTMVP	Sub-Pu Temporal Motion Vector Prediction
AFF	AFFine inter motion compensation
MMVD	Merge with MVD
SMVD	Symmetric MVD
Triang	inter predictions for Triangular Units
GBI	Generalized Bi-prediction
BDOF	Bi-Directional Optical Flow
DMVR	Decoder Side Motion Vector Refinement
AMVR	Adaptive MV Resolution
EMT	Enhanced Multiple Transform
LFNST	Low-Frequency Non-Separable Transform
SBT	Sub-Block Transform for inter blocks
LMCS	Luma Mapping with Chroma Scaling
ALF	Adaptive Loop Filter

Table 5.1: Tools of VTM 5.0 considered in this work

5.3 Main updates in VTM 10.0

Most of the coding tools reviewed in Section 5.2 have been kept for the final version of the VVC encoder, *i.e.*, VTM10. In this section, we summarize the upgrades and new tools added to VTM10.

5.3.1 Partitioning

Specific binary and ternary splits are disallowed to enable blocks of size smaller or at least equal to 64 × 64. These block regions of a CTU are called Virtual Pipeline Data Units (VPDUs) and are used in hardware video decoders for VVC to parallelize the decoding process and increase the throughput. The VPDU size must not exceed 64 x 64 luma samples because the size of the memory buffer in the pipeline stages is proportional to it.

5.3.2 Intra-Picture Prediction

In the intra-prediction module, The Position-Dependent Prediction Combination (PDPC) is included in VTM10 while it was initially removed from version 5.0 of VTM. This tool combines boundary reference samples with specific intra modes like planar, DC, and predefined angular modes. PDPC combination weights depend on the prediction mode and sample locations. In addition, minor changes have been made to the Multiple Reference Line (MRL). This tool may use one or two non-adjacent samples lines as the reference line for intra-prediction. The non-adjacent reference line can be two or three lines away from the current block. However, MRL can not be used with the planar mode and the PDPC.

5.3.3 Inter-Picture Prediction

The Motion Vector Difference (MVD) can also be coded in a half-luma-sample. in this case, an alternative luma interpolation filter is used, in operation known as switchable interpolation filter (SIF). VTM10 adds a new type of MV prediction in the merge mode and an AMVP candidate list called History-Based MV Prediction (HMVP), in addition to spatial and temporal neighbor MVs. HMVP allows VTM 10 to re-use the MVs of previously coded non-adjacent CUs to

the list. Moreover, the first two existing candidates in the merge candidate list can be combined to form a Pairwise Average MV Merge Candidate. The MVD in Merge With MVD (MMVD) of VTM 10 can only be horizontal or vertical. Triang is replaced by Geometric Partitioning Mode (GPM), in which the CU is split into two parts by a straight line parameterized by an angle and an offset. Each partition inherits one MV from the merge candidate list, and the final predicted block is generated by combining the two split blocks using a predefined weighting matrix. Finally, Prediction Refinement With Optical Flow (PROF) is used to adjust the prediction samples of 4 × 4 Luma subblocks resulting from the Affine prediction. It adds an offset derived based on the gradient around the prediction samples.

5.3.4 Quantization and Transform Coding

In addition to the transform tools of VTM 5.0, VTM 10 may use a Subblock Transform (SBT) Mode on residuals of inter-predicted CUs. SBT is applied only on a sub-partition of the CU and skips the remaining partition. This residual subpartition can have half or one-quarter of the size of the CU. For intra-predicted CUs, 1D transforms are used with the ISP mode. Joint Coding of Chroma Residuals (JCCR) is a tool that derives residual blocks of both chroma components from only one residual chroma block. It exploits the quantized chroma residual correlations to signal only one chroma component . Finally, QP values of the chroma components are derived from the QP of the corresponding luma block via look-up tables.

5.3.5 In-loop Filtering

VTM 10 applies a 3 × 4 diamond-shaped high-pass filter to luma samples for each chroma component. After performing ALF, each chroma component uses the filtered corresponding luma sample as a corrective offset. This tool is known as Cross-Component ALF (CC-ALF).

5.3.6 Screen Content Coding Tools

Screen Content Coding Tools is another set of coding tools that were not tested in this work but included in the VVC standard version. These tools

are inspired by the HEVC RExt coding tools and used in VTM 10 to increase the coding efficiency of the screen-captured content (*e.g.*, in screen sharing applications) and the computer-generated content (*e.g.*, gaming applications and animation movies). In this category of tools, we find:

1. Intra-Picture Block Copy (IBC) is an old tool from HEVC that exploits repeated block patterns inside the frame of a screen-captured or computer-generated video. It simply copies a spatially neighboring block as the prediction of another block.

2. Block-Based Differential Pulse-Code Modulation (BDPCM) which is similar to Differential PCM used in the HEVC. BDPCM applies a Differential Pulse-Code Modulation instead of transform coding on the samples resulting from horizontal or vertical intra-prediction.

3. In 4:4:4 chroma sampling, a sample may be represented by an index into a predefined palette table, and its quantized value is directly coded. This type of coding is known as Palette Mode. In addition, a switchable decorrelation to the YCgCo-R color space can be applied on CUs with 4:4:4 chroma sampling in RGB color spaces using the Adaptive Color Transform (ACT) tool.

4. Finally, Transform Skip Residual Coding (TSRC) is used to skip the transform coding of the residuals as it was proved to be more efficient in some computer-generated content.

5.4 Proposed methodology

Our aim in what follows is to identify the subset of coding tools of VTM 5.0 that may be disabled in low-resolution and low-bitrate use cases, to provide a significant reduction in terms of coding complexity, while preserving compression efficiency. This can be formulated as a constrained optimization problem, which is solved using a branch-and-prune approach. This technique identifies the individual tools and their combinations that may be safely disabled, and those that have to be kept activated.

5.4.1 Problem Formulation

Consider a set of video sequences $\mathcal{V} = \{v_1, \ldots, v_N\}$. The performance of a video encoder can be measured by the rate R (in Kbps) required to store the compressed videos, the resulting distortion D of the decoded videos (typically measured using the weighted average *PSNR* of the three components *Y*, *U*, and *V* [34]), and the complexity C of the encoding process (approximated by the run-time, measured in seconds). The values of the triple (R, D, C) depend on the input video sequence v_n and on the *encoding parameter vector* $\mathbf{p} = (p_1, \ldots, p_{n_\mathrm{p}})$ of the video coder as follows

$$(R, D, C) = f(v_n, \mathbf{p}), \tag{5.1}$$

where f is some (unknown) nonlinear function describing the behavior of the considered video coder. The components of $\mathbf{p}$ represent the coder input parameters, which may be adjusted to get different trade-offs between R, D, and C. The parameter vector $\mathbf{p}$ may be partitioned into subvectors. One may identify:

- $\mathbf{p_T}$ representing binary-valued parameters indicating whether some tools are activated or remain unused;

- $\mathbf{p_C}$ representing a finite-valued of configuration inputs for the preceding tools, *e.g.*, the *TargetBitrate* and *InitialQP* must be specified for the Rate Control, both are integer values;

- $\mathbf{p_O}$ corresponding to other parameters which do not belong to any tool, *e.g.*, *GOP size* and *GOP type* configurations.

To properly evaluate the performance of a coding tool, several target values of the rate R have to be considered, which lead to associated values of D and C.

In our work, we use the *Bjontegaard Delta Rate* ($\mathrm{BD_{rate}}$) [35] to evaluate the loss of a set $\mathcal{P}_1 = \{\mathbf{p}_1^{(1)}, \ldots, \mathbf{p}_1^{(n_\mathrm{DR})}\}$ compared to another set $\mathcal{P}_2 = \{\mathbf{p}_2^{(1)}, \ldots, \mathbf{p}_2^{(n_\mathrm{DR})}\}$ of values of the parameter vector.

The vectors $\mathbf{p}_j^{(i)} \in \mathcal{P}_j$, $j = 1, 2$ share the same components $\mathbf{p}_{\mathrm{T},j}, \mathbf{p}_{\mathrm{C},j}$, and $\mathbf{p}_{\mathrm{O},j}$, but take distinct QP values $\mathrm{QP}^{(i)}$, $i = 1, \ldots, n_\mathrm{DR}$, with $n_\mathrm{DR} \geq 4$. Sets of parameter vectors $\mathcal{P}_j$ are called *parameter configuration sets* (PCS) in what follows.

Consider some reference PCS $\overline{\mathcal{P}}$, corresponding, *e.g.*, to the best rate-distortion compromise for a set of video sequences. Our aim is to find a PCS $\underline{\mathcal{P}}$ such that

$$\underline{\mathcal{P}} = \arg\min_{\mathcal{P}} C\left(\mathcal{P}\right) \tag{5.2}$$

$$\text{such that } \mathsf{BD}_{\mathsf{rate}}(v, \overline{\mathcal{P}}, \mathcal{P}) \leqslant \Delta_{\mathsf{rate}}, \tag{5.3}$$

$$\left(R^{(i)}, D^{(i)}, C^{(i)}\right) = f\left(v, \mathbf{p}^{(i)}\right), \mathbf{p}^{(i)} \in \mathcal{P}$$

where $\Delta_{\mathsf{rate}} > 0$ is the largest tolerated loss in terms of $\mathsf{BD}_{\mathsf{rate}}$ and $C\left(\mathcal{P}\right) = \sum_{i=1}^{n_{\mathsf{DR}}} C^{(i)}$. $\underline{\mathcal{P}}$ is a PCS minimizing the complexity, while keeping good compression performance compared to the optimal parameter set.

5.4.2 Search for a good Parameter Configuration Set

In what follows, we propose a method to solve the optimization problem (5.2) in an approximate way. Our approach concentrates on finding the subvector $\mathbf{p}_{\mathsf{T},j}$ indicating the activated and disabled tools.

Consider $\overline{\mathbf{p}}_{\mathsf{T}}^{(i)} \in \overline{\mathcal{P}}$, the parameter vectors indicating the set of tools activated in the reference PCS. First, one builds all candidate PCS $\mathcal{P}_{1,j} = \{\mathbf{p}_j^{(1)}, \ldots, \mathbf{p}_j^{(n_{\mathsf{DR}})}\}$, $j = 1, \ldots, n_1$ with subvectors $\mathbf{p}_{\mathsf{T},j}^{(i)}$ obtained by disabling a *single* tool activated in $\overline{\mathbf{p}}_{\mathsf{T}}^{(i)}$, *i.e.*, $d_{\mathsf{H}}(\mathbf{p}_{\mathsf{T},j}^{(i)}, \overline{\mathbf{p}}_{\mathsf{T}}^{(i)}) = 1$, where d_{H} is the Hamming distance.

Only the candidate PCS such that (5.3) is satisfied are further considered, the others are pruned.

Second, assuming that $n_1' \leqslant n_1$ PCS satisfy (5.3). These PCS are sorted: PCS with gains in terms of $\mathsf{BD}_{\mathsf{rate}}$ are sorted first, and then PCS with a good complexity reduction and a small $\mathsf{BD}_{\mathsf{rate}}$ loss. Let $\mathbb{P}_1 = \left\{\mathcal{P}'_{1,1}, \ldots, \mathcal{P}'_{1,n_1'}\right\}$ be the ordered set of these PCS. The PCS providing a gain in terms of $\mathsf{BD}_{\mathsf{rate}}$ (negative $\mathsf{BD}_{\mathsf{rate}}$) are ordered first in $\mathbb{P}_1$ by decreasing $\mathsf{BD}_{\mathsf{rate}}$ gain. Then, the PCS providing a $\mathsf{BD}_{\mathsf{rate}}$ loss (positive $\mathsf{BD}_{\mathsf{rate}}$) are ordered by decreasing value of

$$\lambda_j = \frac{\left(C\left(\overline{\mathcal{P}}\right) - C\left(\mathcal{P}_{1,j}\right)\right)/C\left(\overline{\mathcal{P}}\right)}{\mathsf{BD}_{\mathsf{rate}}(v, \overline{\mathcal{P}}, \mathcal{P}_{1,j})},$$

where the numerator of λ_j is the relative complexity decrease provided by the PCS $\mathcal{P}_{1,j}$.

Third, the set $\mathbb{P}_1$ is split into two parts $\mathbb{P}_2$, containing the $n_2 \leqslant n_1'$ first elements of $\mathbb{P}_1$ and $\mathbb{P}_3$ containing the remaining elements. The set $\mathbb{P}_2$ contains the most promising candidates PCS with a single tool disabled compared to $\overline{\mathcal{P}}$. The greedy approach presented in Algorithm 1 is then used to combine candidate PCS, *i.e.*, disable more tools, while satisfying (5.3). In Algorithm 1, assuming that $\mathcal{P}_1 = \{\mathbf{p}_1^{(1)}, \ldots, \mathbf{p}_1^{(n_{\text{DR}})}\}$ and $\mathcal{P}_2 = \{\mathbf{p}_2^{(1)}, \ldots, \mathbf{p}_2^{(n_{\text{DR}})}\}$, the notation $\mathcal{P}_1 \wedge \mathcal{P}_2$ corresponds to the PCS $\mathcal{P}_3 = \mathcal{P}_1 \wedge \mathcal{P}_2 = \{\mathbf{p}_3^{(1)}, \ldots, \mathbf{p}_3^{(n_{\text{DR}})}\}$ such that $\mathbf{p}_{\text{T},3}^{(i)} = \mathbf{p}_{\text{T},1}^{(i)} \wedge \mathbf{p}_{\text{T},2}^{(i)}$, with $\wedge$ is the AND function.

Algorithm 1 Evaluating the best PCS

1: ***Input:*** $\mathbb{P}_2$
2: ***Output:*** $\mathcal{P}_1$
3: ***Initialization:*** extract $\mathcal{P}_1$, the first element of $\mathbb{P}_2$
4: **while** $\mathbb{P}_2 \neq \emptyset$ **do**
5: Extract $\mathcal{P}_2$, the next element of $\mathbb{P}_2$
6: ***If*** $C(\mathcal{P}_1 \wedge \mathcal{P}_2) \leq C(\mathcal{P}_1)$ and $BD_{rate}(v, \overline{\mathcal{P}}, \mathcal{P}_1 \wedge \mathcal{P}_2) \leq \Delta_{rate}$
7: $\mathcal{P}_1 = \mathcal{P}_1 \wedge \mathcal{P}_2$
8: **end while**

Algorithm 1 progressively disable tools corresponding to the PCS in $\mathbb{P}_2$, starting with the most promising PCS. When disabling a tool results in a complexity reduction while satisfying (5.3), the PCS is updated. Tools, when disabled, do not reduce the complexity or lead to a large loss in BD_{rate} are kept activated.

Finally, a branch-and-prune approach presented in Algorithm 2 is considered, starting from the PCS $\mathcal{P}_1$ provided by Algorithm 1 to select additional tools to disable corresponding to PCS in $\mathbb{P}_3$. One tries first to disable a single additional tool from $\mathcal{P}_1$ corresponding to the various PCS in $\mathbb{P}_3$. All PCS $\mathcal{P} \in \mathbb{P}_3$ such that $\mathcal{P}_1 \wedge \mathcal{P}$ leading to a performance decrease compared to $\mathcal{P}_1$ are discarded from $\mathbb{P}_3$. Then pairs, triples, *etc.* of PCS remaining in $\mathbb{P}_3$ are considered.

Let $\mathcal{P}_2$ the PCS (or combination of PCS) in $\mathbb{P}_3$ leading to the smallest value of $C(\mathcal{P}_1 \wedge \mathcal{P}_2)$ while $\text{BD}_{\text{rate}}(v, \overline{\mathcal{P}}, \mathcal{P}_1 \wedge \mathcal{P}_2) \leqslant \Delta_{\text{rate}}$. Then the PCS $\mathcal{P} = \mathcal{P}_1 \wedge \mathcal{P}_2$ is an approximate solution of (5.2).

Algorithm 2 Branch-and-prune method

Input: $\mathcal{P}_1$ and $\mathbb{P}_3$
Output: $\mathcal{P}$
Initialization: $i = 1$
while number of PCSs in $\mathbb{P}_3 > 1$ **do**
    Build $\mathbb{P}$, all combinations of i tools from $\mathbb{P}_3$
    while $\mathbb{P} \neq \emptyset$ **do**
        Extract $\mathcal{P}_2$, first PCSs in $\mathbb{P}$
        If $C(\mathcal{P}_1 \wedge \mathcal{P}_2) \geq C(\mathcal{P}_1)$ and $BD_{rate}(v, \overline{\mathcal{P}}, \mathcal{P}_1 \wedge \mathcal{P}_2) \geq \Delta_{rate}$
        Discard $\mathcal{P}_2$ from $\mathbb{P}$
        EndIf
    end while
    Put all PCSs of $\mathbb{P}$ in $\mathbb{P}_3$
    $i = i + 1$
end while
Extract $\mathcal{P}_2$, the only PCS in $\mathbb{P}_3$
$\mathcal{P} = \mathcal{P}_1 \wedge \mathcal{P}_2$

5.5 Performance evaluation

5.5.1 Experimental setup

We selected 14 JVET test sequences defined in the Common Test Conditions (CTC) [82], each of the sequences has at most 300 images. In a first phase, 7 sequences were considered to apply our approach and identify the best PCSs. Then, in a second phase, tests are conducted on all sequences to evaluate the performance obtained with the previously identified best PCSs.

All sequences have been temporally sub-sampled at 30 fps and spatially sub-sampled using FFmpeg [83] resulting in frames of 384×216, 512×288, and 640×360 pixels. A Random Access (RA) configuration is selected according to JVET CTC [82] and QP values are chosen in $\{27, 32, 37, 42\}$. VTM 5.0 is used in the experiments and run on a PC with 2 Intel Xeon CPU E5-2670 v3 24 cores @ 2.30 GHz running under Linux. The threshold Δ_{rate} is fixed to 2% as we have noticed that this loss is subjectively unnoticeable. The value of n_2 helps to get a trade-off between complexity and accuracy in the search for $\underline{\mathcal{P}}$. Here, we take $n_2 = n_1'/2$. The tools of VTM 5.0 considered in this work are listed in Table 5.1.

Disabled Tools	BD_{rate} %	ΔC %	λ	Disabled Tools	BD_{rate} %	ΔC %	λ	Disabled Tools	BD_{rate} %	ΔC %	λ
LMCS	-0.42	10.77	-	LMCS	-0.40	13.28	-	BDOF	-0.29	13.20	-
SMVD	-0.01	8.30	-	SMVD	0.02	8.98	422.56	SMVD	-0.08	9.67	-
AMVR	-0.01	8.46	-	MMVD	0.15	15.56	104.76	LMCS	-0.07	9.85	-
GBI	0.01	8.74	1544.07	BDOF	0.14	10.83	77.86	MMVD	0.10	15.67	161.51
CIIP	0.03	7.98	271.58	ISP	0.37	20.55	55.37	CIIP	0.06	7.65	121.65
MMVD	0.20	14.21	72.12	CIIP	0.14	7.46	52.16	AMVR	0.15	10.06	66.67
AFF	0.29	16.59	57.98	AFF	0.58	19.87	34.48	AFF	0.58	18.72	32.06
MIP	0.14	8.30	57.41	AMVR	0.26	8.60	33.43	Triang	0.41	11.39	28.00
Triang	0.28	10.04	36.47	Triang	0.39	11.69	29.77	MIP	0.45	9.30	20.75
MRL	0.20	7.23	36.01	SBT	0.31	6.72	21.92	SbTMVP	0.28	4.54	16.50
SbTMVP	0.14	4.81	34.35	MIP	0.45	7.81	17.32	SBT	0.24	3.80	15.60
SBT	0.07	1.64	24.45	MRL	0.39	6.38	16.27	EMT	0.42	6.05	14.49
EMT	0.34	7.01	20.77	SbTMVP	0.32	5.08	16.04	GBI	0.69	9.71	14.04
ALF	1.60	28.15	17.55	GBI	0.78	9.72	12.45	ISP	0.49	5.88	11.94
LFNST	0.87	12.30	14.12	LFNST	1.07	12.89	12.03	LFNST	1.36	12.69	9.33
BDOF	0.76	9.56	12.65	EMT	0.51	5.89	11.45	ALF	2.11	18.04	8.56
ISP	0.55	6.35	11.59	ALF	1.98	20.55	10.38	MRL	0.74	5.81	7.87
DMVR	0.59	5.30	8.95	CCLM	0.82	6.28	7.70	CCLM	0.88	5.46	6.18
CCLM	0.79	5.94	7.50	DMVR	0.91	4.35	4.76	DMVR	0.78	4.48	5.72
DualTree	0.29	-2.29	-7.94	DualTree	0.32	-1.71	-5.37	DualTree	0.70	-3.00	-4.32

Table 5.2: BD_{rate} and complexity reduction ΔC in *Johnny* test sequence when disabling one tool at time

5.5.2 Analysis

In this section, we present experimental results in order to illustrate the proposed approach. Table 5.2 presents the detailed BD_{rate} and complexity reduction ΔC when disabling one tool at a time for the *Johnny* sequence of resolutions 384×216, 512×288 and 640×360. These percentages are calculated relative to VTM 5.0 with all tools activated (a negative BD_{rate} indicates a gain with respect to VTM 5.0).

From Table 5.2, one observes that disabling tools related to interframe coding (*e.g.*, AFF, MMVD, and Triang in resolutions 384×216) and inloop filtering (ALF, LMCS) leads to significant gains in complexity. Other tools related to transform operations such as LFNST and EMT also lead to a significant complexity decrease. Disabling a tool can sometimes lead to an improved BD_{rate} when operating at low resolutions and low bitrates, such as LMCS, SMVD and AMVR.

Table 5.3 shows the BD_{rate} and complexity reduction ΔC for the best combination of disabled tools obtained applying the method described in Section 5.4.2 for seven test sequences and three resolutions. When several tools are disabled, the complexity gains accumulate in most of the cases. Never-

Resolution	Video	R Kbps	BD_{rate} %	ΔC %	Disabled Tools
	Johnny	18-72	1.88	56.06	**LMCS GBI LFNST MMVD MIP CIIP** SMVD MRL SBT Triang AFF AMVR SbTMVP
	Basketball	48-368	1.97	37.07	**LMCS GBI LFNST EMT MIP CIIP** SMVD
384×216	DaylightRoad2	48-367	2.00	48.91	**LMCS GBI LFNST MMVD EMT MIP CIIP** MRL SBT Triang
	BQMall	50-331	1.98	44.93	**LMCS GBI MMVD EMT MIP CIIP** SMVD AFF CCLM
	Drums	89-538	1.74	43.27	**LMCS GBI LFNST EMT MIP CIIP** SMVD MRL AFF SbTMVP
	RaceHorses	50-403	2.00	38.82	**LMCS GBI LFNST MMVD EMT** MRL SBT
	Kimono	27-271	1.91	51.21	**LMCS GBI LFNST MMVD EMT** SMVD MRL AFF ISP
Average	-	-	1.93	45.75	-
	Johnny	23-102	1.97	57.01	**LMCS CIIP MMVD** AFF Triang SBT ISP BIO AMVR
	Basketball	75-576	1.81	35.25	**LMCS CIIP EMT LFNST** SMVD MRL MIP SBT
	DaylightRoad2	75-575	1.88	48.37	**LMCS CIIP EMT LFNST** MMVD MRL MIP GBI Triang SBT
512×288	BQMall	71-477	1.82	34.92	**LMCS CIIP EMT MMVD** SMVD MRL GBI CCLM
	Drums	125-661	2.00	35.25	**CIIP EMT LFNST SMVD** AFF CCLM SbTMVP
	RaceHorses	78-763	1.79	34.34	**LMCS CIIP EMT LFNST SMVD** AFF CCLM SbTMVP
	Kimono	41-427	1.84	47.78	**LMCS CIIP EMT LFNST MMVD SMVD** MRL ISP AFF IMV
Average	-	-	1.87	41.85	-
	Johnny	30-148	1.92	55.24	**LMCS MIP SMVD CIIP MMVD** AFF Triang AMVR BIO
	Basketball	99-700	1.89	32.45	**LMCS MIP SMVD EMT** LFNST SbTMVP
	DaylightRoad2	100-702	1.89	41.68	**LMCS MIP SMVD MRL GBI EMT MMVD** LFNST SBT
640×360	BQMall	95-639	2.00	36.99	**LMCS MIP SMVD MRL GBI EMT MMVD** CCLM
	Drums	164-789	1.83	41.26	**LMCS MIP SMVD MRL GBI CIIP** AFF SbTMVP
	RaceHorses	109-631	1.30	40.17	**LMCS MIP SMVD MRL GBI EMT CIIP MMVD** SBT ISP
	Kimono	54-531	1.80	45.77	**LMCS SMVD MRL GBI CIIP** LFNST AFF Triang
Average	-	-	1.80	41.94	

Table 5.3: BD_{rate} and complexity reduction ΔC of best PCS for tested resolutions and sequences; Selected common tools are in bold

	384×216		512×288		640×360	
Common Tools	LMCS GBI LFNST MMVD EMT MIP CIIP		LMCS CIIP MMVD SMVD EMT LFNST		LMCS MIP CIIP EMT SMVD MRL GBI MMVD	
Video	BD_{rate}%	ΔC%	BD_{rate}%	ΔC%	BD_{rate}%	ΔC%
Johnny	**2.34**	38.76	**2.13**	39.13	0.99	42.01
Basketball	2.00	37.95	1.83	24.57	1.57	26.03
DaylightRoad2	0.68	38.96	0.94	35.26	0.98	35.44
BQMall	1.24	37.81	1.59	35.73	1.33	35.31
Drums	1.87	37.94	1.63	34.12	1.21	35.12
RaceHorses	1.87	35.65	1.56	33.96	0.33	26.45
Kimono	0.94	41.60	1.83	38.24	**2.39**	35.47
ParkScene	0.90	42.25	0.77	41.26	0.85	42.71
KristenAndSara	1.99	38.62	1.87	38.93	0.96	39.32
CatRobot	1.42	42.58	1.41	37.39	1.27	23.79
Tango	1.20	38.55	1.58	36.06	0.95	35.18
ToddlerFountain	1.20	38.55	1.58	36.06	1.43	39.89
SlideShow	1.52	27.20	1.44	23.61	**2.88**	23.03
SlideEditing	1.09	41.68	0.63	39.72	1.64	40.25
Average	1.45	38.44	1.49	35.29	1.34	34.29
SlideEditing	1.09	41.68	0.63	39.72	1.64	40.25
Average	1.45	38.44	1.49	35.29	1.34	34.29

Table 5.4: BD_{rate} and complexity reduction ΔC when disabling the common combinations of tools. Cells in bold do not satisfy (5.3).

theless, this observation does not hold for BD_{rate}, due to the complex dependency among tools and their influence on the coding efficiency. Considering *Johnny* at the resolution of 384×216, jointly disabling the tools: {LMCS GBI LFNST MMVD MIP CIIP SMVD MRL SBT Triang AFF AMVR SbTMVP}, leads to a complexity reduction of 56.06%, with a BD_{rate} loss of 1.88%. For resolutions of 512×288 and 640×360, a complexity reduction of 57.01% and 55.24% is achieved with a BD_{rate} loss of 1.97% and 1.92% respectively. Similar results are obtained when applying the same methodology on other sequences such as, *BasketballDrive*, *DaylightRoad2*, *BQMall*, *Drums*, *RaceHorses,* and *Kimono*. The weakest reduction in complexity is observed with he *BasketballDrive* video sequence characterized by a large temporal variance, 37.07% is recorded for resolutions of 384×216 and 32.45% for resolutions 640×360.

These results are obtained by merely disabling tools and without algorithmic optimization. Nevertheless, the combination of tools that provide the optimal complexity reduction is different from one sequence to the other and even from one resolution to the other. The amount of complexity reduction is also varying. This is due to the spatial and temporal properties of sequences.

Yet, for each resolution, using the results of Table 5.3, it is possible to identify one common combination of tools satisfying constraint (5.3) for all sequences. Accordingly, these PCSs are: {LMCS GBI LFNST MMVD EMT MIP CIIP} for resolution 384×216, {LMCS CIIP MMVD SMVD EMT LFNST} for 512×288, and {LMCS MIP CIIP EMT SMVD MRL GBI MMVD} for 640×360. Table 5.4 shows the BD_{rate} and complexity reduction for the previously identified PCSs considering the 14 test sequences. We observe that the constraint (5.3) is satisfied in most cases. Thus, putting these PCSs in separate profiles for each resolution will be beneficial for use cases with real-time and low-bitrate constraints. We conclude that the PCS identification approach presented in this paper provides results that are likely to be generalized to a larger set of video sequences.

5.6 Conclusions

In this chapter, we present an optimization method of VVC encoder targeting low-resolution video sequences encoded at low bitrates (less than 1 Mbps). Our aim is to identify a set of coding tools which may be disabled while preserving coding efficiency. For that purpose, a branch-and-prune approach is proposed to determine the set of coding tools which provide the best complexity reduction, while satisfying a constraint on the BD_{rate} degradation.

Experimental results show that significant reduction of encoding complexity can be achieved, with negligible BD_{rate} loss. For instance, a complexity reduction of 56% was achieved for *Johnny* at a 384×216 resolution by applying our method, with a loss of 1.88% in BD_{rate}. Moreover, due to the spatial and temporal properties of sequences, The best set of coding tools to disable is different across the video sequence and even the tested resolutions.

Nevertheless, we were able to propose a common combination of tools to disable for each resolution. Our experimental results show that these common tools most likely cause less than 2% BD_{rate} loss with up to 35% reduction in terms of encoding complexity. This result is particularly beneficial as we can build coding profiles for each resolution removing the non-useful tools. Therefore, automatically disabling them and reducing the processing complexity for real-time encoding.

We also observe that disabling tools related to interframe coding (*e.g.*, AFF, MMVD) and transform operations (*e.g.*, LFNST, EMT) always leads to a significant complexity decrease without causing a massive drop in coding efficiency. In addition, disabling some tools, such as LMCS, SMVD, may sometimes lead to an improved BD_{rate} when operating at low resolutions and low bitrates which further illustrates that some coding tools are not optimal of high resolutions videos.

Finally, our results are promising but represent the first step toward a real-time VVC encoder. Algorithmic optimization can be applied to partitioning, intra prediction, or transform module to reduce the computational time. Furthermore, hardware optimization is necessary to minimize memory access, favor parallelism and execute repetitive tasks. All of these optimization possibilities are an essential direction for future work.

The Fraunhofer Heinrich Hertz Institute (HHI) released a fast and efficient VVC encoder software known as VVenC [84]. The VVenC software is based on VTM, with optimizations including some algorithmic optimizations, extensive SIMD optimizations, and multi-threading support to exploit parallelization. Additionally, VVenC uses a similar approach to ours by supporting five predefined presets. In each preset, a subset of coding tools is disabled to achieve a given tradeoff between encoder complexity and video quality. In the slowest preset, the encoder reaches the highest compression gain with the most considerable runtime, while in the fastest preset, the runtime is less significant, but VVenC is only 10% more efficient than HEVC. More details about VVenC is available in [85][84].

Chapter 6

Conclusion

6.1 Summary

In this thesis, we address the problem of video bitrate adaptation for low-latency video streaming. Chapter 1 stated our objective: the design of a bitrate adaptation algorithm for low-latency video streaming from a mobile transmitter with 100 to 200 ms glass-to-glass delay targets. In low-latency live streaming, the transmitter (server) has to send the compressed video in an uplink direction through a wireless access and a wired core network to the client. The high variability of the wireless channel characteristics due to the mobility of the transmitter and to the variable number of users sharing the channel combined with the low latency constraint makes this uplink transmission scenario more challenging than classical downlink HTTP adaptive streaming.

Chapter 2 provides some background on the basics of video coding and HTTP adaptive streaming. We also discuss the main end-to-end delay components in video delivery systems. We provide a literature review of the state-of-the-art bitrate adaptation algorithms and their classifications according to the location of the bitrate adaptation logic (server-driven or client-driven) and the input used for the adaptation (bandwidth-based, buffer-based, or hybrid approaches). We stated that in the context of low-latency streaming from a mobile transmitter. Transmitter-driven approaches are better suited for selecting the video bitrate according to the network state and transmitter buffer level. This is because the moving transmitter has a better view of the variations of the channel and network state, and does not have to wait for delayed reports of the network and buffer states provided by the client. In

addition, an adaptation at the frame level is necessary to achieve low delay transmission. The adaptation at the frame level is performed using a model of the relation between the size of the bitstream resulting from the encoding of a video frame and the selected quantization parameter. Accordingly, we recall several state-of-the-art the parametric rate models used for bitrate control. Finally, some technical features of the new Versatile Video Coding (VVC) standard and the state-of-the-art encoder optimization methods has been reviewed.

We present our first contribution in Chapter 3: a novel inter-dependent Rate-QP model, *i.e.*, R-(QP, D). Our model describes the relationship between the bitstream size R_n of frame n, its quantization parameter QP_n, and the MSE distortion D_{n-1} of the reference frame $n-1$. The R-(QP, D) is beneficial when adjusting the QP of the frame according to some target bitrate budget of a frame in case of low latency live streaming. This target bitrate budget is determined via some bitrate adaptation algorithm. Our proposed model outperforms other Rate-QP models when encoding is performed with constant or variable QP. In addition, we have proposed an procedure to estimate iteratively the parameters of the R-(QP, D) model. This procedure allows one to estimate these parameters with a limited number of encoding trials for each frame which is useful in low latency streaming. Part of the material in Chapter 3 has been presented in

- Mourad Aklouf, Marc Leny, Michel Kieffer, and Frédéric Dufaux. "Interframe-Dependent Rate-QP-Distortion Model for Video Coding and Transmission." In 2021 IEEE International Conference on Image Processing (ICIP), pp. 2019-2023. IEEE, 2021.

Chapter 4 proposes a new model predictive bitrate adaptation algorithm for low latency video streaming from a mobile terminal. The proposed approach exploits the transmission buffer level and an estimate of the wireless transmission rate to determine the target encoding bitrate of each frame. The choice of the quantization parameter for each frame is performed via the R-(QP, D) model proposed in Chapter 3. We compare the performance of the proposed approach with four reference rate adaptation algorithms, namely Festive [10], Panda [11], BOLA [12], and BBA [13], considering streaming scenarios with a glass-to-glass latency of less than 200 ms. Some of these algorithms

have been adapted to the transmitter-driven framework. Simulation results involving real 4G bandwidth traces showed that our proposed MPC approach outperforms the algorithms from the literature in both average PSNR and number of lost frames.

Finally, live streaming is delay-sensitive, and it requires an encoder that is capable of compressing the video stream in real-time. Therefore, we present in Chapter 5 an optimization method for VVC encoder targeting low-resolution video sequences encoded at low bitrates. We propose a branch-and-prune approach to identify in a systematic way a set of coding tools which may be disabled while satisfying some constraint on the BD_{rate} degradation, thus preserving coding efficiency. A complexity reduction of up to 56% was achieved for video sequences of resolution 384×216, 512×288, and 640×360 by applying our method, with a loss of less than 2% in BD_{rate}. Moreover, we were able to identify a common set of coding tools to disable in each resolution. These common tools can be used to create coding profiles for each resolution thus reducing the coding complexity while encoding on the fly. The material in Chapter 5 has been presented in

- Mourad Aklouf, Marc Leny, Frederic Dufaux, and Michel Kieffer. "Low complexity versatile video coding (VVC) for low bitrate applications." In 2019 8th European Workshop on Visual Information Processing (EUVIP), pp. 22-27. IEEE, 2019.

6.2 Future work and perspectives

The work presented in this document can be extended in the following directions.

6.2.1 Improvement for the R-(QP, D) model

The R-(QP, D) model is proposed assuming that encoding is performed with a low-latency configuration. The frame n is coded as a P-frame and uses only the previous coded frame $n-1$ as a reference frame. Nevertheless, the x265 encoder enables another type of low delay encoding configuration with better coding efficiency, *i.e.*, smaller bitstream size, with a cost of a slight increase in coding time. The frame n may use two coded frames as references, *e.g.*,

frame $n-1$ and $n-3$. In this case, the coded blocks in frame n use two motion vectors, and the samples are reconstructed based on a weighted combination of the blocks from the reference frames. An improvement would be to investigate the possibility of using a combination of the distorsions of reference frames in the R-(QP, D) model. *e.g.*, $D = \alpha D_1 + \beta D_2$, with D_1 and D_2 being the MSE distortions of the reference frames 1 and 2, α and β are weights of the two reference frames 1 and 2, respectively. Authors in [86, 87] addressed a similar problem for bitrate control in the HEVC encoder with the Random Access configuration.The difficulty comes from the evaluation of α and β prior to encoding. One has to evaluate whether this proportion is stable or not.

Intra-refresh can be enabled to reduce the peaks of video bitrate caused by I-frames. Several images may contain a column of CTUs coded in intra-prediction mode. This increases the size of the bitstream associated to each frame. Our proposed model remains valid for this type of frames. Nevertheless, the iterative estimation of the model parameters is not tested with intra-refresh enabled, and more experimental verification has to be performed.

When a change of scene occurs, the parameters of the R-(QP, D) model may change significantly. The recursive estimation of the model parameters may have to be reset, which may require the first frame to be coded with a large number of trials, and the model parameters to be determined using the Least Squares Estimator presented in Section 3.4.1. Nevertheless, we have noticed that the reset is not necessary for some tests where the scene change does not change the properties of the video, *e.g.*, a shift in camera for shooting in a new angle but in the same environment. A potential solution could be to calculate the spatial and temporal properties of the frame before and after the scene change to determine whether the reset is necessary.

In the iterative parameter estimation process, an initial vector QP_0 of quantization parameters has been considered to initialize the estimate of the R-(QP, D) model for the first frame. Several encoders are then run in parallel with variations ΔQP of the quantization parameters. These variations were adjusted experimentally so as to get the best model accuracy in our experiments. These two vectors must be optimized in the future according to the video properties and the available transmission rate. QP_0 must be chosen so that the bitrate of the video is in the range of the transmission rate. ΔQP

should provide sufficient bitrate diversity to enable a satisfying prediction accuracy, even in case of significant variation of the quantization parameter.

6.2.2 Transmission rate estimator

An important research direction is how to efficiently estimate the channel and network bandwidth. We did not discuss the type of estimator to use in our work. The transmission rate C_n at time step n when coding frame n is taken from the set of bandwidth traces described in [9]. Estimating the bandwidth can be done in two ways.

The first option is to probe the channel multiple times then smooth the transmission rate in a given temporal window to denoise the measures. The available transmission rate at instant n can be determined by measuring the quantity of transmitted data in a time interval or using tracing tools such as QXDM [79] or MobileInsight [80]. These tools are used to capture 4G/5G control messages between the terminal and the base station and understand the behavior of the resource blocks allocation to the users in downlink and uplink direction. Accordingly, the throughput can be estimated using this traced information in a frequency much higher than tools working at application layer, such as G-NetTrack Pro, see [88].

Another option is to build a bandwidth map and estimate the transmission rate in a given geolocated position of the mobile transmitter. In this approach, a database of transmission rates with GPS positions is first built offline [14]. Then, the information collected is used to estimate the future network conditions of the mobile transmitter, and the video bitrate is adjusted accordingly. The transmitter position can be calculated using Kalman-based prediction.

6.2.3 N-steps MPC Algorithm

If the transmission rate can be estimated N steps in the future, It would be possible to perform N-step bitrate control using the MPC algorithm. The target bitrate of the N future frames may be determined at once. N-step MPC allows more efficient bitrate adaptation and client-buffer control: When the controller anticipates a future drop in the transmission rate, for instance, at time $n+3$, it can encode some frames before the drop event (*e.g.*, frame $n+1$ and $n+2$) at a lower bitrate, which allows transmitting the frames in a shorter

time and therefore provide an increased playback margin for frame $n+3$. This could slightly reduce the PSNR quality of the coded frames but avoid freezing events and reduce the latency. In addition, N-step MPC makes it possible to limit the oscillations by smoothing the target bitrates of the frames in a window of size N.

An other direction is a dynamic adjustment of the margin τ^* and the end-to-end playback delay Δ_p by small variations of the frame rate at receiver [89] based on estimated future transmission rate to minimize playout interruptions.

6.2.4 Ensuring fairness

The techniques described in this document enable low-latency live streaming from one mobile acquisition device to one client. The client could be a processing unit that may realize additional treatments on the encoded video or broadcast it to the final consumers. Sometimes, we want to transmit multiple video streams on the same channel, for example in the case of videos acquired by different closely-located cars sharing the same 4G/5G base station. The transmitters are then in competition for the wireless resource. When the wireless channel is saturated, the rate adaptation techniques of each encoder does not allow a fair share of the channel capacity and QoE fairness cannot be guaranteed. One way to solve this problem is to use an in-network coordination proxy in charge of facilitating fair resource sharing among transmitters [90].

With HTTP adaptive streaming in downlink direction, the root of the lack of fairness between clients sharing some wireless channel and of the oscillation of the rate and PSNR among clients is their ON-OFF activity pattern. A client typically operates in two states: the buffering state, in which the client requests a new video segment when the previous segment is fully downloaded, and the steady state, in which the client requests one new segment periodically every Δ seconds, with Δ is the duration of the segment. This creates an activity pattern where the client is either ON when downloading a segment or OFF. During the ON period, the client usually measures the instantaneous downloading rate. The client may overestimate the available bandwidth due to the temporal overlap between the players ON-OFF periods. The players

could also switch back to a lower bitrate when the segment is not successfully downloaded, causing oscillations. Authors in [91] propose a server-based traffic shaping method to avoid the OFF periods during the steady-state. The shaping limits the encoding rate of the segment, so the download duration will be roughly equal to the segment duration Δ . Khan *et al.* [92] clarify the fairness problem and evaluate some bitrate adaption algorithms when competing in bottleneck links.

The fairness problem in our case is quite different from the fairness problem in HTTP streaming because of the uplink streaming and the granularity of adaptation. In our case, the transmitter is in ON period most of the time, so the used bandwidth is less likely to be overestimated. Nevertheless, a proper way to share channel capacity to achieve fair QoE between the served clients must be investigated.

6.2.5 Q-learning bitrate adaptation approach

The problem of bitrate adaptation can be addressed using reinforcement learning (RL) approaches. This technique allows the transmitter to dynamically learn the best actions, *i.e.*, encoding bitrate corresponding to the actual network environment. The bitrate adaptation is performed by an RL agent that interacts with its environment through actions (*i.e.*, the set of possible encoding bitrates) and evaluates them based on some assigned numerical reward. The reward, in this case, can be the PSNR of the received video frames, or a function of the playback marging at the client. The agent goal is to maximize the user QoE and learn the optimal action to take (encoding bitrate) for each state of the environment. The state of the environment is defined by the transmission buffer level and the available transmission rate. When the agent has limited knowledge about the environment dynamics, a commonly used RL algorithm is Q-Learning [93]. In Q-learning, a table of Q values is used to measure the quality of taking specific actions in a particular state, based on the perceived cumulative rewards. These Q-values are updated every time an action is taken. The optimal encoding bitrate is then determined from this table. Another approach is to use a neural network to approximate the Q-value function, *i.e.*, deep Q-learning. The state is given as the continuous input and the Q value is generated as the output for all possible actions.

Several methods based on Q-learning have been proposed [59, 58, 94]. All of them are implemented in client-driven HTTP adaptive streaming architectures and adapt the bitrate at the segment level. In our situation, the target bitrate of the frames could be determined using similar methods, then the optimal QPs to encode the frames are determined with the proposed R-(QP, D) model.

Another approach of using Q-learning is to consider the frames QPs as the output instead of the target bitrates. This allows to remove the R-(QP, D) model of the streaming architecture and integrate it into the learning process. However, the temporal dependency between the frames can decrease the accuracy of the QP prediction. Hence, an additional state that represents the distortion of the reference frame (or its QP) must be added. More experiments are required to determine which approach is more efficient.

6.2.6 Bitrate adaptation in the context of E-sports

Major tech companies including Microsoft, Google, and Amazon compete with video gaming industries such as Xbox and Ubisoft in new game streaming services known as cloud gaming. Cloud gaming is a relatively new concept in which users without powerful gaming devices get access to a large library of online games at the cost of a monthly subscription. Cloud gaming operates by hosting and running the games on powerful servers. The user (client) sends the game input to the server that processes it then streams back the game environment in a compressed video sequence through the internet. The video must be streamed at very low latency. Cloud gaming allows users to enjoy high-level video games on regular portable devices such as smartphones and tablets.

E-sports such as "Fortnite World Cup" and "League of Legends" are increasing in popularity. Individual players or teams can now compete against each other in organized video gaming tournaments for a cash prize. The gaming culture is not just about playing but also about watching other people play. A massive part of the gaming audience is heading towards large streaming platforms such as Youtube and Twitch to watch international gaming championships or video game commentators such as PewDiePie and AboFlah. Accordingly, low latency adaptive video streaming will have a central role in en-

abling these services.

Our MPC-based rate adaptation method for low latency video delivery can be used for such use cases. Nevertheless, some changes must be made. First, the transmitted videos are computer-generated, with different video characteristics from the videos tested in Chapter 4. Hence, we need to check the efficiency of the R-(QP, D) model for this type of video then make changes in the model if necessary. In addition, a transmitter-driven scheme is not necessarily appropriate. The transmitter (server) is located in a fixed position at the edge of the network, while the users could be moving when using portable devices such as smartphones. Hence, the bitrate adaptation should be performed according to the client (reception) buffer level and the available download rate.

6.3 Industrial perspectives

The choice of a reliable live streaming product is an essential aspect of the success of a cultural, political or sports event. Live streaming has become the primary method to reach a new audience and maintain relationships with old customers. It is thus crucial to satisfy the user QoE.

The work presented in this thesis offers an interesting industrial perspective for low latency event streaming. The bitrate adaptation technique described in this thesis can be implemented on products such as:

- Nomad cars [95], an application used to capture video on-board a racing car, transmit the compressed video via 3G/4G network to a remote coach, teams in the pitlane, partner TV channels, or post it on social networks.

- Nomad sails [96] in which the capture video on-board of a boat in the middle of the ocean is uploaded via satellite link and made available anywhere in the world with the lowest possible latency, mostly for TV interviews.

In addition, exclusive broadcasting of cultural events such as music festivals or openings of exhibitions will also be possible in very low latency. A

virtual event platform can be created to provide a new interactive digital environment where people can connect, discover and exchange information.

Lately, the work of this thesis will enable a new type the remote control application: piloting a railway train remotely. This application is an essential part of the autonomous train project, in which the drivers located kilometers away in an operating center may have to manually operate a train stopped by hazardous situation (malfunction, accident, object on rails…). This project is carried out by EKTACOM with several partners, including the french company Société Nationale des Chemins de Fer (SNCF).

A prototype of a train remote control system already exists. The train is equipped with a camera to film the train track and a transmitter that transmits the frames to a remote control station. The driver visualizes the journey of the train from his position in the remote control center and pilots the train according to the received video sequence (*i.e.*, acceleration, deceleration, stopping, railway signals reading). However, the captured video is usually transmitted via a very limited or highly variable channel. Field tests underlined bandwidth variation from 20kb/s to 3 Mb/s, with variations in packets loss and error rate. Therefore, the quality of the received frames is sometimes insufficient for the driver to make safe decisions (mostly because of the difficulty in identifying the lights and their color). Our algorithm can be implemented in this case to fully exploit the channel capacity and maximize the driver QoE.

Bibliography

[1] Cisco visual networking index: Forecast and trends 2017-2022. *White Paper*, 2019, Available: https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf.

[2] Abdelhak Bentaleb, Bayan Taani, Ali C Begen, Christian Timmerer, and Roger Zimmermann. A survey on bitrate adaptation schemes for streaming media over HTTP. *IEEE Communications Surveys & Tutorials*, 21(1):562–585, 2018.

[3] Ícaro Siqueira, Guilherme Correa, and Mateus Grellert. Rate-distortion and complexity comparison of hevc and vvc video encoders. In *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, pages 1–4. IEEE, 2020.

[4] Hyomin Choi, Jonghun Yoo, Junghak Nam, Donggyu Sim, and Ivan V Bajić. Pixel-wise unified rate-quantization model for multi-level rate control. *IEEE Journal of Selected Topics in Signal Processing*, 7(6):1112–1123, 2013.

[5] Yuan Li, Huizhu Jia, Pan Ma, Chuang Zhu, Xiaodong Xie, and Wen Gao. Inter-dependent rate-distortion modeling for video coding and its application to rate control. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2014.

[6] Xiaokang Yang, Yongmin Tan, and Nam Ling. Rate control for H.264 with two-step quantization parameter determination but single-pass encoding. *EURASIP Journal on Advances in Signal Processing*, 2006(1):063409, 2006.

[7] MulticoreWare. x265 software documentation. https://x265.readthedocs.io/en/master/, 2020.

[8] Joint Collaborative Team on Video Coding (JCT-VC). High Efficiency Video Coding (HEVC). https://hevc.hhi.fraunhofer.de/, 2021.

[9] Darijo Raca, Jason J Quinlan, Ahmed H Zahran, and Cormac J Sreenan. Beyond throughput: a 4G LTE dataset with channel and context metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference*, pages 460–465, 2018.

[10] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108, 2012.

[11] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C Begen, and David Oran. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, 2014.

[12] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. *IEEE/ACM Transactions on Networking*, 28(4):1698–1711, 2020.

[13] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[14] Jia Hao, Roger Zimmermann, and Haiyang Ma. Gtube: Geo-predictive video streaming over HTTP in mobile environments. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 259–270, 2014.

[15] Sandvine. The global internet phenomena report. 2022.

[16] whats-on netflix. List of 4K TV Series & Movies on Netflix. https://www.whats-on-netflix.com/library/4k-titles-on-netflix/, 2021.

[17] Injoon Cho, Sangjin Hahm, Sansung Kim, Byungsun Kim, Sanghoon Kim, and Sungho Jeon. Terrestrial 4k uhd live broadcasting of sports. *Journal of Broadcast Engineering*, 20(1):26–39, 2015.

[18] Mauricio Alvarez-Mesa, Sergio Sanz-Rodríguez, Maciej Glowiak, Eryk Skotarczak, and Ravi Velhal. Global 8k live streaming showcase 2020: The technologies behind the scenes. 2021.

[19] Christoph Bachhuber, Eckehard Steinbach, Martin Freundl, and Martin Reisslein. On the minimization of glass-to-glass and glass-to-algorithm delay in video communication. *IEEE Transactions on Multimedia*, 20(1):238–252, 2017.

[20] Alexandru Aloman, AI Ispas, Petrica Ciotirnae, Ramon Sanchez-Iborra, and Maria-Dolores Cano. Performance evaluation of video streaming using MPEG DASH, RTSP, and RTMP in mobile networks. In *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pages 144–151. IEEE, 2015.

[21] Henning Schulzrinne, Anup Rao, and Robert Lanphier. Real time streaming protocol (RTSP). 1998.

[22] Henning Schulzrinne, Stephen Casner, Ron Frederick, Van Jacobson, et al. RTP: A transport protocol for real-time applications, 1996.

[23] Smart Mobile Labs. What is the magic of realtime video streaming with hyperlow latency. 2018.

[24] G. Sullivan. Deployment status of the HEVC standard. document jctvc-ab0020 of jct-vc,, JCT-VC, Torino, Italy, 2017.

[25] Joint Video Exploration Team. Versatile video coding (vvc). https://jvet.hhi.fraunhofer.de/, 2018.

[26] Benjamin Bross, Jianle Chen, Jens-Rainer Ohm, Gary J Sullivan, and Ye-Kui Wang. Developments in international video coding standardization after avc, with an overview of versatile video coding (vvc). *Proceedings of the IEEE*, 2021.

[27] Convenor of MPEG. N17482: Versatile video coding project starts strongly in the joint video experts team. *MPEG Press Release*, 2018.

[28] Convenor of JVET. N15340: Requirements for a future video coding standard v1. In *MPEG 112th, Warsaw, Poland*, 2015.

[29] Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[30] Benjamin Bross, Ye-Kui Wang, Yan Ye, Shan Liu, Jianle Chen, Gary J Sullivan, and Jens-Rainer Ohm. Overview of the versatile video coding (vvc) standard and its applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3736–3764, 2021.

[31] Boyce Jill, Suehring Karsten, Li Xiang, and Seregin Vadim. Document: Jvet-j1010-v1, jvet common test conditions and software reference configurations. In *Joint Video Experts Team (JVET)*, 2018.

[32] E François, M Kerdranvat, R Julian, C Chevance, P DeLagrange, F Urban, T Poirier, and Y Chen. Vvc per-tool performance evaluation compared to hevc. In *IBC*, 2020.

[33] Chen Jianle and Hwan Kim Yan, Ye andSeung. Algorithm description for versatile video coding and test model 12 (vtm 12). 2021.

[34] T. Grois D., Nguyen and Marpe D. Performance comparison of av1, jem, vp9, and hevc encoders. In *Applications of Digital Image Processing XL*, volume 10396, page 103960L. International Society for Optics and Photonics, 2018.

[35] P. Hanhart and T. Ebrahimi. Calculation of average coding efficiency based on subjective quality scores. *Journal of Visual Communication and Image Representation*, 25(3):555–564, 2014.

[36] A Tissier, Alexandre Mercat, Thomas Amestoy, Wassim Hamidouche, Jarno Vanne, and Daniel Menard. Complexity reduction opportunities in the future vvc intra encoder. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2019.

[37] Farhad Pakdaman, Mohammad Ali Adelimanesh, Moncef Gabbouj, and Mahmoud Reza Hashemi. Complexity analysis of next-generation vvc encoding and decoding. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3134–3138. IEEE, 2020.

[38] Mário Saldanha, Gustavo Sanchez, César Marcon, and Luciano Agostini. Complexity analysis of vvc intra coding. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3119–3123. IEEE, 2020.

[39] Tianyi Li, Mai Xu, Runzhi Tang, Ying Chen, and Qunliang Xing. Deepqtmt: A deep learning approach for fast qtmt-based cu partition of intra-mode vvc. *IEEE Transactions on Image Processing*, 2021.

[40] Alexandre Tissier, Wassim Hamidouche, J Vanney, F Galpinz, and Daniel Menard. Cnn oriented complexity reduction of vvc intra encoder. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3139–3143. IEEE, 2020.

[41] Jens Brandenburg, Adam Wieckowski, Tobias Hinz, Anastasia Henkel, Valeri George, Ivan Zupancic, Christian Stoffers, Benjamin Bross, Heiko Schwarz, and Detlev Marpe. Towards fast and efficient vvc encoding. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2020.

[42] Wei Ding and Bede Liu. Rate control of mpeg video coding and recording by rate-quantization modeling. *IEEE transactions on circuits and systems for video technology*, 6(1):12–20, 1996.

[43] Siwei Ma, Wen Gao, and Yan Lu. Rate-distortion analysis for H.264/AVC video coding and its application to rate control. *IEEE transactions on circuits and systems for video technology*, 15(12):1533–1544, 2005.

[44] Liang-Jin Lin and Antonio Ortega. Bit-rate control using piecewise approximated rate-distortion characteristics. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(4):446–459, 1998.

[45] Xi Min Zhang, Anthony Vetro, Yun Q Shi, and Huifang Sun. Constant quality constrained rate allocation for FGS-coded video. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(2):121–130, 2003.

[46] Meng Liu, Yi Guo, Houqiang Li, and Chang Wen Chen. Low-complexity rate control based on ρ-domain model for scalable video coding. In *Proc. IEEE International Conference on Image Processing*, pages 1277–1280, 2010.

[47] Minhao Tang, Jiangtao Wen, and Yuxing Han. A generalized rate-distortion-λ model based HEVC rate control algorithm. *arXiv preprint arXiv:1911.00639*, 2019.

[48] Bin Li, Houqiang Li, Li Li, and Jinlei Zhang. $lambda$ domain rate control algorithm for high efficiency video coding. *IEEE transactions on Image Processing*, 23(9):3841–3854, 2014.

[49] Stefan Lederer, Christopher Müller, and Christian Timmerer. Dynamic adaptive streaming over http dataset. In *Proceedings of the 3rd multimedia systems conference*, pages 89–94, 2012.

[50] Christian Timmerer and Carsten Griwodz. Dynamic adaptive streaming over http: from content creation to consumption. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 1533–1534, 2012.

[51] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[52] Thorsten Lohmar, Torbjörn Einarsson, Per Fröjdh, Frédéric Gabin, and Markus Kampmann. Dynamic adaptive http streaming of live content. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–8. IEEE, 2011.

[53] Christian Timmerer, Daniel Weinberger, Martin Smole, Reinhard Grandl, Christopher Müller, and Stefan Lederer. Live transcoding and streaming-as-a-service with mpeg-dash. In *2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–4. IEEE, 2015.

[54] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. From theory to practice: Improving bitrate adaptation in the DASH reference player. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(2s):1–29, 2019.

[55] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Feedback control for adaptive live video streaming. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 145–156, 2011.

[56] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.

[57] Wei Bao and Stefan Valentin. Bitrate adaptation for mobile video streaming based on buffer and channel state. In *2015 IEEE International Conference on Communications (ICC)*, pages 3076–3081. IEEE, 2015.

[58] Chao Zhou, Chia-Wen Lin, and Zongming Guo. mDASH: A markov decision-based rate adaptation approach for dynamic HTTP streaming. *IEEE Transactions on Multimedia*, 18(4):738–751, 2016.

[59] Maxim Claeys, Steven Latré, Jeroen Famaey, Tingyao Wu, Werner Van Leekwijck, and Filip De Turck. Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming. In *Adaptive and Learning Agents Workshop, part of AAMAS2013 (ALA-2013)*, pages 30–37, 2013.

[60] Kuntai Du, Ahsan Pervaiz, Xin Yuan, Aakanksha Chowdhery, Qizheng Zhang, Henry Hoffmann, and Junchen Jiang. Server-driven video streaming for deep learning inference. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 557–570, 2020.

[61] Zhimin Xu, Xinggong Zhang, and Zongming Guo. QoE-driven adaptive K-push for HTTP/2 live streaming. *IEEE Transactions on Circuits and Systems for Video Technology*, 29(6):1781–1794, 2018.

[62] Mariem Ben Yahia, Yannick Le Louedec, Gwendal Simon, Loutfi Nuaymi, and Xavier Corbillon. HTTP/2-based frame discarding for low-latency adaptive video streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 15(1):1–23, 2019.

[63] Tongtong Feng, Haifeng Sun, Qi Qi, Jingyu Wang, and Jianxin Liao. Vabis: video adaptation bitrate system for time-critical live streaming. *IEEE Transactions on Multimedia*, 22(11):2963–2976, 2019.

[64] Theodoros Karagkioules, Cyril Concolato, Dimitrios Tsilimantos, and Stefan Valentin. A comparative case study of HTTP adaptive streaming algorithms in mobile networks. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 1–6, 2017.

[65] Sylvester J Udoh and Viranjay M Srivastava. Analytical modeling of radio network performance for 5G (non-standalone) and it's network connectivity. *J. Commun.*, 15(12):886–895, 2020.

[66] Robert Peck, Jordi Cenzano, Xiangbo Li, and Yuriy Reznik. Towards mass deployment of cmaf. In *Proceedings of NAB Broadcast Engineering and Information Technology Conference*, 2019.

[67] Lei Kang, Wei Zhao, Bozhao Qi, and Suman Banerjee. Augmenting self-driving with remote control: Challenges and directions. In *Proc. of the 19th International Workshop on Mobile Computing Systems & Applications*, pages 19–24, 2018.

[68] Sourabh Khire, Scott Robertson, Nikil Jayant, Elena A Wood, Max E Stachura, and Tamer Goksel. Region-of-interest video coding for enabling surgical telementoring in low-bandwidth scenarios. In *Proc. IEEE Military Communications Conference*, pages 1–6, 2012.

[69] Gary J Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on circuits and systems for video technology*, 22(12):1649–1668, 2012.

[70] Joint Video Exploration Team JVET. Versatile video coding (VVC). https://jvet.hhi.fraunhofer.de/, 2019.

[71] Alliance for Open Media AOMedia. AV1. https://github.com/AOMediaCodec/av1-spec, 2019.

[72] Teruhiko Suzuki. JVET-d1002 : Work plan for assessment of test material. *JVET 4th Meeting, Chengdu, CN.*, 2016.

[73] Mourad Aklouf. Magnycours Video Sequence (EKTACOM), 12 2021.

[74] Mohammad Kazemi, Mohammad Ghanbari, and Shervin Shirmohammadi. A review of temporal video error concealment techniques and their suitability for HEVC and VVC. *Multimedia Tools and Applications*, pages 1–46, 2021.

[75] Joerg Ott, Stephan Wenger, Noriyuki Sato, Carsten Burmeister, and Jose Rey. Extended RTP profile for real-time transport control protocol (RTCP)-based feedback (RTP/AVPF). *Request for Comments*, 4585, 2006.

[76] Dixon David Salcedo Morillo, Julián Guerrero, and Cesar D Guerrero. Overhead in available bandwidth estimation tools: Evaluation and analysis. 2017.

[77] FFmpeg Developers. ffmpeg tool (build: ffmpeg-20180716-8aa6d9a-win64-static). http://ffmpeg.org/, 2019.

[78] Gyokov Solutions. G-NetTrack Lite. https://gyokovsolutions.com/g-nettrack/, 2017.

[79] QUALCOMM. QxDM Professional QUALCOMM eXtensible Diagnostic Monitor. https://developer.qualcomm.com/forum/qdn-forums/software/hexagon-dsp-sdk/toolsinstallation/34529, 2012.

[80] Li Yuanjie, Peng Chunyi, and Lu Songwu. Mobileinsight, a fine-grained mobile network analytics inside the smartphones. http://www.mobileinsight.net/, 2021.

[81] Y. Ye J. Chen and S. Kim. Jvet-n1002-v1: Algorithm description for versatile video coding and test model 5 (vtm 5). In *Jvet 14th Meeting: Geneva, CH, 19 27 Mar. 2019*.

[82] Teruhiko Suzuki. Jvet-d1002 : Work plan for assessment of test material. *JVET 4th Meeting, Chengdu, CN.*, 2016.

[83] FFmpeg Developers. ffmpeg tool (build: ffmpeg-20180716-8aa6d9a-win64-static). http://ffmpeg.org/, 2019.

[84] Fraunhofer Heinrich Hertz Institute (HHI). Fraunhofer Versatile Video Encoder (VVenC). https://github.com/fraunhoferhhi/vvenc, 2021.

[85] Brandenburg Jens, Wieckowski Adam, Hinz Tobias, and Bross Video Benjamin. Vvenc fraunhofer versatile video encoder v1.0.0. 2021.

[86] Shuai Li, Ce Zhu, Yanbo Gao, Yimin Zhou, Frédéric Dufaux, and Ming-Ting Sun. Lagrangian multiplier adaptation for rate-distortion optimization with inter-frame dependency. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(1):117–129, 2015.

[87] Jing He, En-Hui Yang, Fuzheng Yang, and Kehu Yang. Adaptive quantization parameter selection for h. 265/hevc by employing inter-frame dependency. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(12):3424–3436, 2017.

[88] Mah-Rukh Fida, Andres Ocampo, and Ahmed Elmokashfi. Measuring and localising congestion in mobile broadband networks. *IEEE Transactions on Network and Service Management*, 2021.

[89] Guanghui Zhang and Jack YB Lee. LAPAS: Latency-aware playback-adaptive streaming. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2019.

[90] Nesrine Changuel. *Régulation de la qualité lors de la transmission de contenus vidéo sur des canaux sans fils*. PhD thesis, Paris 11, 2011.

[91] Saamer Akhshabi, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali C Begen. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In *Proceeding of the 23rd ACM workshop on network and operating systems support for digital audio and video*, pages 19–24, 2013.

[92] Koffka Khan and Wayne Goodridge. What happens when adaptive video streaming players compete in time-varying bandwidth conditions? *International Journal of Advanced Networking and Applications*, 10(1):3704–3712, 2018.

[93] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[94] Jie Liu, Xiaoming Tao, and Jianhua Lu. QoE-oriented rate adaptation for DASH with enhanced deep Q-learning. *IEEE Access*, 7:8454–8469, 2018.

[95] Ektacom. Nomade Cars. https://www.ektacom.net/nomade, 2021.

[96] Ektacom. Nomade Sails. https://www.ektacom.net/sails, 2021.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Vidéo pour l'événementiel : Compression et transport de la nouvelle génération de codec vidéo
Mots clés: transmission basse latence, streaming vidéo, codage adaptatif de la vidéo, contrôle du débit, Versatile Video Coding, compression video

Résumé: L'acquisition et la diffusion de contenus avec une latence minimale sont devenus essentiel dans plusieurs domaines d'activités tels que la diffusion d'évènements sportifs, la vidéoconférence, la télé-opération de véhicules ou le contrôle à distance. L'industrie de la diffusion en direct a connu en 2020 une forte croissance et va encore croitre au cours des prochaines années grâce à l'émergence de nouveaux codecs vidéo à haute efficacité tel que Versatile Video Coding (VVC), et à la cinquième génération de réseaux mobiles (5G).

Les méthodes de streaming de type HTTP Adaptive Streaming (HAS) telles que MPEG-DASH, grâce aux algorithmes d'adaptation du débit, se sont révélées très efficaces pour améliorer la qualité d'expérience (QoE) dans un contexte de vidéo à la demande (VOD). Ces algorithmes d'adaptation sont mis au niveau du client. Ils exploitent les mesures du débit du réseau et/ou du niveau de remplissage du tampon de réception afin d'optimiser la QoE du client.

Dans les applications où la latence est critique, minimiser le délai entre l'acquisition de l'image et son affichage au récepteur est essentiel. La plupart des algorithmes d'adaptation de débit sont développés pour optimiser la transmission vidéo d'un serveur situé dans le cœur de réseau vers des clients mobiles. Dans les applications nécessitant un streaming à faible latence, telles que le contrôle à distance de drones, le rôle du serveur est joué par un terminal mobile qui va acquérir, compresser et transmettre les images via une liaison montante comportant un canal radio vers un ou plusieurs clients. Les approches d'adaptation de débit pilotées par le client sont par conséquent inadaptées dans ce contexte à cause de la variabilité des caractéristiques du canal. De plus, les HAS, pour lesquelles la prise de décision se fait avec une périodicité de l'ordre de la seconde ne sont pas suffisamment réactives lors d'une mobilité importante du serveur et peuvent engendrer des délais importants. Il est donc essentiel d'utiliser une granularité d'adaptation très fine.

L'objet de cette thèse est d'apporter des éléments de réponse à la problématique de la transmission vidéo à faible latence depuis des émetteurs mobiles. Nous présentons d'abord un algorithme d'adaptation de débit image-par-image pour la diffusion à faible latence. Une approche de type Model Predictive Control (MPC) est proposée pour déterminer le débit de codage de chaque image à transmettre. Cette approche utilise des informations relatives au niveau de tampon de l'émetteur et aux caractéristiques du canal de transmission. Les images étant codées en direct, un modèle reliant le paramètre de quantification (QP) au débit de sortie du codeur vidéo est nécessaire. Nous avons donc proposé un nouveau modèle reliant le débit au paramètre de quantification et à la distorsion de l'image précédente. Ce modèle fournit de bien meilleurs résultats dans le contexte d'une décision prise image par image du débit de codage que les modèle de référence de la littérature.

En complément des techniques précédentes, nous avons également proposé des outils permettant de réduire la complexité de codeurs vidéo tels que VVC. Par rapport à son prédécesseur, le High-Efficiency Video Coding (HEVC), ce codeur vidéo permet de réduire de moitié la quantité de bits à transmettre à qualité équivalent. Cependant, les nouveaux outils introduits dans le standard VVC conduisent à une explosion de la complexité. La version actuelle du codeur VVC (VTM10) a un temps d'exécution dix fois supérieur à celui du codeur HEVC. Par conséquent, le codeur VVC n'est pas adapté aux applications de codage et diffusion en temps réel sur les plateformes actuellement disponibles. Dans ce contexte, nous présentons une méthode systématique, de type branch-and-prune, permettant d'identifier un ensemble d'outils de codage pouvant être désactivés tout en satisfaisant une contrainte sur l'efficacité de codage. Ce travail contribue à la réalisation d'un codeur VVC temps réel.

Title: Video for events : Compression and transport of the next generation video codec
Keywords: low latency transmission, video streaming, adaptive video coding, rate control, video compression

Abstract: The acquisition and delivery of video content with minimal latency has become essential in several business areas such as sports broadcasting, video conferencing, remote vehicle operation, or remote system control. The live streaming industry has grown in 2020 and will expand further in the next few years with the emergence of new high-efficiency video codecs such as the Versatile Video Coding (VVC) standard, and the fifth generation of mobile networks (5G).

HTTP Adaptive Streaming (HAS) methods such as MPEG-DASH, using algorithms to adapt the transmission rate of compressed video, have proven to be very effective in improving the quality of experience (QoE) in a video-on-demand (VOD) context. Most of these adaptation algorithms are implemented at the client level. They exploit measurements of network throughput and/or receive buffer level to optimize the client's QoE.

Nevertheless, minimizing the delay between image acquisition and display at the receiver is essential in applications where latency is critical. Most rate adaptation algorithms are developed to optimize video transmission from a server situated in the core network to mobile clients. In applications requiring low-latency streaming, such as remote control of drones, the role of the server is played by a mobile terminal. The latter will acquire, compress, and transmit the video and transmit the compressed stream via a radio access channel to one or more clients. Therefore, client-driven rate adaptation approaches are unsuitable in this context because of the variability of the channel characteristics. In addition, HAS, for which the decision-making is done with a periodicity of the order of a second, are not sufficiently reactive when the server is moving, which may generate significant delays. It is therefore important to use a very fine adaptation granularity.

The aim of this thesis is to provide some answers to the problem of low-latency delivery of video acquired, compressed, and transmitted by mobile transmitters. We first present a frame-by-frame rate adaptation algorithm for low latency broadcasting. A Model Predictive Control (MPC) approach is proposed to determine the coding rate of each frame to be transmitted. This approach uses information about the buffer level of the transmitter and about the characteristics of the transmission channel. Since the frames are coded live, a model relating the quantization parameter (QP) to the output rate of the video encoder is required. We have proposed a new model linking the rate to the QP of the current frame and to the distortion of the previous frame. This model provides much better results in the context of a frame-by-frame decision on the coding rate than the reference models in the literature.

In addition to the above techniques, we have also proposed tools to reduce the complexity of video encoders such as VVC. Compared to its predecessor, High-Efficiency Video Coding (HEVC), this video encoder can reduce the number of bits to be transmitted by half at equivalent quality. Nevertheless, the new tools introduced in the VVC standard lead to an explosion of complexity. The current version of the VVC encoder (VTM10) has an execution time ten times higher than that of the HEVC encoder. Therefore, the VVC encoder is not suitable for real-time encoding and streaming applications on currently available platforms. In this context, we present a systematic branch-and-prune method to identify a set of coding tools that can be disabled while satisfying a constraint on coding efficiency. This work contributes to the realization of a real-time VVC coder.