# Compression for interactive communication of visual contents

Navid Mahmoudian Bidgoli

# THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
**C**OMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
*Mathèmatique et Sciences et Technologies de l'Information et de la Communication*
Spécialité : *Signal, Image, Vision*

Par

**Navid MAHMOUDIAN BIDGOLI**

## Compression for interactive communication of visual contents

**Thèse présentée et soutenue à** RENNES , le 29 novembre 2019
**Unité de recherche : SIROCCO, Inria Rennes - Bretagne Atlantique**

**Rapporteurs avant soutenance :**

| | |
|---|---|
| Michel KIEFFER | Professor, Université Paris-Sud |
| Kenneth ROSE | Professor, University of California Santa Barbara, US |

**Composition du jury :**

| | | |
|---|---|---|
| Président : | Luce MORIN | Professor, INSA, Rennes |
| Examinateurs : | Joao ASCENSO | Professor, Instituto Superior Técnico, Portugal |
| | Michel KIEFFER | Professor, Université Paris-Sud |
| | Enrico MAGLI | Professor, Politecnico di Torino, Italie |
| Dir. de thèse : | Aline ROUMY | Researcher, Inria Rennes - Bretagne Atlantique |
| Co-dir. de thèse : | Thomas MAUGEY | Researcher, Inria Rennes - Bretagne Atlantique |

# Acknowledgement

This thesis would not have been completed without the support of many persons to whom I am extremely grateful. I was quite lucky to be surrounded by these incredible people who accompanied me shoulder to shoulder during my PhD study. They were always willing to share their experiences, exchange their ideas, and provide advices in times of need.

First of all, I wish to express my deepest gratitude to my best supervisors: Aline and Thomas. I should emphasize that you were not acting only as my supervisors who guided me in my academic life, but also your supports went beyond my doctoral work as if you were part of my family who are supporting me a lot also in my daily life. I have learned a lot from both of you. Your insightful expertise and meticulous guidance were crucial for this thesis, and it was not possible to make it feasible without your patience, advices, and availability.

I would also like to express my gratitude towards all my thesis jury members: Prof. João Ascenso, Prof. Michel Kieffer, Prof. Enrico Magli, Prof. Luce Morin, Prof. Kenneth Rose (in alphabetical order). Thank you for kindly agreeing to be a part of this committee and for the time and efforts invested in the evaluation of this thesis.

I would also like to thank Huguette Bechu for always assisting me to do my administration stuff. My special thanks go to Prof. Kadi Bouatouch, who is always supporting me and being so kind to me. I always appreciate discussing with him, and I will never forget him and his endless help. I would also like to thank Prof. Christine Guillemot for supporting and welcoming me to the SIROCCO team.

Inria is a wonderful place that not only provided me a wonderful opportunity to meet and work with brilliant people, but it also gave me the chance to obtain valuable friends. Thanks to all the people I met there who made my life much easier: Pierre, Javad, Roberto, Pierre, Elian, Thierry, Simon, Laurent, Fatma, Xiaoran, Maja, Cédric, Olivier, Ehsan, Fatemeh, Nam, Mai, Mira, Alex, Jinglei, Zhaolin, Lara (in alphabetical order of their family name). I hope our friendship continues.

I would also like to thank my Iranian friends: Mohsen, Fatemeh, Sina, Mina, Mohsen, Omid, etc. You all made my daily life more interesting, which made it easy for me to get through the ups and downs of my PhD journey.

Most importantly, I have to admit that none of my achievements would ever have been possible without the unwavering support of my beloved family. I deeply thank my parents (Farzaneh and Mohammadreza) for everything they have been doing for me, for their love and encouragement. I would also like to thank my lovely wife Neda from the bottom of my heart for being my dearest companion through the ups and downs of this long journey. Thank you again for all the enormous and heroic sacrifices all of you have made for me. Many thanks go to my other family members as well: Farisha, Reza, Amirabbas, Maral, Nahid, Farrokh, Homayoun, Farid, Morteza, Parvaneh, Ahmadreza, Roya, Parvin, Ziba, Mohammad, etc.

# TABLE OF CONTENTS

# List of Abbreviations

**2D** two-dimensional.

**3D** three-dimensional.

**AI** All Intra.

**AR** Augmented Reality.

**BD** Bjontegaard-Delta.

**BIAWGN** Binary Input Additive White Gaussian Noise.

**BP** Belief Propagation.

**BSC** Binary Symmetric Channel.

**CN** Check node.

**DCT** Discrete Cosine Transform.

**DE** Density Evolution.

**DOF** Degrees of Freedom.

**DPCM** Differential Pulse-Code Modulation.

**DSC** Distributed Source Coding.

**ES** Exhaustive Storage.

**EXIT** extrinsic information transfer.

**FOV** Field of View.

**FTV** free-viewpoint television.

**GA** Geometry-Aware.

**GFT** Graph Fourier Transform.

**HEALPix** Hierarchical Equal Area isoLatitude Pixelization.

**HEVC** High Efficiency Video Coding.

**HMD** Head-Mounted Display.

**i.i.d.** independent and identically-distributed.

**IP** Internet Protocol.

**KL** Kullback–Leibler.

**LDPC** low-density parity-check.

**LDPCA** LDPC accumulate.

**LLR** Log-Likelihood Ration.

**M-frame** merge-frame.

**PMF** Probability Mass Function.

**PSNR** Peak Signal-to-Noise Ratio.

**QER** Quality Emphasized Region.

**QP** Quantization Parameter.

**RD** rate-distortion.

**S-block** Spherical Block.

**S-R-D** Storage-Rate-Distortion.

**SD** storage-distortion.

**SI** Side Information.

**SMRA** Sequential Massive Random Access.

**SW** Slepian-Wolf.

**VN** Variable node.

**VR** Virtual Reality.

# RÉSUMÉ EN FRANCAIS

Dans la communication interactive de contenu visuel, l'objectif est de donner à l'utilisateur la possibilité de naviguer librement dans une scène tridimensionnelle (3D). Par exemple, dans un système vidéo interactif, l'utilisateur peut changer l'emplacement ou l'orientation de la caméra et regarder la scène du point de vue qu'il souhaite. De nos jours, il existe différents systèmes de communication interactifs pour le contenu visuel, tels que les visites virtuelles, la télévision à point de vue libre, la réalité virtuelle 3D (RV) et la réalité augmentée (RA) pour explorer des lieux emblématiques en 3D, etc. Ces systèmes diffèrent par le nombre de degrés de liberté (DDL) et le type de contenu visuel (réel ou virtuel) qu'ils fournissent.

Ces systèmes de communication interactifs sont de plus en plus utilisés de nos jours car ils offrent une meilleure expérience utilisateur et peuvent donc attirer davantage l'attention. Par exemple, différentes études montrent que les clients sont plus susceptibles (presque deux fois plus) de visiter une entreprise (par exemple, restaurant, magasin, vente / location d'appartements) lorsqu'une visite virtuelle interactive (visite virtuelle) est disponible [93, 37]. De plus, la technologie nécessaire pour fournir et présenter ces contenus a été améliorée et est devenue plus abordable. Selon Cisco Visual Networking Index, le trafic RV et RA Internet Protocol (IP) sera multiplié par 12 entre 2017 et 2022 dans le monde [17].

Un exemple de média interactif avec 3 DDL sont les images et vidéos omnidirectionnelles. Ce type de contenu est utilisé dans différentes applications telles que les systèmes d'information géographique comme Google Street View [3] (dans lequel chaque point de capture est une image omnidirectionnelle), Cinematic RV [111] et la robotique. Un contenu omnidirectionnel peut être modélisé comme un signal sphérique car il offre une vue à 360 degrés de la scène à partir d'un seul point. Lors de la visualisation de ces images, les utilisateurs sont placés au centre de la sphère, et ils peuvent librement changer leur direction de visualisation et observer la partie de la sphère qui leur fait face à l'aide d'un Casque de Réalité Virtuelle (CRV).

Les modèles et environnements 3D vont plus loin que les contenus sphériques en permettant à l'utilisateur de regarder plus naturellement autour d'un environnement RV avec 6 DDL (dont trois en translation et trois en orientation) [41]. Un modèle 3D doit être représenté dans une structure adaptée au rendu et à l'affichage. La structure la plus courante est le maillage 3D [79, Chapitre 14]. Un objet maillé 3D est généralement décrit comme une combinaison de plusieurs entités, y compris des informations géométriques (un ensemble de polygones en 3D) et des atlas / cartes de texture qui sont mappés à la surface de l'objet 3D au stade du rendu. Avec les progrès récents des algorithmes de reconstruction 3D automatisés, des cartes 3D en haute qualité de lieux et d'objets sont largement accessibles. Dans ces environnements 3D virtuels, le système simule la présence physique des utilisateurs et une vision humaine naturelle à l'aide d'une caméra perspective virtuelle (avec un champ de vision spécifique) se déplaçant dans la scène. De cette façon, les utilisateurs peuvent naviguer librement dans la scène et découvrir une

partie du modèle 3D qui les intéresse avec beaucoup de réalisme [41]. En regardant ces modèles 3D, seule une partie du modèle et par conséquent une partie des informations texturales sont observées à un moment donné.

Pour ressentir un réel sentiment d'immersion, tel que l'utilisateur se sent dans un monde virtuel qui est une copie précise de la réalité, il est nécessaire de capturer et de stocker les données en haute résolution avec de grandes tailles (grande base de données). Cependant, lors de l'interaction, seule une petite partie des données est demandée par chaque utilisateur et cette partie est elle-même également de haute qualité en soi. Par exemple, de nos jours avec les HMD récents, la résolution spatiale de la fenêtre d'affichage (c'est-à-dire l'écran qui montre la partie des données visuelles que l'utilisateur observe) a une résolution d'au moins 4K (4096 x 2160). Cela oblige l'ensemble du contenu sphérique à avoir une résolution d'au moins 12K [20]. Cela vaut également pour une carte de texture d'un modèle 3D. Par exemple, la carte de texture d'un site historique montré dans [47] a une carte de texture de taille 8192x8192 pixels.

L'interactivité permise par ces supports a suscité de nouveaux défis pour leur compression car elle nécessite la distinction entre les taux de stockage et de transmission. D'une part, pour compresser efficacement ces contenus, l'encodeur doit utiliser toutes les dépendances existantes entre les données. De cette façon, les données peuvent être stockées dans l'état le plus compressé sur le serveur. En revanche, lors de la remise des données, seule une sous-partie des données compressées est demandée par l'utilisateur, et cette demande est aléatoire du point de vue codeur. Le caractère aléatoire de la demande est dû au fait que la demande dépend du choix de l'utilisateur et que le système n'en a pas connaissance lors de la compression. Par conséquent, la base de données doit être compressée et stockée en tenant compte de toutes les dépendances, mais le schéma de compression doit également avoir la capacité d'extraire et d'envoyer uniquement la partie demandée dans l'état le plus compressé. Cependant, une compression efficace entre en conflit avec l'idée d'un accès aléatoire à une partie des données car les données sont compressées compte tenu des dépendances avec des parties non demandées également. Plus précisément, une extraction de sous-ensemble pour réduire le coût de transmission est généralement incompatible avec la compression conjointe visant à réduire la taille de stockage.

## Motivation

Les méthodes de compression classiques disponibles ne permettent pas d'accéder à une partie des données compressées (possibilité d'accès aléatoire). Cependant, elles peuvent être adaptées pour répondre aux contraintes d'accès aléatoire. Il existe plusieurs façons de procéder. Une façon consiste à envoyer l'intégralité de la base de données compressée à chaque utilisateur. Bien que cette méthode soit appropriée en termes de stockage, elle est certainement inefficace en termes de transmission (livraison) de données. La deuxième méthode consiste à diviser la base de données en morceaux séparés et à compresser chaque partie séparément. Lors de la transmission, seules les morceaux demandés par l'utilisateur sont envoyés. Cette méthode réduit le coût de transmission (car seuls les morceaux liés à la demande de l'utilisateur sont transmis), mais elle n'est pas adaptée au stockage car elle ne prend pas en compte les dépendances entre morceaux. En outre, l'utilisateur peut ne demander qu'une petite fraction de morceau, mais le système doit toujours envoyer le morceau entier. Ainsi, cette méthode ne parvient pas à atteindre le coût de transmission d'un *codeur Oracle*, c'est-à-dire un codeur optimal hypothétique

qui connaît les demandes à l'avance et code uniquement les parties demandées. En d'autres termes, cette méthode n'est pas parfaitement optimale du point de vue de la transmission également.

Pour atteindre le taux de transmission idéal, comme troisième méthode, on pourrait coder et stocker toutes les demandes possibles, mais cette solution exhaustive entraîne un coût de stockage énorme. Comme quatrième méthode, on peut supposer que la base de données entière est décodée sur le serveur et que seule la partie demandée est recompressée et envoyée. Bien que cette méthode soit à la fois optimale en termes de stockage et de transmission, elle est en pratique irréalisable, notamment lorsque le nombre d'utilisateurs est élevé, car elle impose un calcul étendu au serveur.

On peut voir que les méthodes existantes ne peuvent pas atteindre le taux de transmission optimal à moins qu'elles ne sacrifient la taille de stockage avec une surcharge élevée. Fait intéressant, il a été récemment montré dans [70] qu'en utilisant des codes incrémentiels et avec une surcharge de stockage raisonnable, un ensemble de données peut être compressé conjointement avec une capacité d'accès aléatoire de sorte qu'à la demande de l'utilisateur, seule la partie requise des données puisse être extraite de la compression données et transmises à l'utilisateur. Plus précisément, en utilisant les codes incrémentiels, nous pouvons nous attendre à maintenir les performances de codage du point de vue du débit de transmission (par rapport au codeur Oracle) avec une surcharge de stockage raisonnable.

Comme on peut le voir, il existe un écart entre ce qui peut être réalisé théoriquement [70] et ce qui a déjà été réalisé dans la pratique. L'objectif de cette thèse est de combler le fossé entre la théorie et la pratique en proposant des solutions pratiques pour la compression des médias interactifs. Nous construisons un codeur interactif qui compresse l'image entière mais est capable d'extraire et de transmettre une sous-partie de celle-ci de manière compressée. Ce codeur interactif est générique dans le sens où il peut gérer différents médias interactifs. Cette généricité est obtenue en projetant les médias interactifs sur une image deux dimensions (2D). Dans cette thèse, nous considérerons deux modalités d'image : les images 360° et la carte de texture des modèles 3D. Cependant, le codeur générique proposé est également adapté à d'autres scénarios de visualisation interactive (par exemple, panorama, images à ultra-haute résolution, carte de ville / pays). Pour atteindre cet objectif, nous devons répondre à quatre questions essentielles à cet égard :

— *Comment évaluer différents schémas de codage interactifs pour la compression ?* En s'éloignant des méthodes de communication classiques vers une communication interactive, le premier point qui attire l'attention est la nécessité de distinguer les taux de stockage et de transmission. Ce n'est pas le cas dans les méthodes classiques dans lesquelles toutes les données stockées sont envoyées à l'utilisateur. Étant donné que les données sont généralement compressées avec perte (ce qui introduit une distorsion), la question se pose de savoir comment comparer les différentes méthodes qui fournissent une communication interactive du point de vue de la compression, alors que les trois facteurs, à savoir le stockage, les taux de transmission et les distorsions sont pris en compte ensemble.

— *Comment relier la compression interactive de contenus visuels 2D aux résultats théoriques obtenus dans [70] ?* L'objectif est de rechercher le lien entre le résultat lié à la théorie de l'information et la tâche pratique de compression des images 2D à accès aléatoire. Pour cela, il faut une définition de problème qui permette de modéliser dans l'image 2D les équivalents des définitions telles que la source, les informations secondaires (comme aide pour compresser les données), le voisinage

et les dépendances entre les sources, etc., qui sont définis dans [70]. Cela conduit à la définition d'une stratégie de codage générique.

— *Comment intégrer cette structure générique de codage dans un codeur interactif pratique pour les deux modalités d'imagerie considérées dans cette thèse (images 360° et cartes de texture des modèles 3D) ?* Chacune de ces modalités a ses propres caractéristiques uniques. Par conséquent, il est nécessaire de prendre en compte les hypothèses concernant la demande de l'utilisateur qui sont spécifiques à chaque modalité. Plus précisément, pour chaque média, le voisinage entre sources permettant aux utilisateurs de naviguer entre eux doit être déterminé. De plus, la représentation 2D n'est utilisée, dans la communauté, que pour avoir une représentation commune entre différents médias interactifs. Cependant, la géométrie sous-jacente des données n'est pas nécessairement bidimensionnelle. Par exemple, le contenu à 360° se trouve sur une surface sphérique ou les informations de texture se trouvent sur le maillage d'un modèle 3D. Par conséquent, il est également nécessaire de prendre en compte les propriétés spécifiques de la géométrie des données.

— *Comment économiser du taux avec la sélection du modèle ?* Le codeur incrémental proposé dans [70] est basé sur un codage *basé sur un modèle*, c'est-à-dire que la distribution conjointe entre les sources, appelée modèle, est nécessaire à la fois au codeur et au décodeur pour compresser et décoder les sources. Par conséquent, connaître le modèle est une hypothèse clé. Étant donné que la nature du contenu visuel n'est pas stationnaire, l'envoi de la description du modèle au décodeur est essentiel. Cependant, l'envoi du modèle est parfois très coûteux et nécessite beaucoup de bits pour les décrire. Par conséquent, il est nécessaire d'estimer la relation entre le nombre de bits que nous pouvons économiser en envoyant un modèle approximatif plus simple et le débit nécessaire pour décoder la source à l'aide de ce modèle approximatif.

## Feuille de route de la thèse

La thèse est divisée en 5 chapitres. Après cette introduction, le Chapitre 1 passe en revue l'état de l'art de la compression pour la communication interactive. Plus précisément, à l'aide de l'introduction d'un graphique de navigation, nous proposons un cadre commun par lequel nous pouvons comparer différentes approches du point de vue de la théorie de l'information. Nous calculons les taux de transmission et de stockage réalisables de chaque approche et les comparons. Nous passons ensuite en revue les solutions existantes proposées pour compresser de manière interactive les deux modalités d'image 2D abordées dans ce travail : contenu à 360° et carte de texture des modèles 3D. Pour cela, tout d'abord, nous expliquons comment ces contenus sont présentés avec des plans 2D, puis nous expliquons comment la compression d'interactivité est fournie pour ces modalités.

Ensuite, dans les chapitres suivants, nous présentons nos différentes contributions qui répondent aux quatre questions soulevées dans la section précédente (Section Motivation).

— Chapitre 2 : Ce chapitre répond à la première question. Malgré de nombreuses contributions qui ont été proposées jusqu'à présent pour la compression interactive d'images 2D, peu a été fait pour évaluer différentes approches interactives. Ici, nous proposons un cadre d'évaluation qui prend en compte le stockage, le taux de transmission et la distorsion du côté de l'utilisateur. Cela permet de comparer différentes méthodes en fonction de la capacité de stockage sur le serveur et

de la bande passante réseau des utilisateurs.

— Chapitre 3 : Dans ce chapitre, nous abordons la deuxième question de la section précédente (Section Motivation). Nous proposons un nouveau schéma de compression interactif générique pour les images 2D qui atteint le coût de transmission Oracle tout en conservant un coût de stockage raisonnable. Tous les schémas de compression d'image 2D conventionnels fonctionnent par blocs. Pour avoir une compression efficace, il est nécessaire de prévoir chaque bloc en fonction des blocs déjà décodés. Les schémas classiques utilisent un ordre de balayage prédéfini fixe pour le codage et le décodage des blocs, ce qui contredit l'idée d'avoir une capacité d'accès aléatoire à une partie des données. Par conséquent, ces schémas imposent d'envoyer plus de blocs que nécessaire. En revanche, nous proposons une solution pour coder les blocs en utilisant des codes incrémentiels de sorte que tout ordre de décodage soit possible du côté du décodeur. Pour cela, pour chaque bloc à coder, plusieurs prédictions sont générées correspondant à différents ordres de décodage des blocs. Ensuite, un bitstream est généré pour chaque bloc sur la base de codes d'entropie incrémentiels. Avec ce bitstream, nous pouvons décoder le bloc pour la pire prédiction. Pour des prévisions de meilleure qualité, un sous-flux peut être extrait du même flux binaire pour effectuer l'opération de décodage. Par conséquent, seules les informations nécessaires sont transmises une fois la demande de l'utilisateur reçue.

— Chapitre 4 : Ce chapitre apporte des réponses à la troisième question soulevée dans (Section Motivation). Sur la base du codeur générique proposé en Chapitre 3, nous concevons des outils de compression pour encoder deux modalités d'imagerie qui assurent l'interactivité par leur nature : images 360° et cartes de texture des modèles 3D. Chacune de ces modalités d'imagerie a ses propres caractéristiques qui doivent être étudiées séparément. Pour chacune d'entre elles, nous proposons une stratégie qui permet de démarrer la procédure de décodage quelle que soit la demande de l'utilisateur. Les sources demandées sont décodées l'une après l'autre avec un ordre de balayage optimisé pour réduire le débit de transmission. Pour capturer les redondances au sein d'une image, nous proposons de nouveaux modes de prédiction adaptables à tout ordre de décodage et donc à toute demande de l'utilisateur. De plus, nous avons également proposé de nouvelles représentations pour chacune de ces modalités qui permettent de compresser les images plus efficacement.

— Chapitre 5 : Le codeur interactif proposé s'appuie sur des modèles de corrélation pour compresser la base de données. Plus le modèle est précis, meilleure est la compression des données. Cependant, puisque ce modèle doit être envoyé à l'utilisateur pour décodage, l'utilisation d'un modèle plus précis nécessite un grand nombre de bits pour décrire le modèle lui-même, ce qui augmente les taux de stockage et de transmission. Une autre façon de résoudre ce problème consiste à approximer le modèle de corrélation avec un modèle paramétrique plus simple et à le transférer à la place. Néanmoins, l'utilisation d'un modèle incorrect augmente le taux de compression. Par conséquent, lorsque nous décrivons un modèle avec un modèle approximatif, le nombre de bits que nous enregistrons dans la description du modèle doit être supérieur au nombre de bits que nous perdons en raison de la compression des données à l'aide de ce modèle approximatif. Le but de ce chapitre est de calculer ce taux excédentaire pour la compression interactive. Avec les outils fournis dans ce chapitre, nous pouvons sélectionner le modèle qui nécessite des débits in-

férieurs tout en prenant en compte simultanément la taille des données compressées et la taille de description du modèle.

Il est bon de noter que nous nous concentrons sur la compression d'images de scènes statiques sans considérer les aspects temporels. En fait, cette étape est la première étape clé des schémas de compression vidéo car elle permet la resynchronisation. Néanmoins, inspiré par les outils fournis dans cette thèse, le codeur générique proposé peut être étendu pour couvrir également l'aspect temporel.

Enfin, nous concluons cette thèse par un résumé et donnons un aperçu des directions possibles pour les travaux futurs. La liste des publications produites par l'auteur se trouve dans la section des publications de l'auteur.

# INTRODUCTION

In interactive communication of visual content, the goal is to give a user the ability to freely navigate within a three-dimensional (3D) scene. For example, in an interactive video system, the user can change the location or orientation of the camera and watch the scene from the viewpoint that he wants. Nowadays, there exist various interactive communication systems for visual content, such as virtual tours, free-viewpoint television, 3D Virtual Reality (VR) and Augmented Reality (AR) to explore iconic locations in 3D, etc. These systems differ in the number of Degrees of Freedom (DOF) and the type of visual content (real vs virtual) they provide.

These interactive communication systems are being used more and more these days because they offer better user experience, and therefore can attract more attention. For instance, different studies show that customers are more likely (almost twice as likely) to visit a business (e.g., restaurant, shop, apartment selling/renting) when an interactive virtual visit (virtual tour) is available [93, 37]. Moreover, the technology needed to provide and present these contents has been improved and become more affordable. According to Cisco Visual Networking Index, VR and AR Internet Protocol (IP) traffic will increase 12-fold between 2017 and 2022 globally [17].

An example of interactive media with 3 DOF is omnidirectional images and videos. This type of content is used in different applications such as geographical information systems like Google street view [3] (in which each capture point is an omnidirectional image), Cinematic VR [111], and robotics. An omnidirectional content can be modeled as a spherical signal because it provides a 360-degree view of the scene from a single point. When viewing these images, the users are placed at the center of the sphere, and they can freely change their viewing direction and observe the *portion* of the sphere facing them using a Head-Mounted Display (HMD).

3D models and environments take one step further than spherical contents by allowing the user to look around a VR environment more naturally with 6 DOF (consisting of three in translation and three in orientation) [41]. A 3D model must be represented in a structure suitable for rendering and display. The most common such structure is the 3D mesh [79, Chapter 14]. A 3D mesh object is usually described as a combination of several entities including geometrical information (a set of polygons in 3D) and some texture atlas/maps that are mapped to the 3D object surface at the rendering stage. With the recent advances in automated 3D reconstruction algorithms, high-quality 3D maps of locations and objects are widely accessible. In these virtual 3D environments the system simulates the physical presence of the users and a natural human vision using a virtual perspective camera (with a specific field of view) moving in the scene. This way, users can freely navigate within the scene and discover part of the 3D model they are interested in with great realism [41]. When looking at these 3D models, only *subpart* of the model and consequently a *portion* of the textural information is observed at any given moment.

To experience a real sense of immersion, such that the user feels he is in a virtual world that is a precise copy of reality, it is necessary to capture and store the data in high-resolution with large sizes

(*large database*). However, during the interaction, only a *small portion* of the data is requested by each user and this part is also in high quality by itself. For example, nowadays with recent HMDs, the spatial resolution of the viewport (i.e., the screen that shows the portion of the visual data the user observes) has at least 4K (4096 x 2160) resolution. This forces the whole spherical content to have a resolution of at least 12K [20]. This holds also for a texture map of a 3D model. For instance, the texture map of a historical site shown in [47] has a texture map of size 8192x8192 pixels.

The interactive feature that these media are providing has sparked new challenges in compressing them because it necessitates the distinction between *storage* and *transmission* rates. On the one hand, to compress these contents efficiently, the encoder must use all the existing dependencies between the data. This way the data can be stored in the most compressed state on the server. On the other hand, during delivery of the data, only a *subpart* of the compressed data is requested by the user, and this request is *random* from the encoder point of view. The randomness of the request is due to the fact that the request depends on the user's choice and the system is not aware of it during compression. Therefore, the database must be compressed and stored considering all dependencies, but the compression scheme must also have the ability to extract and send only the requested part in the most compressed state. However, efficient compression conflicts with the idea of random access to a portion of the data because the data is compressed considering the dependencies with also non-requested parts. More precisely, a subset extraction to reduce the transmission cost is usually incompatible with the joint compression aimed for reducing the storage size.

## Motivation

Available classical compression methods do not allow access to part of the compressed data (random access ability). However, they can be adapted to meet random access constraints. There are several possible ways to do this. One way is to send the entire compressed database to each user. Although this method is appropriate in terms of storage, it is certainly inefficient in terms of data transmission (delivery). The second method is to split the database into separate pieces and compress each part separately. Upon transmission, only the pieces that the user has requested are sent. This method reduces the transmission cost (because only pieces that are related to the user's request are transmitted), but it is not suitable for the storage as it does not consider the dependencies between pieces. Besides, the user may request only a small fraction of a piece, but the system still needs to send the whole piece. Thus, this method fails to achieve the transmission cost of an *oracle coder*, i.e., a hypothetical optimal coder that knows the requests in advance and encodes the requested parts only. In other words, this method is not perfectly optimal from the transmission point of view as well.

To achieve the ideal transmission rate, as the third method, one could encode and store all possible requests, but this exhaustive solution leads to a huge storage cost. As the fourth method, one can assume that the entire database is decoded at the server and again only the requested part is re-compressed and sent. Although this method is both optimal in terms of storage and transmission, it is practically infeasible, especially when the number of users is high, because it imposes an extensive computation on the server.

It can be seen that existing methods cannot achieve the optimal transmission rate unless they sacri-

fice the storage size with high overhead. Interestingly, it has recently been shown in [70] that by using incremental codes and with reasonable storage overhead, a dataset can be compressed jointly with random access ability such that upon user's request only the required portion of the data can be extracted from the compressed data and transmitted to the user. More precisely, using the incremental codes, we can expect no coding performance drop from the transmission rate perspective (compared to the oracle coder) with reasonable storage overhead.

As can be seen, there is a gap between what can be achieved theoretically [70] and what has been achieved already in practice. The goal of this thesis is to fill in the gap between theory and practice by proposing practical solutions for compression of interactive media. We construct an interactive coder that compresses the whole image but is able to extract and transmit a subpart of it in a compressed way. This interactive coder is generic in the sense that it can deal with various interactive media. This genericity is achieved by projecting the interactive media onto a two-dimensional (2D) image. In this thesis, we will consider two image modalities: 360° images and 3D models texture map. However, the proposed generic coder is also fitted to other interactive visualization scenarios (e.g., panorama, ultra-high-resolution images, city/country map). To achieve this goal, we need to answer four essential questions in this regard:

— *How to evaluate different interactive coding schemes for compression?* By moving away from classical communication methods towards interactive communication, the first point that draws attention is the need to distinguish between storage and transmission rates. This is not the case in classical methods in which all the stored data is sent to the user. Given the fact that data is usually lossily compressed (which introduces distortion), it arises the question of how to compare different methods that provide interactive communication from the compression perspective, while all three factors, i.e., storage, transmission rates, and distortions are considered together.

— *How to relate interactive compression of 2D visual contents with the theoretical results obtained in [70]?* The aim is to look for the connection between the information theoretical result and the practical compression task of 2D images with random access ability. For that, there is a need for a *problem definition* that allows modeling in the 2D image the equivalents for definitions such as source, side information (as a helper to compress the data), neighborhood and dependencies between sources, etc., that are defined in [70]. This leads to the definition of a generic coding strategy.

— *How to bring this generic coding structure into a practical interactive coder for the two imaging modalities considered in this dissertation (360° images and texture maps of 3D models)?* Each of these modalities has its own unique characteristics. Therefore, it is necessary to take into account the assumptions about user's request that are specific to each modality. More precisely, for each media, the neighborhood between sources that enables users navigating between them must be determined. Moreover, the 2D representation is only used in the community to have common representation among different interactive media. However, the underlying geometry of the data is not necessarily two-dimensional. For instance, the 360° contents lie on a spherical surface, or the texture information lie on the mesh of a 3D model. Therefore, it is necessary to also take into account the specific properties of the data geometry.

— *How to save rate with model selection?* The incremental coder proposed in [70] is based on *model-based coding*, i.e., the joint distribution between sources, called model, is needed at both encoder

and decoder for compressing and decoding the sources. Therefore, knowing the model is a key assumption. Since the nature of visual contents is *non-stationary*, sending the model description to the decoder is critical. However, sometimes sending the model is very costly and requires a lot of bits to describe them. Therefore, there is a need to estimate the relation between the number of bits we can save by sending a simpler approximate model and the rate that is needed to decode the source using this approximate model.

## Thesis roadmap

The dissertation is divided into 5 chapters. Following this introduction, Chapter 1 reviews the state of the art in compression for interactive communication. Specifically, with the help of introducing a navigation graph, we propose a common framework by which we can compare different approaches from the information-theoretic perspective. We derive achievable transmission and storage rates of each approach and compare them together. We then review existing solutions that are proposed to compress interactively the two 2D image modalities tackled in this work: 360° contents and texture map of 3D models. For that, first, we explain how these contents are presented with 2D planes, and then we explain how the interactivity compression is provided for these modalities.

Next, in the following chapters we present our different contributions which answer the four questions raised in previous section (Section Motivation).

— Chapter 2: This chapter answers the first question. Despite many contributions that have been proposed so far for interactive compression of 2D images, not much has been done to evaluate different interactive approaches. Here we propose an evaluation framework that considers storage, transmission rate, and distortion at the user's side altogether. This brings the ability to compare different methods based on the storage capacity on the server and network bandwidth of users.

— Chapter 3: In this chapter, we tackle the second question of the previous section (Section Motivation). We propose a new generic interactive compression scheme for 2D images that achieves the oracle transmission cost while keeping the storage cost reasonable. All conventional 2D image compression schemes work block-wise. To have efficient compression, it is necessary to predict each block based on the blocks already decoded. Classical schemes use a fixed pre-defined scanning order for encoding and decoding of the blocks, which contradicts the idea of having random access ability to part of data. Therefore, these schemes impose to send more blocks than what is needed. In contrast, we propose a solution to encode the blocks using incremental codes such that any decoding order is possible at the decoder's side. For that, for each block to be encoded, several predictions are generated corresponding to different decoding order of blocks. Then, a bitstream is generated for each block based on incremental entropy codes. With this bitstream, we are able to decode the block for the worst prediction. For better quality predictions, a substream can be extracted from the same bitstream to perform the decoding operation. Therefore, only the information which is needed is transmitted once the user's request is received.

— Chapter 4: This chapter finds answers to the third question raised in (Section Motivation). Based on the generic coder proposed in Chapter 3, we design compression tools to encode two imaging

modalities that provide interactivity by their nature: 360° images and texture maps of 3D models. Each of these imaging modalities has its own characteristic that needs to be studied separately. For each, we propose a strategy that allows *starting* the decoding procedure whatever the user's request is. The requested sources are decoded one after the other with a scanning order that is optimized to reduce the transmission rate. To capture the redundancies within an image, we propose novel prediction modes that can be adapted to any decoding order and consequently any user's request. Moreover, we also proposed new representations for each of these modalities that help to compress the images more efficiently.

— Chapter 5: The proposed interactive coder relies on correlation models to compress the database. The more accurate the model is, the better compression can be achieved for the data. However, since this model has to be sent to the user for decoding, the use of a more accurate model requires a large number of bits to describe the model itself which increases the storage and transmission rates. An alternative way to solve this problem is to approximate the correlation model with a simpler parametric model and transfer it instead. Nevertheless, using an incorrect model increases the compression rate. Therefore, when we describe a model with an approximate model, the number of bits we save in the model description must be greater than the number of bits we lose due to the data compression using this approximate model. The goal of this chapter is to compute this excess rate for interactive compression. With the tools provided in this chapter, we can select the model that requires lower rates while both the compressed data size and the model description size are taken into account simultaneously.

As a remark, we focus on compression of static scene images without considering the temporal aspects. In fact, this step is the first key step in video compression schemes because it allows resynchronization. Nevertheless, inspired by the tools provided in this dissertation, the proposed generic coder can be extended to cover the temporal aspect as well.

Finally, we conclude this dissertation with a summary and provide an outlook on possible directions for future work. The list of publications produced by the author can be found in the section of author's publications.

CHAPTER 1

# STATE OF THE ART

In interactive communication, users can make requests to a database, and each request concerns only a subpart of the database. These requests are not known in advance by the server, and are classically modeled as random. A desirable system should be able to extract the relevant part from the compressed stream on the server. Theoretical results show that this is achievable using a coding tool called incremental codes. However, most of the existing practical solutions rely on classical compression schemes which are not designed for this purpose. To enable random access ability these coding schemes are adapted to the problem, and this results in sub-optimality of the compression scheme.

In this chapter, we first introduce the mathematical set-up and notations we have used throughout the dissertation in Section 1.1. We then review the related works in interactive communication from the information-theoretic perspective in Section 1.2. In Section 1.3, we study existing practical solutions for interactive compression of two modalities which are interactive media by their nature, i.e., 360° contents and texture map of 3D models.

## 1.1 Notations and Definitions

A random scalar source/variable $X$ is denoted by uppercase and consequently, we define the following notations:

— Lowercase italic letter $x$ denotes the realization of $X$.
— Calligraphic letter $\mathcal{X}$ represents the alphabet of random variable $X$.
— The cardinality of the set $\mathcal{X}$ is denoted by $|\mathcal{X}|$ .
— Uppercase bold letter $\boldsymbol{X}^n = (X_1, ..., X_n)$ denotes a random sequence of length $n$.
— Lowercase bold italic letter $\boldsymbol{x}^n$ is used to show the realization of random sequence $\boldsymbol{X}^n$.
— $P_X(x)$ denotes the Probability Mass Function (PMF) of the *discrete* random variable $X$.
— The cross-entropy of PMFs $P_X(x)$ and $Q_X(x)$ is denoted by

$$H\big(P_X(x), Q_X(x)\big) \triangleq -\sum_{x\in\mathcal{X}} P_X(x)\log_2 Q_X(x).$$

— The entropy of $P_X(x)$ is denoted by $H(X) \triangleq -H\big(P_X(x), P_X(x)\big)$.
— The Kullback–Leibler (KL) divergence between PMFs $P_X(x)$ and $Q_X(x)$ is denoted by

$$D_{KL}(P||Q) \triangleq H\big(P_X(x), Q_X(x)\big) - H(X).$$

We denote the joint probability distribution between discrete sources $X$ and $Y$ by $P_{X,Y}(x, y)$. The

two sources are said to be independent if and only if $P_{X,Y}(x,y) = P_X(x)P_Y(y)$, otherwise they are dependent. As a consequence we have:

— The conditional PMF of the discrete source $X$ given $Y$ is denoted by $P_{X|Y}(x|y)$.

— The cross-entropy of conditional PMFs $P_{X|Y}(x|y)$ and $Q_{X|Y}(x|y)$ given $Y = y$ is denoted by

$$H(P_{X|Y}, Q_{X|Y}|y) \triangleq -\sum_{x\in\mathcal{X}} P_{X|Y}(x|y)\log_2 Q_{X|Y}(x|y).$$

— The conditional entropy of $X$ given $Y$ is written as

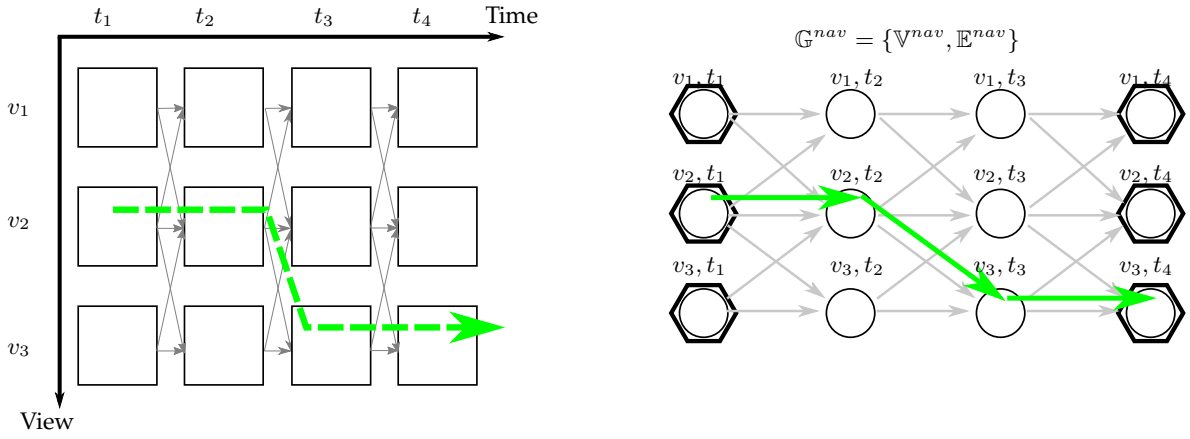
$$H(X|Y) \triangleq \sum_{y\in\mathcal{Y}} P_Y(y)H(P_{X|Y}, P_{X|Y}|y).$$

We denote a deterministic vector also by lowercase bold letters. Sets that are not alphabet of random variables are denoted either by blackboard bold letter typeface like $\mathbb{G}$ or within brackets $\{\}$. For example $\mathbb{N}$ is the set of natural numbers, i.e., $\{1, 2, ...\}$. An ordered occurrence of events (sequence) is written with round parentheses $()$.

## 1.2 Review of the compression schemes for interactive communication

Consider a *large database* of several sources in which there are many redundancies among the sources. This database must be compressed efficiently before being *stored* on a server. Therefore, the dependencies between the sources have to be exploited to capture the redundancies. Now consider the situation that a *large number* of users are navigating through the sources and each user requests a *small subset* of the data (this subset differs among users). To have an efficient *transmission*, the server should avoid transmitting all the stored bitstreams, and it should be able to *extract* and transmit only the requested part to the user. The user's request is modeled as a *random access* process from the server's perspective because the choice of the subset depends on the user only. However, for each source the server knows the *possible set* of sources from which the user can navigate from. Moreover, we assume that the requests of each user are sequential meaning that he requests sources one after the other and the previous requests are kept in memory to help the decoder and therefore improve the compression efficiency. The compression of this paradigm is often called interactive coding or more precisely Sequential Massive Random Access (SMRA) [29, 28, 30].

The applications of such a sequential model are widespread nowadays, a well known of which is free-viewpoint television (FTV) [90] where users can navigate (with some constraints) within views of a multiview video dataset to control the viewpoint. A common scene such as a sporting arena is captured using a large number of cameras placed around the scene (*large database*). Since the cameras are capturing the same scene, there are many redundancies among views that must be exploited to have efficient compression. Besides, as is the case in broadcasting a soccer match, a *large number* of users may view the scene. Each user requests one view at a time (*a subset of the dataset*) and the choice of the view is *random* from the server perspective because it depends on the user and differs between different users.




(a) (b)

Figure 1.1 – Navigation graph of a multiview video for FTV system where a user can navigate within the views. (a) The thin lines with arrows (in gray) show the allowed navigation transitions between the views. The thick dashed line (in green) shows an example of user navigation. (b) The corresponding navigation graph. Circles (graph nodes) represent views at different times as sources and gray arrows show directed edges of the graph. Access sources are shown with circles inside hexagons. A user's navigation is shown with green arrows.

However, usually the system restricts the navigation of the user by imposing some *navigation constraints* to propose smooth navigations to the user. For instance, the system shown in Fig. 1.1a does not allow the user to jump directly from *view 1* to *view 3*, and it forces the user to pass the middle view (*view 2*) before watching *view 3*. Note that this makes it possible for the system to know what the *possible previous requests* of the user are before requesting a given view.

Interactive compression has been done in many different ways. Some approaches partition the database into separate pieces and encode each one separately. Upon request of a user, the pieces that the user has requested are transmitted [53, 62, 113, 68, 39]. Some others, encode and store all possible requests, at the cost of increasing the storage size, so that the system can extract the exact sources that users are interested in [57, 44, 20, 5]. To review all these works, with the help of terminologies defined in [29], we propose a common framework to compare different approaches fairly. For that, we first formally define the interactive coding problem and then using this formal definition, we theoretically analyze the achievable performance of each interactive coding scheme. Finally, we study different interactive image modalities and review the existing coding schemes applied in these modalities.

### 1.2.1 Problem formulation

Let $\{X_{(l)}\}_{1\leq l\leq L}$ be a database of a set of $L$ sources with joint distribution $P(X_{(1)}, ..., X_{(L)})$. Assume that a user is able to navigate in the database by sequentially asking (one source after the other) $n$ realizations of $K$ sources ($K \leq L$). As proposed in [29], the user's navigation is constrained by a structure defined between sources $\{X_{(l)}\}_{1\leq l\leq L}$ and this structure depends on the application (like FTV transition constraints shown in Fig. 1.1). This structure is known at the encoder and can be described by a graph

$\mathbb{G}^{nav} = \{\mathbb{V}^{nav}, \mathbb{E}^{nav}\}$, in which the set of $L$ vertices $\mathbb{V}^{nav}$ represents sources $\{X_{(l)}\}_{1 \leq l \leq L}$ and $\mathbb{E}^{nav}$ is the set of edges between vertices. The edges can be directed or undirected. If it is directed, there is an edge from vertex $l$ to vertex $l'$ if the user is allowed to request source $X_{(l')}$ right after requesting source $X_{(l)}$. An undirected edge means that a user can navigate in both directions. An example of the data structure graph with undirected edges is shown in Fig. 1.2a. The corresponding graph with directed edges for the FTV example explained above is depicted in Fig. 1.1b.

The user can only initiate a request from a specific set of sources. These sources are called *access sources* and their indices are available in the set $\mathbb{A}$. In order words, a sequence of requests is valid if the index of the first requested source is in $\mathbb{A}$. For subsequent requests, only the requests that respect the defined neighborhood for each source in the navigation graph $\mathbb{G}^{nav}$ are accepted. Given the structure $\mathbb{G}^{nav}$, if a user requests a source $X_{(k)}$, the encoder knows the set of indices of *possible* neighboring sources $\mathbb{J}_{(k)}$ from which the user can request $X_{(k)}$. In other words, as shown in Fig. 1.2b, for each source $X_{(k)}$ the encoder is sure that the previous user's request can not be outside of this set.

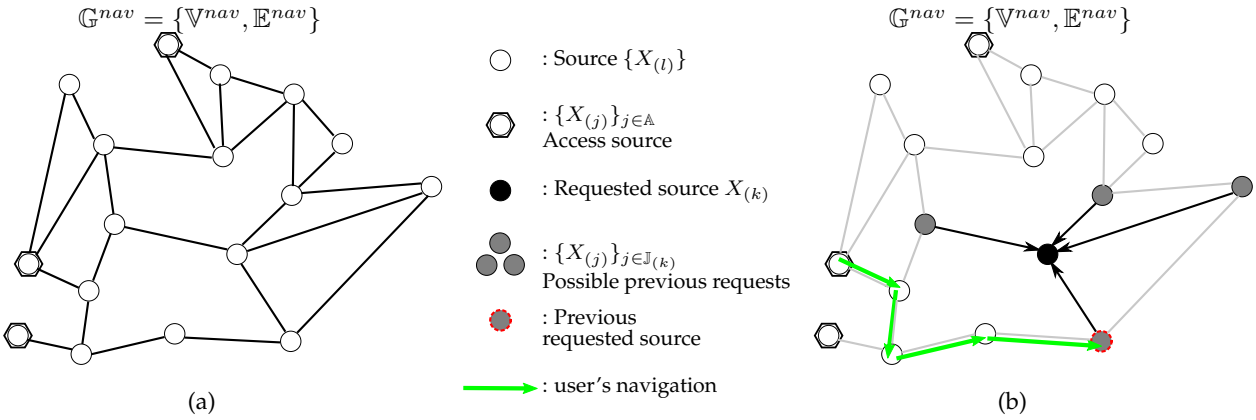



Figure 1.2 – The navigation graph of an interactive system. (a) Navigation within the database is modeled by a graph. (b) An example of the user navigation in the graph is shown in green. For a given node (black node), the possible previous requests are depicted by grey nodes.

#### 1.2.1.1 Related problems: interactive compression

It is worth noting that this definition of interactive coding is different from the one proposed by El Gamal and Orlitsky [31]. They define interactive compression as follows: given two sources $X$ and $Y$ at two sides of a two-way channel. Suppose that the two terminals wish to communicate over the channel such that at the end of the communication process, they both know $X$ and $Y$. They showed that at least $H(X|Y) + H(Y|X)$ bits must be exchanged on average and $H(X,Y) + 2$ bits are sufficient. In other words, in [31] the scheme is completely symmetric and the two sides perform compression and decompression. Instead, in this thesis, interactive compression is asymmetric and one side performs compression (the sender) whereas the other part (the user) performs decompression by using part of the compressed stream provided by the other side (server).

#### 1.2.1.2 Definitions: achievable rates and coding/decoding functions

The critical aspect of interactive coding is to study the storage size and the transmission rate together. The storage size is the size of the entire database stored on the server, and the transmission rate refers to the size of the subpart of the database that is sent from the server to the decoder. These two terms are denoted by $S$ and $R$, respectively. The main challenge in interactive compression is that a user requests a small part of the entire database, but the encoder is not aware of the requests during compression. Therefore, the request can be modeled as random. As a result, in this system, the entire database must be compressed and stored in such a way that the system can be adapted to the user's request during data transmission. More precisely, the system should be able to extract the requested parts from the compressed stream to avoid sending all the stored bitstream to the user. It is worth noting that due to the large number of users, this extraction operation must be very simple. In particular, re-encoding the data is not acceptable.

The overall scheme of interactive compression is shown in Fig. 1.3 and includes the following steps:

1. First, the **offline encoding** is performed. All proposed schemes in the literature can be efficiently explained by introducing the notion of encoding graph. This encoding graph differs from the navigation graph and describes the SI sources used to encode each source while allowing the navigation possibilities explained in the navigation graph. Let

$$\mathbb{G}^{enc} = \{\mathbb{V}^{enc}, \mathbb{E}^{enc}\}$$

   where $\mathbb{V}^{enc}$ represents the sources $\{X_{(l)}\}_{1\leq l\leq L}$ as in $\mathbb{G}^{nav}$, i.e. $\mathbb{V}^{enc} = \mathbb{V}^{nav}$. There is an edge from vertex $i$ to vertex $j$ in $\mathbb{E}^{enc}$, if source $X_{(i)}$ is encoded given source $X_{(j)}$ as Side Information (SI). The edge weight is equal to the cost of encoding $X_{(i)}$ given $X_{(j)}$.

   Finally, given this encoding graph, the offline encoder compresses the sources while taking into account their connections depicted in the encoding graph. More specifically, for each source $X_{(k)}, 1 \leq k \leq L$, there exists a function $f_{(k)}$ to compress length-n random sequences $\boldsymbol{X}^n_{(k)}$, generated from source $X_{(k)}$. **Encoding function** $f_{(k)}$ assigns a sequence of $|\mathbb{J}_{(k)}|$ indices to each possible set of realization vectors $\left(\boldsymbol{x}^n_{(k)}, (\boldsymbol{x}^n_{(j)})_{j\in\mathbb{J}_{(k)}}\right) \in \mathcal{X}^n \times \mathcal{X}^{n\cdot|\mathbb{J}_{(k)}|}$:

$$\begin{aligned} f_{(k)} : \mathcal{X}^n \times \mathcal{X}^{n\cdot|\mathbb{J}_{(k)}|} &\to \prod_{i=1}^{|\mathbb{J}_{(k)}|} \{1, .., 2^{n\cdot r_{k,i}}\} \\ \boldsymbol{x}^n_{(k)}, \left(\boldsymbol{x}^n_{(1)}, ..., \boldsymbol{x}^n_{(j)}, ..., \boldsymbol{x}^n_{(|\mathbb{J}_{(k)}|)}\right)_{j\in\mathbb{J}_{(k)}} &\mapsto (\boldsymbol{b}_{k,1}, ..., \boldsymbol{b}_{k,j}, ..., \boldsymbol{b}_{k,\mathbb{J}_{(k)}}) \end{aligned} \tag{1.1}$$

   where the set of rates $\{1, .., 2^{n\cdot r_{k,m}}\}$ is introduced in order to allow extraction. Each $\boldsymbol{b}_{k,j}$ is a sequence of compressed bitstreams. The storage size per source $k$ is $S_{(k)} = |\boldsymbol{b}_{k,1}| + |\boldsymbol{b}_{k,2}| + ... + |\boldsymbol{b}_{k,\mathbb{J}_{(k)}}|$.

2. Upon request of the source $X_{(k)}$, the server extracts part of the stored bitstream according to the user's previous requests and sends it to the user. More specifically, consider the user is requesting source with index $k$, and the index of the actual previous source in the memory at the decoder is $j^*$ ($j^* \in \mathbb{J}_{(k)}$). The **online extractor** $g_{(k)|(j^*)}$ extracts a subsequence of bitstreams from

$(\boldsymbol{b}_{k,1}, ..., \boldsymbol{b}_{k,j}, ..., \boldsymbol{b}_{k,\mathbb{J}_{(k)}})$:

$$g_{(k)|(j^*)} : \prod_{i=1}^{|\mathbb{J}_{(k)}|} \{1, .., 2^{n \cdot r_{k,i}}\} \rightarrow \prod_{m \in \mathbb{I}_{(k)|(j^*)}} \{1, .., 2^{n \cdot r_{k,m}}\}$$
$$(\boldsymbol{b}_{k,1}, ..., \boldsymbol{b}_{k,j}, ..., \boldsymbol{b}_{k,\mathbb{J}_{(k)}}) \mapsto (\boldsymbol{b}_{k,i_1}, ..., \boldsymbol{b}_{k,i_j}, ..., \boldsymbol{b}_{k,i_{|\mathbb{I}_{(k)|(j^*)}|}})_{i_j \in \mathbb{I}_{(k)|(j^*)}} \tag{1.2}$$

where $\mathbb{I}_{(k)|(j^*)} \subseteq \{1, ..., |\mathbb{J}_{(k)}|\}$ and $R_{(k)|(j^*)} = \sum_{m \in \mathbb{I}_{(k)|(j^*)}} |\boldsymbol{b}_{k,m}| \leq S_{(k)}$.

3. Assuming that the previous user's request $j^* \in \mathbb{J}_{(k)}$ with realization $\boldsymbol{x}^n_{(j^*)}$ is available in the memory (at the decoder), the **decoding function** $h_{(k)|(j^*)}$ recovers an estimate $\hat{\boldsymbol{x}}^n_{(k)}$ by $\boldsymbol{x}^n_{(k)}$. More specifically,

$$h_{(k)|(j^*)} : \prod_{m \in \mathbb{I}_{(k)|(j^*)}} \{1, .., 2^{n \cdot r_{k,m}}\} \times \mathcal{X}^n \rightarrow \mathcal{X}^n$$
$$(\boldsymbol{b}_{k,i_1}, ..., \boldsymbol{b}_{k,i_j}, ..., \boldsymbol{b}_{k,i_{|\mathbb{I}_{(k)|(j^*)}|}})_{i_j \in \mathbb{I}_{(k)|(j^*)}}, \boldsymbol{x}^n_{(j^*)} \mapsto \hat{\boldsymbol{x}}^n_{(k)} \tag{1.3}$$

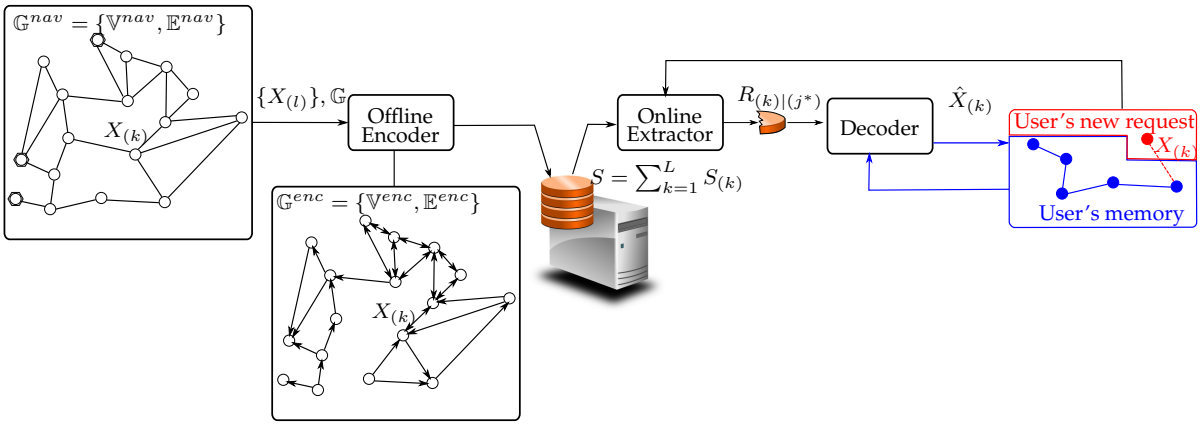



Figure 1.3 – Overall scheme of an interactive coder. The database is stored with rate $S$ on the server. The offline extractor builds an encoding graph $\mathbb{G}^{enc}$ that can differ from the navigation graph $\mathbb{G}^{nav}$, and uses this graph to encode the database. Each time a user requests a new source, the online extractor extract part of stored bitstream considering the sources available in user's memory. For the same navigation requests depicted in Fig. 1.2b, the user's previous requests are shown in blue and the current request is shown in red.

A summary of the notations and definitions related to problem formulation is given in Table 1.1

### 1.2.2 Information theoretical bounds

Here we study the storage and transmission cost for different coding schemes that propose interactivity. To ease the comparison, we assume that lossless compression is performed in each scheme. Note that this comparison gives also insights to compare practical lossy compression, since lossy compression is usually achieved by quantizing the transformed values of the signal. The quantized values are then compressed using a lossless entropy coder.

Table 1.1 – Notations and definitions used for problem formulation of compression for interactive communication

| | |
|---|---|
| $L$ | Number of sources in the database |
| $X_{(l)}$ | Source $l$ in the database ($1 \leq l \leq L$) |
| $\mathbb{J}_{(k)}$ | Neighborhood of source $k$ which indicates the set of possible previously requested source indices |
| $\mathbb{A}$ | Set of indices of access sources |
| $\mathbb{G}^{nav}$ | Navigation graph $\mathbb{G}^{nav} = \{\mathbb{V}^{nav}, \mathbb{E}^{nav}\}$<br>($\mathbb{V}^{nav}$: Nodes of the navigation graph, $\mathbb{E}^{nav}$: Edges of the navigation graph) |
| $\mathbb{G}^{enc}$ | Encoding graph $\mathbb{G}^{enc} = \{\mathbb{V}^{enc}, \mathbb{E}^{enc}\}$<br>($\mathbb{V}^{enc}$: Nodes of the encoding graph, $\mathbb{E}^{enc}$: Edges of the encoding graph) |
| $S_{(k)}$ | Storage rate cost for coding the source of index $k$ |
| $j^*$ | Index of the source actually requested by the client at previous instant before requesting source $k$<br>($j^* \in \mathbb{J}_{(k)}$) |
| $R_{(k)\|(j^*)}$ | Transmission rate cost for coding the source of index $k$ with source $j^*$ in memory |

#### 1.2.2.1 Interactive coding based on classical predictive coding

Interactive compression can be achieved with classical *predictive coding* scheme. Predictive coding is based on the principle of conditional coding illustrated in Fig. 1.4. Given two i.i.d. sources $X$ and $Y$, in this scheme source $X$ is compressed losslessly while source $Y$ is available at both encoder and decoder as SI source to help the compression. The achievable lossless compression rate is the conditional entropy $H(X|Y)$ which is smaller that the rate $H(X)$ when no SI is available [21, Theorem 5.4.1], [82]. This approach is efficient, but it imposes a fixed pre-defined encoding and decoding order, which contradicts the idea of random access to part of the database. To achieve interactivity, two approaches are used to adapt predictive coding scheme to interactive coding: tile-based approach and exhaustive storage.

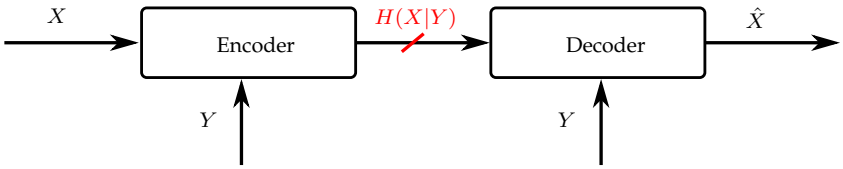



Figure 1.4 – Lossless predictive coding [21] when SI $Y$ is available at both encoder and decoder. The source $X$ is compressed at rate $H(X|Y)$, which is smaller than its entropy $H(X)$.

**a) Predictive coding for interactive compression based on tile-based approach**

In this scheme the database is partitioned into separate pieces. Each piece is encoded separately from the others, but all the sources in one piece are coded all together with the predictive coding scheme. Therefore, there exists one compressed bitstream per piece. We can see this partitioning under the scope of a graph as follows. First, the encoding graph is built from the navigation graph by removing edges to obtain separated subgraphs. Then, the sources in each subgraph are compressed. The used compression algorithm defines the edges of the subgraph, which may differ from the navigation graph. Then, upon user's request, any piece (subgraph) which contains at least one requested source, must be transmitted. An example of such an encoding graph for a navigation graph depicted in Fig. 1.2 is shown in Fig. 1.5a. As shown in Fig. 1.5b, due to the imposed dependencies between sources in the compression of each

subgraph, more sources are being transmitted than needed.

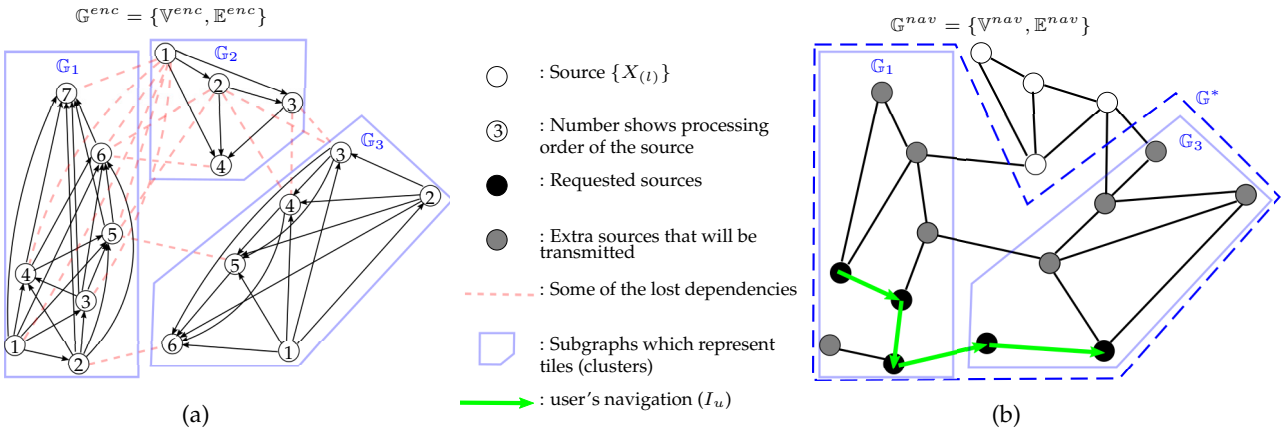



Figure 1.5 – Encoding graph for the tile-based approach. (a) An example which shows the partitioning of the encoding graph for the same navigation graph shown in Fig. 1.2. Numbers in each node show the encoding/decoding order within each subgraph. (b) A sample of user navigation and subgraphs that will be transmitted. The elements of graph $\mathbb{G}^*$ is also shown in blue dashed line.

More formally, the steps of the scheme are as follows:

1. The offline encoder converts the navigation graph $\mathbb{G}^{nav}$ into an encoding graph that is a set of separated subgraphs

$$\mathbb{G}^{enc} = \mathbb{G}^{Tile} = \{\mathbb{G}_1, \mathbb{G}_2, ..., \mathbb{G}_{|\mathbb{G}^{Tile}|}\}, \text{where} \begin{cases} \mathbb{G}_c = \{\mathbb{V}_c, \mathbb{E}_c\} \\ \bigcup_{c=1}^{|\mathbb{G}^{Tile}|} \mathbb{V}_c = \mathbb{V}^{enc}, \quad \mathbb{V}^{enc} = \mathbb{V}^{nav} \\ \mathbb{G}_i \cap \mathbb{G}_j = \emptyset \quad \forall i, j \in [1, ..|\mathbb{G}^{Tile}|], i \neq j \end{cases}$$

where the combination of edges within each subgraph varies between different encoders. An example of such a subgraph partitioning is shown in Fig. 1.5a. Here, to ease the comparison, we assume that the encoder is optimal within a tile (subgraph) such that all the previously compressed sources are used in the compression of the current source, i.e., the source with index $c_v (X_{c_v} \in \mathbb{V}_c)$ is encoded given $c_1, c_2, ..., c_{v-1}$. However, in practice a Markov chain of order one or two is assumed for the dependencies between sources, so only one or two of the previously coded sources are used for compression. The sources within the same tile are jointly compressed and this leads to a non separable compressed bitstream. The cost of compressing the sources, whose nodes are in the same subgraph is:

$$cost^{Tile}(\mathbb{G}_c) = \sum_{v=1}^{|\mathbb{V}_c|} H(X_{c_v}|X_{c_0}, X_{c_1}, ..., X_{c_{v-1}}), \quad \text{where } X_{c_0} = \emptyset \tag{1.4a}$$

$$= H(X_{c_1}, X_{c_2}, ..., X_{c_v}) = H(\mathbb{V}_c) \tag{1.4b}$$

where (1.4b) is the result of the chain rule property of conditional entropies [21, Theorem 2.2.1] and $H(\mathbb{V}_c)$ represents the joint entropy of sources within each subgraph. The storage cost per source of storing all compressed subgraphs is

$$S^{Tile} = \frac{1}{L}\sum_{c=1}^{|\mathbb{G}^{Tile}|} cost^{Tile}(\mathbb{G}_c) = \frac{1}{L}\sum_{c=1}^{|\mathbb{G}^{Tile}|} H(\mathbb{V}_c) \tag{1.5a}$$

$$\geq H(\mathbb{V}^{enc}). \tag{1.5b}$$

where $L = |\mathbb{V}^{enc}|$ is the number of all the sources in the data base and $H(\mathbb{V}^{enc})$ is the joint entropy which represents the cost of encoding all sources all together. The relation (1.5b) comes from the fact that some inter-graph dependencies are not exploited (red edges in Fig. 1.5a).

2. In this scheme, the access to the sources is at the tile level rather than the source level. Therefore, the extractor identifies a set of tiles such that this set of tiles contains at least all requested sources. Finally, it transmits the compressed streams of the corresponding tiles to the user. Consider user $u$ is sequentially requesting a sequence of sources

$$\boldsymbol{I}_u = (i_1, i_2, ..., i_{|\boldsymbol{I}_u|}), \text{where} \begin{cases} X_{i_t} \in \mathbb{V}^{enc} \\ i_{(t-1)} \in \mathbb{J}_{(i_t)} \\ |\boldsymbol{I}_u| \ll |\mathbb{V}^{enc}| \end{cases} \tag{1.6}$$

where $i_t$ is the index of the source requested by the user at time instance $t$. An example of such a sequential request is depicted with green arrows in Fig. 1.5b. The transmission cost per requested source is

$$R_u^{Tile} = \frac{1}{|\boldsymbol{I}_u|}\sum_{c\in\mathbb{C}_u} cost^{Tile}(\mathbb{G}_c) = \frac{1}{|\boldsymbol{I}_u|}\sum_{c\in\mathbb{C}_u} H(\mathbb{V}_c) \tag{1.7a}$$

$$\geq \frac{1}{|\boldsymbol{I}_u|} cost(\mathbb{G}^*) \tag{1.7b}$$

$$= H(\mathbb{V}^*) \tag{1.7c}$$

where $\mathbb{C}_u$ is the *smallest* set of subgraph indices such that all requested sources are in the union of subgraphs, i.e., $\mathbb{V}^* = \bigcup_{c\in\mathbb{C}_u} \mathbb{V}_c$ and $\boldsymbol{I}_u \subseteq \mathbb{V}^*$. For the example shown in Fig. 1.5b, $\mathbb{C}_u = \{1, 3\}$. The graph $\mathbb{G}^* = \{\mathbb{V}^*, \mathbb{E}^*\}$ in (1.7b) is a graph that contains all graphs in $\mathbb{C}_u$ and it considers also the missed dependencies between subgraphs. (1.7b) holds because $G^*$ considers the inter-graph missed dependencies in $\mathbb{C}_u$.

3. The decoder receives the subgraphs $\mathbb{C}_u$ and decode the sources in the subgraphs. Each subgraph is decoded separately. Finally, an estimate $\hat{X}_{(k)}, k \in \boldsymbol{I}_u$ is assigned to each recovered source requested.

**b) Predictive coding for interactive compression based on exhaustive storage approach**

Another approach to use predictive coding scheme in interactive compression is to encode each source with respect to all possible navigations that user can request. With this exhaustive solution, we

can achieve the transmission rate of an *oracle coder*: a hypothetical coder which knows the request in advance and encodes the requested sources only. However, this scheme results in an enormous storage overhead. More formally, the steps of the exhaustive storage coding scheme are as follows:

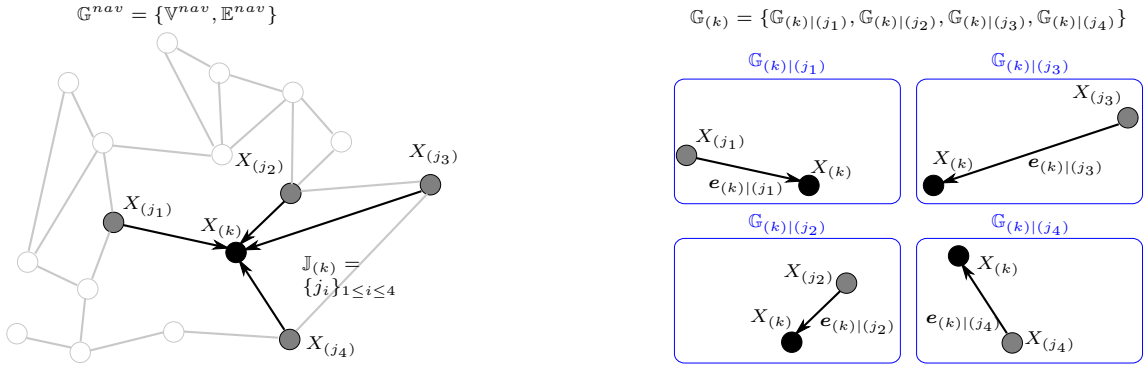



Figure 1.6 – Encoding graph for the exhaustive storage approach. (a) The neighborhood sources (in gray) of a given source $X_{(k)}$ (in black) is a set of sources which their indices are in $\mathbb{J}_{(k)}$. (b) The encoding graph related to source $X_{(k)}$. This graph is composed of several graphs where each consists of two vertices and one edge.

1. The encoding graph is defined *per source* and it contains $|\mathbb{J}_{(k)}|$ subgraphs. In other words, for each source $X_{(k)}$ a subgraph per element of the set $\mathbb{J}_{(k)}$ is generated. For each $l \in \mathbb{J}_{(k)}$, the corresponding subgraph has two vertices $X_{(k)}, X_{(l)}$ and one edge from $X_{(l)}$ to $X_{(k)}$. An example of such a graph for a source $X_{(k)}$ is shown in Fig. 1.6. Let

$$\mathbb{G}^{enc}_{(k)} = \mathbb{G}^{Ex}_{(k)} = \{\mathbb{G}_{(k)|(1)}, ..., \mathbb{G}_{(k)|(|\mathbb{J}_{(k)}|)}\}, \text{where} \begin{cases} \mathbb{G}_{(k)|(j)} = \{\mathbb{V}_{(k)|(j)}, \boldsymbol{e}_{(k)|(j)}\} \\ \mathbb{V}_{(k)|(j)} = \{X_{(l)}, X_{(k)}\}, l \in \mathbb{J}_{(k)} \\ \boldsymbol{e}_{(k)|(j)} \in \mathbb{E}^{nav} \end{cases} \tag{1.8}$$

be the encoding graph related to the source $X_{(k)}$ which consists of several sub graphs. Each subgraph $\mathbb{G}_{(k)|(j)}$ contains two vertices with an edge $\boldsymbol{e}_{(k)|(j)}$ between them such that the union of all edges is equal to the edges of the navigation graph:

$$\bigcup_{k=1}^{L} \bigcup_{j=1}^{|\mathbb{J}_{(k)}|} \boldsymbol{e}_{(k)|(j)} = \mathbb{E}^{nav}$$

where $L = |\mathbb{V}^{nav}|$ is the total number of sources in the database. Then exhaustive storage scheme encodes each subgraph of each source independently. The cost of compressing each subgraph is

$$cost^{Ex}\big(\mathbb{G}_{(k)|(j)}\big) = H(X_{(k)}|X_{(j)}), \text{where } j \in \mathbb{J}_{(k)}. \tag{1.9}$$

The average storage cost per source is

$$S^{Ex} = \frac{1}{L}\sum_{k=1}^{L}\sum_{j=1}^{|\mathbb{J}_{(k)}|} cost^{Ex}\big(\mathbb{G}_{(k)|(j)}\big) = \frac{1}{L}\sum_{k=1}^{L}\sum_{j=1}^{|\mathbb{J}_{(k)}|} H(X_{(k)}|X_{(j)}) \tag{1.10a}$$

$$\gg H(\mathbb{V}^{enc}). \tag{1.10b}$$

Comparing (1.5) with (1.10) one can conclude that the storage overhead of exhaustive storage is much larger than tile-based approach.

2. Considering what the user has in the memory from the previous request, for each navigation request, the online extractor picks the compressed stream of the corresponding subgraph and transmits it to the user. Considering the same sequential request of user $u$ in (1.6), the transmission cost per requested source is

$$R_u^{Ex} = \frac{1}{|\boldsymbol{I}_u|}\sum_{t=0}^{|\boldsymbol{I}_u|} cost^{Ex}\big(\mathbb{G}_{(i_t)|(i_{(t-1)})}\big) = \frac{1}{|\boldsymbol{I}_u|}\sum_{t=0}^{|\boldsymbol{I}_u|} H(X_{(i_t)}|X_{(i_{(t-1)})}), \quad \text{where } X_{i_0} = \emptyset \tag{1.11a}$$

$$\approx H(X_{(i_1)}, X_{(i_2)}, ..., X_{(i_{|\boldsymbol{I}_u|})}) \tag{1.11b}$$

$$= R^{oracle} \tag{1.11c}$$

where (1.11b) is the joint entropy of all requested sources and the approximation holds because it is usually assumed the sources are Markov processes of order one, i.e.,

$$H(X_{(i_t)}|X_{(i_1)}, X_{(i_2)}, ..., X_{(i_{(t-1)})}) = H(X_{(i_t)}|X_{(i_{(t-1)})}). \tag{1.12}$$

Therefore, with the exhaustive storage approach we can achieve the oracle transmission rate.

3. Finally, the decoding function (1.3) decodes the requested source using the received bitstream and what it has in the memory.

#### 1.2.2.2 Interactive coding using model-based coding schemes

To have a flexible encoding/decoding for interactive communication, an *encoding graph* same as the *navigation graph* is desired (see Fig. 1.3). More precisely, the source has to be compressed once in a single encoding process, but it is decoded in several ways depending on which SI (from the neighborhood) is available at the decoder. The compression in this case, i.e. when the encoding graph is equal to navigation graph, is an instance of *model-based coding* [29, 30]. In this scheme, the SI is not available at the encoder and only the statistical model between the source and its SI is used to determine the compression rate and encode the source. However, the SI is available at the decoder and the decoder uses both the model and the SI to recover the source from the compressed stream.

The first step to model-based coding is provided by Slepian and Wolf [82]. In their seminal work, they show that both the predictive coding and the model-based coding approaches can achieve the same compression rate even though they have completely different setups. More precisely, consider two

correlated source sequences $\boldsymbol{x}^n$ and $\boldsymbol{y}^n$ generated from the sources $X$ and $Y$, respectively. To compress $\boldsymbol{x}^n$, the predictive encoder explicitly uses the realization $\boldsymbol{y}^n$ of the other source, while the Slepian-Wolf encoder does not have access to $\boldsymbol{y}^n$ and it uses only the joint distribution (called model) between $\boldsymbol{X}^n$ and $\boldsymbol{Y}^n$ for compression. However, Slepian and Wolf showed that both approaches can achieve the compression rate of $H(X|Y)$.

Later, Sgarro generalized the Slepian and Wolf results to the problem of source coding with multiple decoders, where different correlated sources are available at the decoders as SI [77]. The sources are considered to be discrete, memoryless, and stationary. Fig. 1.7 shows the coding scheme proposed in [77] where source $X_{(k)}$ is transmitted to $|\mathbb{J}_{(k)}|$ decoders with different side information $X_{(j)}, j \in \mathbb{J}_{(k)}$. The source is compressed considering all SI sources and then the *same* compressed bit stream is transmitted to each decoder. Let us denote the storage rate per source symbol for encoding source $X_{(k)}$ by $S_{(k)}$. Sgarro showed that in this case the source can be compressed with respect to the least correlated source:

$$S_{(k)} = \max_{j \in \mathbb{J}_{(k)}} H(X_{(k)}|X_{(j)}). \tag{1.13}$$

Moreover, no extractor is considered in this approach and the same bitstream is transmitted to all decoder. Therefore, considering $X_{(j^*)}$ is in user's memory where $j^* \in \mathbb{J}_{(k)}$, the transmission rate for delivering the compressed bitstream of source $X_{(k)}$ is

$$R_{(k)|(j^*)} = \max_{j \in \mathbb{J}_{(k)}} H(X_{(k)}|X_{(j)}) = S_{(k)}. \tag{1.14}$$

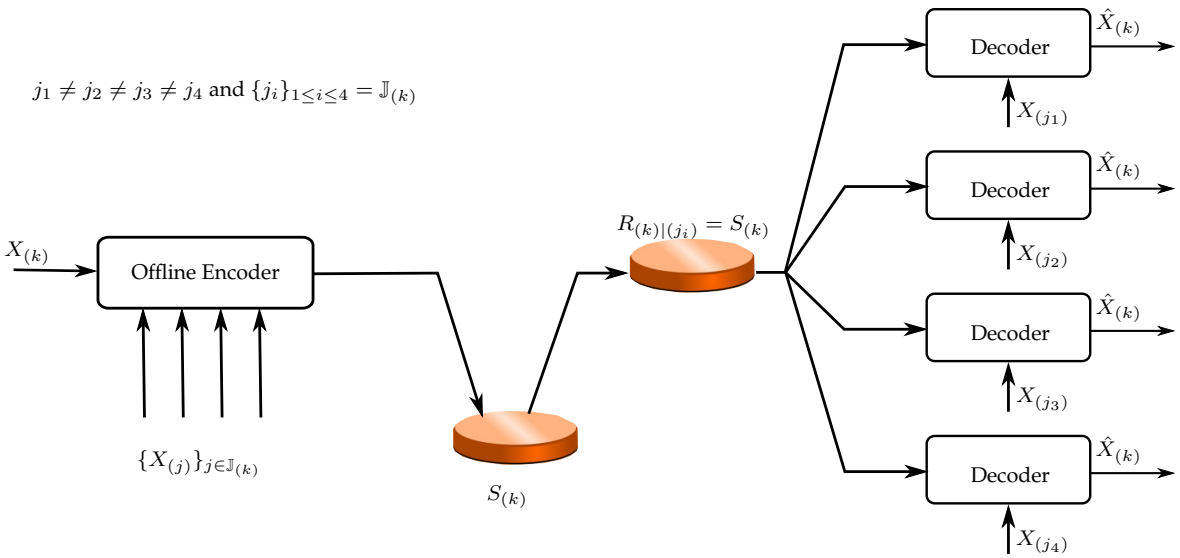



Figure 1.7 – Source coding with different side information at decoders proposed by Sgarro [77]. Note that due to maintaining the connection with interactive coding, the scheme is shown for asymmetric case, i.e., the recovered side information source is available at the decoder, but [77] proves the results for symmetric encoding as well.

**Compound conditional source coding**

Draper and Martinian [27] proposed the first version of interactive coding from an information-theoretic perspective. In their approach, named compound conditional source coding, they do not make any distinction between transmission and storage and similar to [77] the server transmits the same stored bitstream to all users. Therefore, in a compound setting each source is compressed according to the worst-case joint distribution between the source under compression and the sources which are in $J_{(k)}$ (user is allowed to navigate from them to the source). The authors prove that same result as in (1.13) and (1.14) can be stated for compound conditional coding. The scheme of this coding scheme is shown in Fig. 1.8.

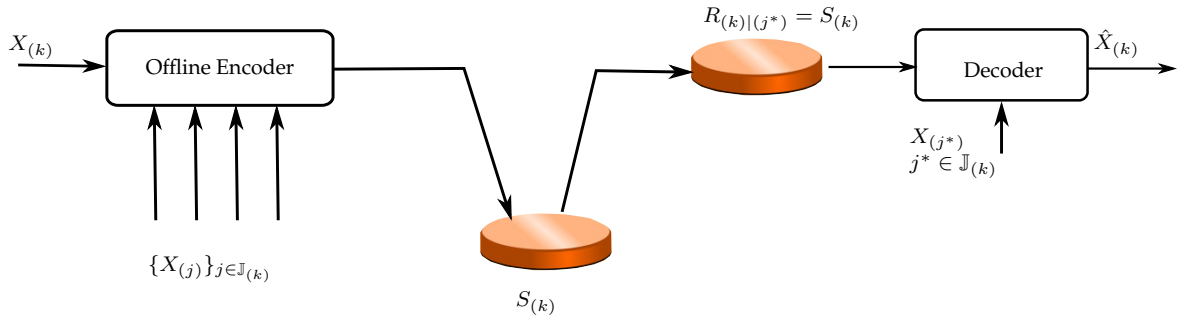



Figure 1.8 – Compound conditional source coding proposed by [27].

**Sequential Massive Random Access (SMRA)**

Although the storage rate (1.13) sounds logical because the offline encoder is not aware of the sources that users will request, the transmission rate can be improved because the server knows the user's previous requests during delivery of data. By using incremental codes and considering an online extractor module, authors in [70, 28, 30] show that the knowledge of current and previous user's request can be used during data delivery to send the data at the optimal rate:

$$R_{(k)|(j^*)} = H(X_{(k)}|X_{(j^*)}). \tag{1.15}$$

The encoding steps are as follows:

1. The encoding graph is equal to the navigation graph. For each source $X_{(k)}$, the offline encoder uses the neighborhood $\mathbb{J}_{(k)}$ defined by the navigation graph and sorts the sources in $\mathbb{J}_{(k)}$ by an ordering mapping function

$$\begin{aligned} \pi_{(k)} : \mathcal{J} &\to [1, \mathbb{J}_{(k)}] \\ j &\mapsto \pi_{(k)}(j) \end{aligned}$$

such that

$$H(X_{(k)}|X_{(\pi_{(k)}^{-1}(1))}) \leq ... \leq H(X_{(k)}|X_{(\pi_{(k)}^{-1}(m))}) \leq ... \leq H(X_{(k)}|X_{(\pi_{(k)}^{-1}(|\mathbb{J}_{(k)}|))})$$

where $m$ denotes the order of a particular source index $j : m = \pi_{(k)}(j)$. Finally, the encoder $f_{(k)}$

provides an incremental codes that stores the bitstreams corresponding to the ordered sources as in (1.1). The existence of such incremental codes is proved in [70, 28, 30]. In practice, this encoding function can be achieved using rate-adaptive incremental codes such as low-density parity-check (LDPC) accumulate (LDPCA) codes [97] and protograph-based LDPC codes [109]. The average storage cost per source for compressing all subgraphs is:

$$S^{SMRA} = \frac{1}{L}\sum_{k=1}^{L} \max_{j\in\mathbb{J}_{(k)}} H(X_{(k)}|X_{(j)}) \tag{1.16a}$$

$$\geq H(\mathbb{V}^{enc}) \tag{1.16b}$$

where (1.16b) holds because in model-based coding the encoding graph is equal to the navigation graph.

2. Upon user's request, the online extractor is aware of the previous user request. The extractor function (1.2) extracts the exact amount of bitstream that is needed based on what user has in the memory. Considering the same sequential request of user $u$ in (1.6), the transmission cost per requested source is

$$R_u^{SMRA} = \frac{1}{|\boldsymbol{I}_u|}\sum_{t=0}^{|\boldsymbol{I}_u|} H(X_{(i_t)}|X_{(i_{(t-1)})}), \quad \text{where } X_{i_0} = \emptyset. \tag{1.17a}$$

$$\approx H(X_{(i_1)}, X_{(i_2)}, ..., X_{(i_{|\boldsymbol{I}_u|})}) \tag{1.17b}$$

$$= R^{oracle} \tag{1.17c}$$

where same as (1.11b), the approximaiton of (1.17b) is because usually, the dependency between sources are Markov chain of order one.

3. The decoder function (1.3) decodes the requested source using the received bitstream and what it has in the memory. In practice the decoding function can be achieved with LDPC iterative Belief Propagation (BP) decoding.

Note that there is no obligation for the elements of $\mathbb{J}_{(k)}$ to be single sources and they can be combination of multiple sources as well. For example, if the user is allowed to navigate from sources $X_{(l)}$ and $X_{(m)}$ to $X_{(k)}$, then the set $J_{(k)}$ can be $J_{(k)} = \{l, m, \{l, m\}\}$. This means that not only each source can be used alone as SI, but also their joint distribution can be used as SI. Note that this doesn't have any impact on storage cost because

$$\begin{aligned} H(X_{(k)}|X_{(l)}, X_{(m)}) &\leq H(X_{(k)}|X_{(l)}) \\ H(X_{(k)}|X_{(l)}, X_{(m)}) &\leq H(X_{(k)}|X_{(m)}). \end{aligned} \tag{1.18}$$

Therefore,

$$\begin{aligned} S_{(k)} &= \max\Big(H(X_{(k)}|X_{(l)}), H(X_{(k)}|X_{(m)}), H(X_{(k)}|X_{(l)}, X_{(m)})\Big) \\ &= \max\Big(H(X_{(k)}|X_{(l)}), H(X_{(k)}|X_{(m)})\Big). \end{aligned} \tag{1.19}$$

The advantage of combining sources is that it not only has no storage overhead but can also reduce

the transmission rate because according to (1.18) this joint distribution can decrease the entropy if both sources $X_{(l)}$ and $X_{(m)}$ are available at the decoder. The result can be extended to multiple sources similarly.

Using this property, we can extend the results to higher-order Markov chain dependencies between sources without having any overhead on the storage. Let $\mathbb{F}_{(k)}$ denote the family of any possible combination of source indices except $k$ with at least one index of $J_{(k)}$ in each combination ($\mathbb{J}_{(k)} \subset \mathbb{F}_{(k)}$). More specifically, the difference between $\mathbb{J}_{(k)}$ and $\mathbb{F}_{(k)}$ is that $\mathbb{J}_{(k)}$ restricts itself to the navigation structure $\mathbb{G}^{nav}$ and contains only the indices of the sources which users are allowed to navigate from them to source $X_{(k)}$ while $\mathbb{F}_{(k)}$ is a set of subsets which contains any possible combination of source indices. If we encode the source $X_{(k)}$ with respect to $\mathbb{F}_{(k)}$ rather than $\mathbb{J}_{(k)}$, we have

$$S^{SMRA} = \frac{1}{L}\sum_{k=1}^{L} \max_{j\in\mathbb{F}_{(k)}} H(X_{(k)}|\mathbb{W}_{(j)}) \tag{1.20a}$$

$$= \frac{1}{L}\sum_{k=1}^{L} \max_{j\in\mathbb{J}_{(k)}} H(X_{(k)}|X_{(j)}) \tag{1.20b}$$

where $\mathbb{W}_{(j)}$ is the combination of sources which their indices are the $j$-th element of $\mathbb{F}_{(k)}$ and (1.20b) holds due to (1.19). The transmission rate per requested source is:

$$R_u^{SMRA} = \frac{1}{|\boldsymbol{I}_u|}\sum_{t=0}^{|\boldsymbol{I}_u|} H(X_{(i_t)}|X_{(i_1)}, X_{(i_2)}, ..., X_{(i_{(t-1)})}), \quad \text{where } X_{i_0} = \emptyset. \tag{1.21a}$$

$$= H(X_{(i_1)}, X_{(i_2)}, ..., X_{(i_{|\boldsymbol{I}_u|})}) \tag{1.21b}$$

$$= R^{oracle} \tag{1.21c}$$

where (1.21a) holds because any possible combination of sources is an element of $\mathbb{F}_{(k)}$. Therefore, the approximation of (1.17b) turns into an equality for higher-order Markov chains. This shows that with no storage overhead, one can achieve the optimal rate as an oracle coder with SMRA. It is worth noting that although considering all combinations for encoding a source has no storage overhead and improves transmission rate, it increases the complexity of the encoding and decoding functions. Later in Chapter 4 we will explain what combination of sources must be exploited depending on the imaging modality that is used.

### 1.2.3 Different compression schemes for interactive communication at a glance

Consider an oracle coder as the baseline defined as a non-interactive coder which knows the user's request in advance or as a coder that encodes all sources in the database jointly (to have optimal storage size) and upon user's request, it decodes the compressed stream, re-encodes only the requested part, and transmits it to the user (to have optimal transmission rate). Comparing SMRA and the tile-based approaches, it can be observed that SMRA can achieve the optimal transmission rate of the oracle encoder, while the tile-based approach is suboptimal in terms of the transmission rate. As for the storage, both approaches experience storage overhead, but no ordering of the two storage costs can be done.

The ordering depends on the source statistics. Comparing the results with the exhaustive storage approach reveals the fact that both SMRA and exhaustive storage have the same performance in terms of transmission rate, which is equal to the optimal transmission rate of the oracle coder. Nevertheless, SMRA can achieve much lower storage cost than the exhaustive storage approach (maximum entropy for SMRA vs sum of entropies for exhaustive storage). Comparing SMRA with compound conditional coding, one can see that both have the same storage overhead, but the SMRA approach can achieve the transmission rate of the oracle coder.

The achievable rates of these interactive coding schemes are summarized in Fig. 1.9. For the tile-based approach we depicted only the two extreme case of tiles: (i) i.e. each source is encoded separately (number of tiles is equal to the number of sources in the database). (ii) all sources are jointly compressed (1 tile). For tile size in between, the performance depends on the source statistics.

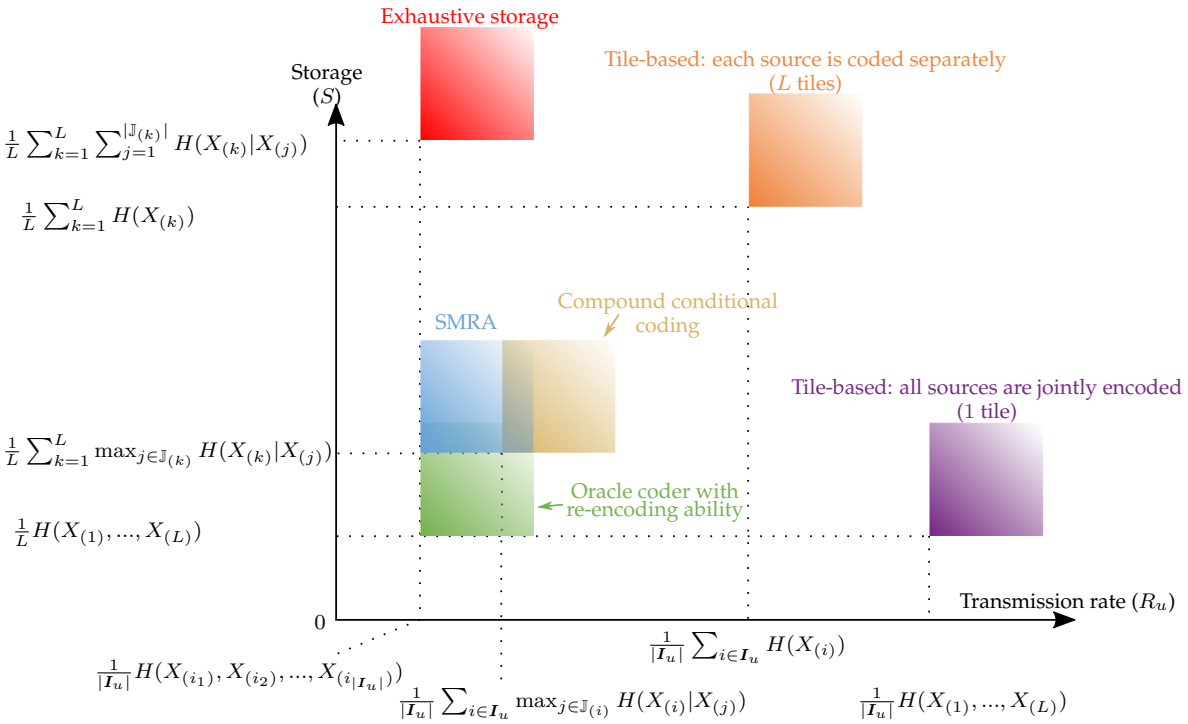



Figure 1.9 – Achievable rates for each interactive compression scheme. The database has $L$ sources and the user requests a sequence $\boldsymbol{I}_u$ of sources as in (1.6).

## 1.3 Image modalities for interactive compression

After reviewing the theoretical bounds of different coding schemes in Section 1.2, we now study existing solutions proposed for different image modalities that are well suited for interactivity. A generic representation of interactive image modalities is 2D images. Therefore, these imaging modalities are usually mapped onto 2D images (if they are not already represented in 2D) to be compressed. In this

section, we introduce three interactive imaging modalities, and then we review existing coding schemes that allow interactivity for each modality. The problem formulation for interactive compression of 2D visual data will be discussed in the next chapter (Section 2.2).

Part of the past works have focused on multiview visual contents for FTV systems in which the user is able to navigate between views. Nowadays, omnidirectional contents and 3D models have gained a lot of interest to be used as interactive media. In this section, for the sake of completeness, we briefly review some of the coding schemes proposed for interactive communication of FTV. We then review compression schemes used for omnidirectional images and texture maps of 3D models which are the main focus of this dissertation.

### 1.3.1 Multiview for FTV

A tile-based approach is used in [53] to encode multiview images capturing a static scene. In particular, the multiview navigation domain is partitioned into segments. Then each navigation segment is encoded and stored independently of the others on a server. Upon sending a navigation segment, the user can navigate freely in the segment without any further data request to the server. The server transmits additional data only when the user moves to a different segment.

In [15, 14], a compound conditional coding scheme is proposed for intearctive compression of multiview videos. More precisely, a Distributed Source Coding (DSC) solution is used to merge multiple decoding paths (each with a different predictor) into an identical reconstruction. This technique is called merge-frame (M-frame) and is used to avoid error propagation (coding drift) in the subsequent frames. Then, the compound conditional coding scheme is used to exploit the worst-case correlation between the set of potential predictor frames and the target frame.

In [24], another approach to achieve M-frame is proposed using a piecewise constant function as the merge operation. In this technique, the difference between the set of predictors and the M-frame is coded using an optimized type of exhaustive storage. More precisely, to avoid storing all possible navigations, rate-distortion optimization techniques are used to choose whether a frame must be coded independently or with the M-frame technique. Later, this technique is extended for interactive light field streaming [57]. To do so, by proposing an interaction model, typical user navigation tendencies are taken into account to store only navigations that are highly likely to be requested by the user. Then, a greedy algorithm is proposed to find redundant frame structures while minimizing a weighted sum of expected transmission cost and storage.

### 1.3.2 Omnidirectional content

Omnidirectional or 360° images/videos are spherical visual signals that can represent the scene from one point with full 360° Field of View (FOV). These contents are becoming increasingly popular as they offer an immersive experience in virtual environments. Although these images are modeled as spherical images [56], a single omnidirectional camera sensor with a full 360° FOV does not exist in practice. Therefore, commercial omnidirectional cameras consist of multiple sensor systems in which their images are stitched together to provide the final 360° image [88].

Before speaking about interactive compression, we discuss here the specificity of 360° image modality. For that, we start by presenting different 2D representations that exist for these contents and we then review the drawbacks of these 2D representations in terms of compression. Finally, we review the current solutions that are proposed for their interactive compression.

#### 1.3.2.1 Omnidirectional content representations

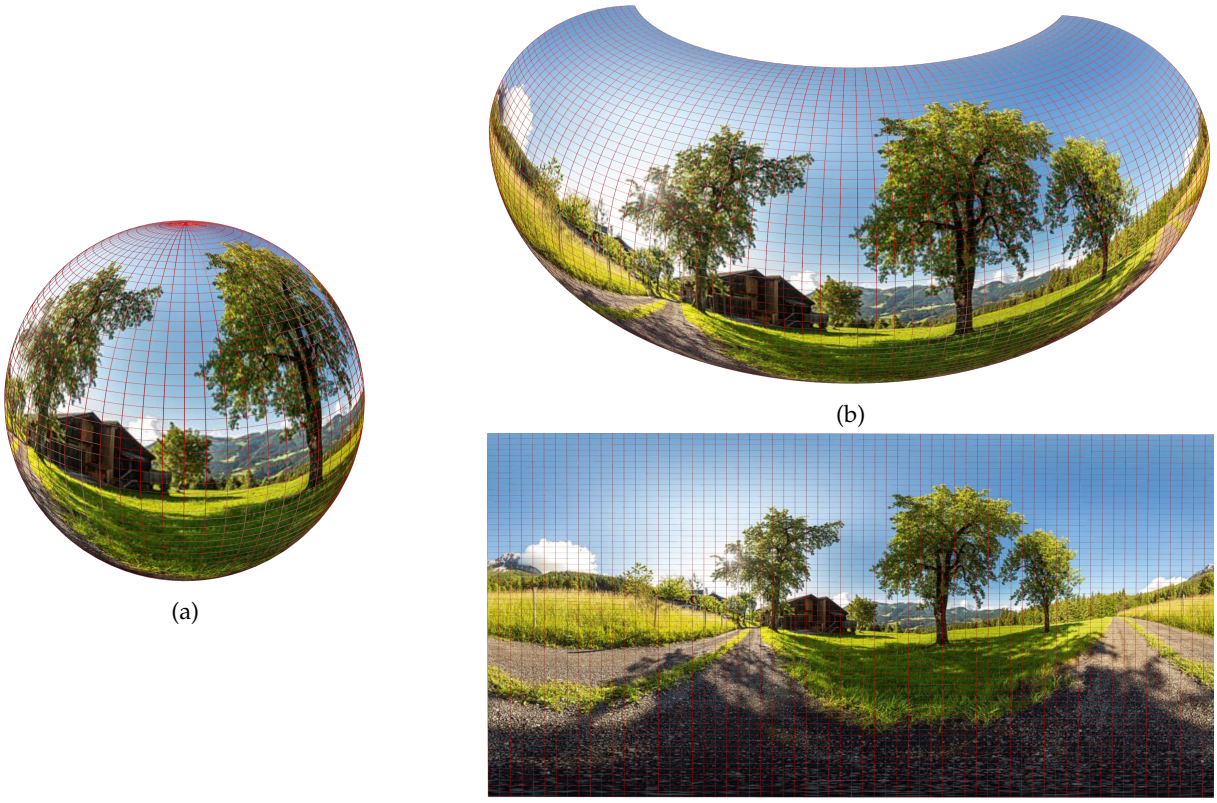
The output of the stitching process, which in principle is a spherical signal, is often projected onto one or multiple 2D planes. The planar representation allows re-using latest technologies (which are optimized for the 2D planar video) to compress, store and transmit the 360° contents. Different representations have been proposed, such as equirectangular, cube map, rhombic dodecahedron, etc. [13]. In the following we review the most commonly used representations.

***Equirectangular projection.*** The most popular mapping for 360° images is the equirectangular (also called equiangular) projection which samples the spherical signal based on an equiangular grid of longitude ($\in [-\pi, \pi)$) and latitude $\in [-\pi/2, \pi/2]$ values. More specifically, it maps the equidistant meridians and equidistant circles of latitude to an image plane where rows represent the circles of latitudes and columns represents meridians [85]. This can be seen as unfolding the sphere along a meridian seam that connect the north pole to the south pole (see Fig. 1.10). For compression, the equirectangular representation is fed into conventional 2D compression tools to compress the signal. The main drawback of this projection is its horizontal oversampling near the poles of the sphere which stretches the shape of objects around this area and results in large shape distortion.

***Cube map projection.*** Another projection that is often used to represent 360° contents is the cube map projection in which the sphere is circumscribed by a cube [58]. Then, the sphere is projected onto the faces of the cube using a radial projection. Fig. 1.11 shows this projection. To store the data, the faces of the cube are packed together into one planar image and then compressed using conventional 2D compression tools. Facebook reported that cube map projection reduces file size by 25 percent against the equirectangular projection [44].

***Dyadic projection.*** To solve the horizontal oversampling problem of equirectangular projection, dyadic projection is proposed in [110]. In this projection, they halve the horizontal resolution for areas where the absolute of latitude values is greater than or equal to $\pi/3$.

***Rhombic dodecahedron projection.*** Authors in [35] propose a rhombic dodecahedron projection. As shown in Fig. 1.12, a circumscribed rhombic polyhedron of the sphere is constructed such that the center of each rhombus touches the sphere surface. Then the 12 rhombi are projected on the sphere surface using radial projection to have 12 equal area spherical rhombi on the sphere. Then, to have a finer uniform sampling, each spherical rhombus is subdivided hierarchically using a skew great circles technique. Finally, for compression, the 12 rhombi are packed into 4 rectangular planes with $4 \times 3$ arrangement, and each plane is compressed separately.



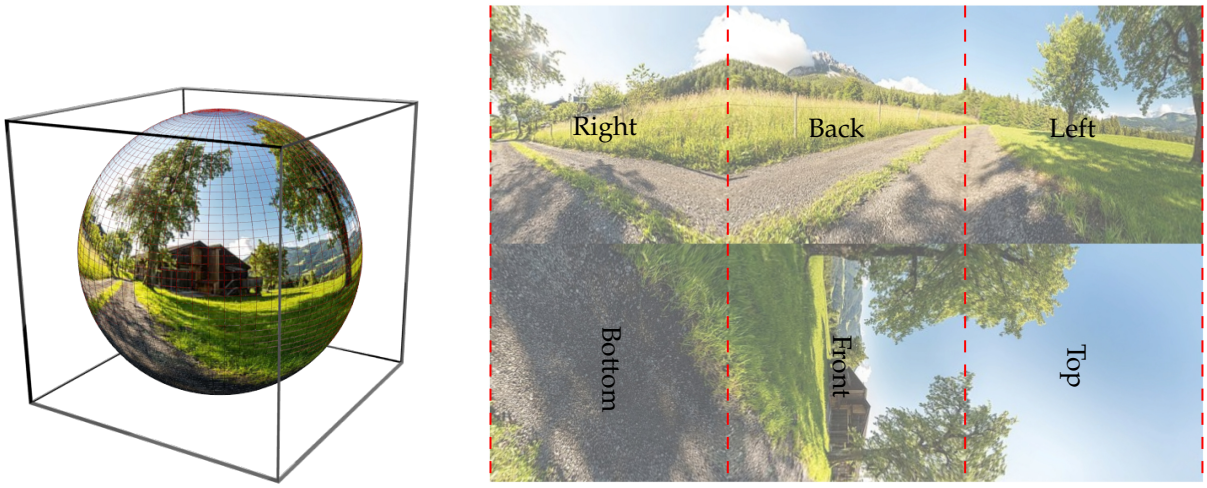
(a) (b) (c)

Figure 1.10 – Equirectangular projection. (a) spherical image. (b) Unfolding along a meridian seam to generate equirectangular image. (c) Equirectangular projection. The original image is taken from "Ansgar Koreng / CC BY-SA 4.0":`https://fr.wikipedia.org/wiki/Fichier:Oberkaisern,_Ellmau,_Tirol,_360x180,_160620,_ako.jpg`.

#### 1.3.2.2 Drawbacks of 2D representations

To achieve good compression performance, a 2D projection must satisfy the three following properties. First, *(i)* the projection should lead to the same distortion and the same image statistics as the ones induced by perspective projection, such that the projected images have the characteristics which the conventional 2D video coders have been optimized for. Second, *(ii)* the sampling should be uniform on the sphere, to achieve fair rate-allocation. Finally, *(iii)* the connectivity between neighboring samples on the sphere should be preserved after projection, to better exploit spatial dependencies.

***(i) Same statistics as perspective projection.*** The traditional 2D compression tools are optimized for images which are captured by perspective cameras. To preserve their encoding efficiency for compression of the spherical contents, the mapping between the 2D representation and the sphere must be an isometric isomorphism. This means that the mapping must be a distance-preserving transformation that preserves the shape of the objects. In other words, to have the same encoding performance as we expect in perspective images, the objects should have the same shape in both the perspective image and




(a) (b)

Figure 1.11 – Cube map projection. (a) A sphere is inscribed in a cube. (b) Packing of the 6 faces into one 2D rectangular plane.

spherical image. However, an interesting theory proven by Carl Friedrich Gauss states that there exists no mapping between a 2D plane and a sphere that preserves the statistics. More precisely, the theorem is stated as follows:

**Theorem 1** (Gauss's Theorema Egregium [61])**.** *The Gaussian curvature of a surface is preserved by isometry.*

This implies that if two surfaces do not have the same curvature, there is no isometry between them [18]. As a consequence, a sphere which has a curvature value of 1 cannot be displayed on a "flat" 2D plane (with 0 curvature) without distortion, or vice versa. A comprehensive review of visual distortion in spherical content is given in [1]. The existence of this distortion states that the 2D representation of a sphere cannot have the same statistic as a perspective image thus traditional 2D encoding tools lose their optimal performance when they are encoding spherical images.

***(ii) Uniformity of the sampling.*** An ideal representation positions the pixels such that the number of stored pixels in the representation participating in viewport production be almost equal to the number of pixels in the viewport. For instance, if the viewport has 10000 pixels, it should almost require 10000 pixels of the stored representation file. A useful visual method to analyze the uniformity of spacing between different projection types is to use *saturation maps* [9]. A saturation map shows a color-coded ratio of video pixel density to display pixel density for each direction the viewer is looking at. Note that saturation is highly dependent on the size of the image as well as the resolution of the viewport and we can change the overall colors on a saturation map by increasing the resolution of the representation. Therefore, an ideal representation is a representation that has less variation, i.e., it must have uniform color in the saturation map as much as possible.

The saturation maps for the two most commonly used representations, i.e., equirectangular and cube map representations, are shown in Fig. 1.13. As can be seen, in the equirectangular projection there are

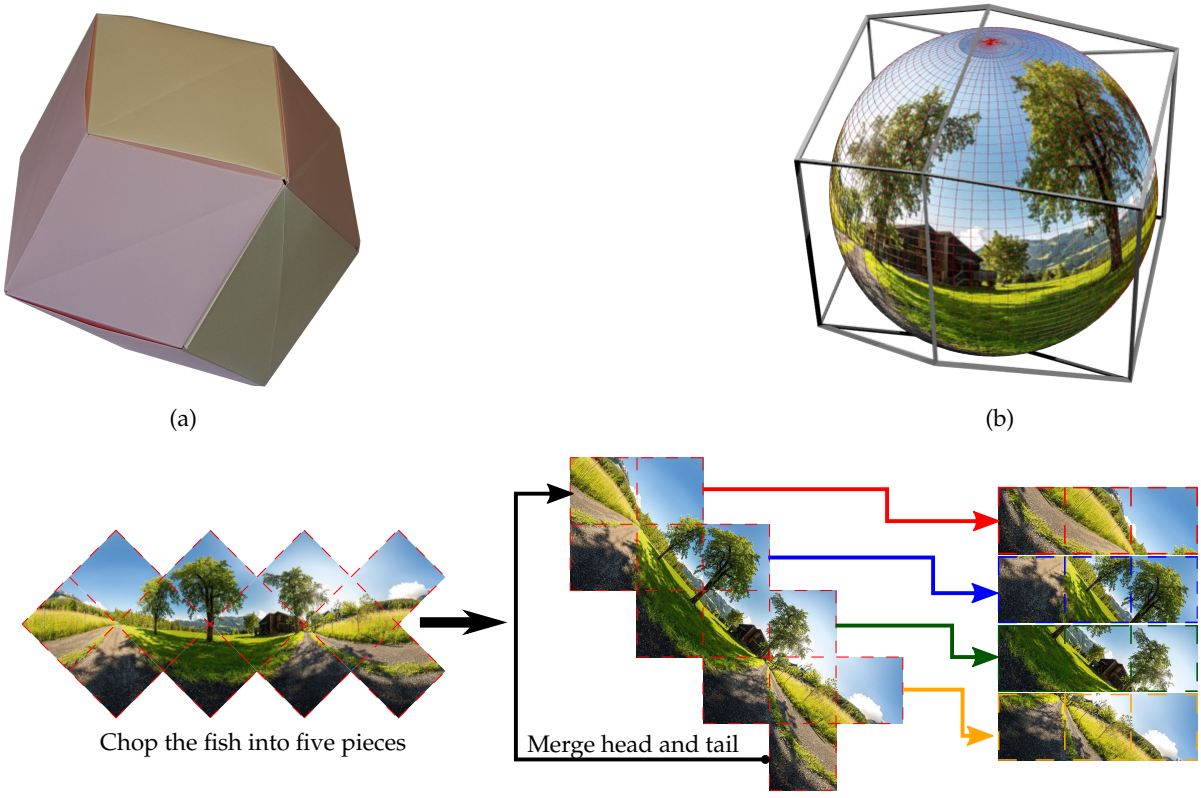
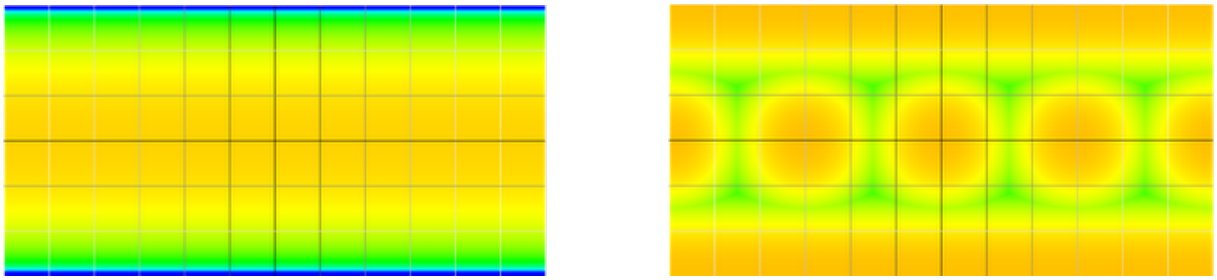



Figure 1.12 – Rhombic dodecahedron projection. (a) A paper model of rhombic dodecahedron (special thanks to Joachim Wiest for providing this paper model). (b) A sphere is inscribed in a rhombic dodecahedron. Each rhombi is projected on the surface of the sphere using radial projection . (c) Packing into 4 rectangular planes for compression.

too many wasteful pixels around the poles and on the contrary, the equator has too many few pixels compared to the poles. In the cube map representation, one can see the improvement with the removal of the wasteful areas around the poles, but the variation of pixel densities is high around the equatorial region.

***(iii) Connectivity preservation.*** A sphere is a *closed surface*, i.e., a surface that has no boundaries. Therefore, any projection of a surface into the 2D plane (which are not closed by their nature) imposes disconnectivity in some regions of the spherical content. For example, in the equirectangular projection the vertical boundaries which are far from each other in the 2D image are highly correlated because they are connected on the sphere surface, or in the cube map projection as can be seen in Fig. 1.11b, there is no continuity between the upper and lower parts around the centerline of the image. The dis-connectivity in rhombic dodecahedron is also obvious due to the fact that it projects the spherical signal into four different 2D planes. By losing these connectivities, the encoder is not able to exploit the spatial dependencies, which is a vital operation in 2D compression schemes and this results in a performance drop of the compression tool.



(a) (b)

Figure 1.13 – Saturation maps for the equirectangular and cube map representations taken from [9]. Green indicates an optimal pixel density ratio that is near 1:1. Red, orange and yellow indicate insufficient density (too few projected pixels with respect to the pixels on the sphere), and blue indicates wasted resources. (a) Equirectangular representation. (b) Cube map representation.

To sum up, all the 2D projections listed in Section 1.3.2.1 fail to meet simultaneously these three requirements (see Table 1.2 for more details). For these reasons, there is the need to redesign conventional video coders to adapt to the characteristics of spherical contents.

Table 1.2 – Characteristics of different 2D representations in 360° contents

| projection / requirement | cube map [58, 44] | rhombic [35] | equirect. [85] | dyadic [110] |
|---|---|---|---|---|
| (i) same statistics as perspective projection | almost Yes | No* | No | No |
| (ii) uniform sampling on the sphere | No | almost Yes | No | approx. Yes |
| (iii) connectivity preservation | No | No | No | No |

* Due to fitting rhombi to rectangles

To cope with these drawbacks, one approach is to extend the 2D plane compression algorithm and take into account the 3D geometry. For instance, new motion compensation predictions are proposed in [25, 99, 100]. In [25], instead of applying block matching on the equirectangular domain, the motion is modeled as a translation on a plane tangential to the sphere and the block matching is performed among all motion vectors on this tangential plane. In [99] the authors propose a rotational motion model to perform motion compensation. For that, they map the center of the coding block in the equirectangular domain on to the sphere. Then, they apply a radial search pattern around the center of the block by shifting the longitude and latitude values and the rotation corresponding to this shift is computed. Finally, all neighboring pixels in the block are rotated with the same rotation and mapped back to the reference frame in the equirectangular domain. Later, the authors extended this idea for the case when the camera has translational motion [100].

Another approaches focus only on the transform coding of the content (no prediction is used to further remove the spatial correlation within the image). In [67], a graph-based coder adapted to the spherical surface is proposed for equirectangular projection. This graph transform compensates for the non-uniform sampling by weighting the graph with the geodesic distance between samples. In [94], by

proposing a redundant dictionary of atoms living on the 2D sphere, the geometry of the data is taken into account. Then, the matching pursuit algorithm is used to choose atoms from the dictionary. Finally the sorted atoms are quantized using an adaptive quantization.

We can see that all these methods compensate for some of the drawbacks of the projection but do not handle all of them. Later in Section 4.2 we propose a spherical compression scheme that performs all processing operations directly on the sphere.

#### 1.3.2.3 Interactive communication of 360° contents

In the spherical visual contents, users can freely observe the surrounding area by means of HMDs or hand-held pointing devices as with the Google Maps Platform. When a user watches a 360° video in direction $\theta$, at each time only a portion of the whole spherical content is displayed on the user's *viewport*. This viewport (shown in Fig. 1.14a) is the projection of the spherical image onto a plane perpendicular to the direction the user is pointing to ($\theta$), and is tangent to the sphere. It can be generated by placing a virtual camera at the center of the sphere and rendering the spherical video in its FOV.

(a) (b)

Figure 1.14 – Viewport requested from a 360° visual content. (a) An example of a viewport (in blue) requested by the user. The viewport plane is tangent to the sphere at the viewport center. The user direction is shown with a green line inside the sphere (from the sphere center to the viewport center). (b) An example of a portion corresponds to a request of full HD viewport with FOV of 110°. A request of the viewport corresponds to a small portion of the spherical content.

With today's technologies, available HMDs on the market provide up to 4K (HP Reverb VR headset [1]) or 8K (Pimax 8K VR Headset [2]) viewport resolution with around 110° to 220° of FOV, which covers only around 15%-40% of the whole spherical content (Fig. 1.14b). This means that to provide a high feeling of immersion the original spherical content should be stored with resolution 12K or higher [20]. Consequently, compression of this data must be performed interactively so that only the portion of the data that the user needs can be extracted from the compressed data. To relate the subject to the formal definition given in Section 1.2.1, it can be said that the user moves smoothly within the content and cannot make sudden jumps in the content. This allows the system to know the set of possible

1. https://www8.hp.com/us/en/workstations/mixed-reality-headset/index.html
2. https://pimaxvr.com/products/8k

next requests per request. We explain in more details the definition of sources and navigation graph constraints in Chapter 3.

A number of *viewport-dependent* streaming, also called *viewport-adaptive* or *viewport-aware*, schemes have been proposed to exploit this fact. In contrast to the *viewport-independent* 2D representations (discussed in Section 1.3.2.1) in which the whole content representation is transmitted, these representations consider the FOV and try to transmit (as much as possible) the needed information requested by the user. These approaches are divided into two categories: The approaches that rely on Quality Emphasized Region (QER) and the ones that are based on tile-based approaches. It is worth noting that all these approaches use classical predictive coding technique (see Section 1.2.2.1) to compress the data.

In the tile-based approach, the video is split into interdependently decodable parts (tiles) and each tile is encoded separately. In [62], the system guesses the next user request using a head movement prediction system, and then the server transmits only the tiles (in the equirectangular domain) corresponding to the request. In [113] the tiles containing visible area are transmitted in high quality and the other tiles are streamed in low-quality level. Other works reduce the transmission rate from the network perspective by considering the interactive information to choose the best tile configuration to download at the client's side. For example, a navigation-aware transmission strategy is proposed in [68] to optimize user viewport quality while respecting the network bandwidth. Representations other than equirectangular are also considered. A cube map representation with tiles is proposed in [89]. In [39] along with the video, the underlying 3D geometry of the sphere is also divided into six 3D submeshes (called hexaface) with a unique identifier. Then, a mapping between the tiles in the video and sub-meshes is defined using the MPEG-DASH-SRD.

One problem with tile-based approaches is that the video at the user's side must be reconstructed from independent tiles. This can increase the latency [20]. To solve this problem, QER-based approaches are proposed. This scheme can be considered as a variant of the exhaustive storage scheme in which a certain area is favored in the representation with higher quality than the rest of the spherical regions. As an example, a pyramid projection is proposed in [44]. In this projection, the spherical content is mapped to multiple pyramids (30 pyramids in different directions) where the base of each pyramid presents a pre-defined viewport in higher resolution and the rest of the pyramid represents the non-viewport part in lower resolution. Upon user request, the pyramid whose base is closer to the user's viewing direction will be transmitted to the user. A similar approach is proposed with the truncated square pyramid which uses quadrangular frustum pyramid [5]. In [20] the authors show that among different geometrical layouts with QER property which are used for spherical-to-plane projections (equirectangular with 8x8 tiles, cube map, pyramid, and dodecahedron), cube map projection offers the best quality for a given bit-rate budget. It is worth noting that although these schemes reduce the transmission cost, they lead to high storage overhead due to the overlap between QER in different directions.

### 1.3.3 3D model texture

With the recent development of 3D scanning technologies, 3D models have been finely detailed. This has made the 3D representation a trending media format for novel applications, such as architectural or historical heritage presentation [59], virtual reality [48], etc. As a result, the ability to store and transmit

these content easily through networks has attracted great attention in both academia and industry. A 3D model is usually represented by 3D meshes (shown in Fig. 1.15) where a mesh contains some geometrical information and several texture/atlas maps. The geometrical information consists of some 3D vertices and their corresponding topology (connectivity among elements usually represented by triangles), as well as optional attribute information such as normals, colors, etc. To make the rendering more realistic, usually, giant 2D texture maps such as the diffusion, specular and bump maps are applied on mesh surface to present material properties, high-frequency details, *etc*.

(a) (b) (c)

Figure 1.15 – Components of a 3D model. (a) A 3D model. (b) The triangular mesh and (c) its texture atlas.

The compression of 3D mesh *geometry* has been widely studied. A survey of such methods can be found in [51]. Since the focus of this thesis is on compression of imaging modalities, in the following we briefly review the available techniques exist in compression of mesh geometry for the sake of completeness. In general, the compression of the mesh geometry can be categorized into static mesh and dynamic mesh compression approaches. *Static* mesh compression approaches by themselves are divided into three categories: *single-rate algorithms* try to build a compact representation of an input mesh. In *progressive algorithms*, as in [54], the input mesh is iteratively decimated until a base mesh is generated. This provides a successive levels of details for the input mesh in which a coarse version of the mesh can be quickly displayed to the user and as more data are decoded, this coarse mesh is then progressively refined. *Random-accessible algorithms* like [50] allow decompressing only the required parts of the mesh to avoid the need to load and decompress the full model. On the other hand, *dynamic* mesh compression, also known as mesh sequence compression, can be classified into two types: the approaches that try to compress temporally coherent mesh sequences in which each frame has the same number of vertices and connectivity and the ones that try to compress temporally incoherent mesh sequences like [107] that have variable connectivity.

The *textural* information generally consists of a giant image describing the whole texture information. While the compression of textures from the perspective of rendering systems has been thoroughly studied [7, 43], the texture compression from the perspective of transmission or storage is almost overlooked. The former aims at compressing the texture maps to reduce the memory usage at the run-time and to provide the ability to render directly from the compressed textures loaded in rendering systems

(having fast access to image pixels in graphics processing unit). The latter compresses the texture for storage and transmission, i.e., to reduce download or disk size.


Figure 1.16 – An example of user's navigation around the 3D model (left), what he observes at a given instant (center), and the corresponding visible blocks $\mathcal{V}$ (in red) in the atlas (right). For better visibility of the blocks, the black background area is turned into white in the atlas.

Up to now, traditional still image compression schemes such as JPEG [101], are often used for storing texture maps. Although such approaches may provide up to 50:1 compression rate for texture maps of the meshes [43], they are not optimal yet for this type of content because of the two following reasons. First, a texture image is usually defined as an atlas of *texture patches*, with a flat/smooth background (see Fig. 1.15(c)). Even if optimizations of the patching arrangement have been proposed to lower the background size [6, 34, 98, 108], the projection of the texture content onto a 2D image introduces unavoidable artificial edges, and this leads to a rate overhead when the encoding is performed with a regular image coder. Second, the atlas encoder inherits its property from conventional coder, *i.e.*, the whole image has to be decoded. Nevertheless, this is not desirable since during user navigation in the scene only a subpart of the 3D model, and therefore part of the atlas, is usually observed at a given time during the rendering (Fig. 1.16). It would be much more efficient in terms of rate or memory load if only the needed part of the model could be decoded, as pointed in [81].

## 1.4 Summary

In this chapter, we first introduced the problems that exist in interactive compression of data. Then, with the help of a navigation graph structure that introduces the possible set of navigation per request, we proposed a framework to revisit different interactive coding schemes from a theoretical perspective. More specifically, by making a distinction between storage and transmission rates, which is a critical property in interactive compression, we analyzed the transmission and storage theoretical bounds that are achievable in each approach. In general, the existing solutions for interactive compression can be divided into two categories. The first is the classical approaches which are based on predictive coding of the data. The second category is the methods that use model-based coding to provide interactive compression.

The predictive coding scheme by nature is not efficient for interactive coding as it imposes a fixed pre-defined encoding and decoding order of sources. To support interactivity, this scheme is modified by

Table 1.3 – Performance of different interactive coding schemes.

| | **Predictive coding** | | **Model-based coding** | |
|---|---|---|---|---|
| | **Tile-based** | **Exhaustive storage** | **Compound conditional** | **SMRA** |
| **Storage rate** | Not optimal | High storage overhead | Reasonable storage overhead | Reasonable storage overhead |
| **Transmission rate** | Not optimal | Optimal* | Not optimal | Optimal* |

* Optimal for interactive coding, i.e. as an oracle coder

one of the two following methods: tile-based approach or exhaustive storage approach. The tile-based approach provides interactivity by partitioning the data into separated parts. Let's consider an oracle coder as the baseline defined as a non-interactive coder which knows the user's request in advance or as a coder that encodes all sources in the database jointly (to have optimal storage size) and upon user's request, it decodes the compressed stream, re-encodes only the requested part, and transmits it to the user (to have optimal transmission rate). The tile-based approach is neither optimal in terms of storage rate nor transmission rate. This is mainly because by splitting the database into separate pieces, some dependencies between sources that could help to have better compression are lost. This approach is also inefficient for transmission because more sources that what is needed are transmitted. To achieve optimal transmission rate, exhaustive storage is proposed in which all possible navigations are compressed separately. However, this method imposes a very high storage overhead.

The model-based coding scheme is a good alternative to the classical predictive coding approaches of interactive compression. This coding scheme inherits its power from DSC. The first version of this coding scheme is compound conditional coding where a source is compressed with respect to a set of SI sources knowing that one of them is available at the decoder. The compression is performed according to the worst SI (less correlated). An interactive coding scheme based on the compound scheme does not make distinction between storage and transmission, and it results in a small overhead for both storage and transmission rates with respect to the oracle scheme. Therefore, even if a better SI is available at the decoder, the system still transmits the information according to the worst-case SI. Recently, SMRA approach was proposed and it is proved that by the help of incremental codes, we can perform better in terms of the transmission rate. Therefore, with the same storage overhead as compound conditional coding, we can achieve the optimal transmission rate with SMRA. A summary of the performance of each method is given in Table 1.3.

We then studied the existing solutions proposed for different interactive imaging modalities. For that, we mostly focused on omnidirectional contents and texture images of 3D models which are tackled in this dissertation. We first reviewed existing suggested 2D representations for omnidirectional contents and their drawbacks in compression. We then studied different approaches that are currently used for their interactive compression. For texture maps of 3D models not much has been done for interactive compression of such images, and often the entire image is compressed and sent upon request. One can notice that all practical solutions proposed so far for interactive compression of these contents rely on classical predictive coding which is unfortunately not optimal neither in terms of storage rate nor transmission rate. However, the theoretical results of SMRA show that we can expect no transmission rate penalty with reasonable storage overhead. The focus of this dissertation, which will be discussed in later chapters, is to fill this gap by providing a generic interactive compression scheme that can deal with various image modalities.

CHAPTER 2

# EVALUATION FRAMEWORK WITH VIEW-DEPENDENT TRANSMISSION

The immersive visual experience is obtained by allowing the user to navigate in the visual content. Usually, these contents are mapped onto 2D images for presentation. These data are stored in high resolution and need a large amount of memory on the server to store them. However, the transmission depends on the user, and only the spatial region requested by the user is sent to prevent wasting network bandwidth. Splitting the rates into storage and transmission has not been formally considered in the literature for evaluating interactive compression algorithms of 2D images. In this chapter, we first formulate the compression problem of interactive 2D visual contents by extending the result of Section 1.2.1 to 2D images and considering lossy compression. This formulation will be the basis for the rest of this dissertation. We then propose a framework to evaluate the interactive coding efficiency while discriminating between storage and transmission rate and taking into account the user-dependent requests. This brings the flexibility to compare different coding methods based on the storage capacity on the server and the network bandwidth of users.

The organization of the chapter is as follows. After giving the motivation, with the help of Section 1.2.1, we reformulate the interactive coding problem for 2D imaging modalities. We then explain the proposed evaluation framework. Finally, we illustrate the evaluation methodology using the existing classical coding schemes on 360° content.

## 2.1 Motivation

The first step before presenting any method to compress different interactive imaging modalities is to provide a framework for comparing different methods. We have already introduced two important components of interactive compression, namely storage $S$ (the amount of data stored on the server), and transmission rate $R$ (the amount of data sent to the user). However, these interactive visual data, which are represented with 2D images, are often lossily compressed. Therefore, a distortion $D$ of the images (quality of the view displayed to the user) should also be considered in the evaluation process.

Despite several contributions that have been made to improve *user-dependent* compression of interactive 2D media, there is no proper evaluation framework to compare different coding schemes. This might be explained by the fact that the proposed methods (see Section 1.3) rely on classical 2D compression standards. Therefore, the evaluation of these coding schemes still relies on classical 2D metrics. For instance, the classical rate-distortion (RD) plots are used for the evaluation in [110] because the

whole 360° content is sent to the user ($R = S$). They state that to have a proper evaluation for the *quality of experience*, the distortion must be computed on the viewport shown to the users. They showed experimentally that the computation of the average users' viewport quality can be approximated by a weighted spherical Peak Signal-to-Noise Ratio (PSNR). In [96], again the whole 360° content is sent ($R = S$), and the authors provide subjective quality evaluation and compare the results with a range of objective quality metrics. In [39], a rate allocation strategy is proposed to lower the transmission rate, therefore only $R$ is considered and not the trade-off between $R$ and $D$. In [112], the trade-off between $R$ and $D$, displayed with RD curves, and the sum of storage are used for the evaluation. However, the trade-off between $S$ and $D$ is not considered. Moreover, to reduce the computational cost of computing the distortion on each requested viewports, they approximate the distortion using a set of discrete viewing orientations that do not depend on the requests of the users.

To the best of our knowledge, none of the existing evaluation methods in interactive communication consider jointly storage, transmission rate, and distortion. The goal of this chapter is to provide a framework that can evaluate the efficiency of different interactive compression algorithms while considering all these three criteria together. For that, we propose a *user-dependent* evaluation framework to compare different compression methods in which the coding error is computed in the viewport shown to users (same as [110]). Therefore, the distortion that users sense at their side is considered for the evaluation. Moreover, the framework differentiates the transmission rate from storage. More specifically, we propose three metrics. First, we propose to use Bjontegaard metric [8] to compute average storage and transmission rate saving for the same quality experienced by the user. Second, we propose iso points to compare methods when the system must satisfy fixed constraints. Finally, we extend the Bjontegaard metric by combining the transmission rate and storage to take into account further constraints in the evaluation. The evaluation framework brings the ability to achieve a compromise between the capacity of the server to store the data and users' bandwidth.

## 2.2 Problem formulation for interactive compression of 2D visual contents

Before describing the evaluation framework, we first formulate the compression problem of interactive imaging modalities represented in 2D. We extend the formulation explained in Section 1.2.1 to 2D visual data by considering mapping function for data representation and lossy compression. This model is depicted in Fig. 2.1 for 360° contents and 3D models, and includes the three steps of interactive compression: encoding (Section 2.2.1), extraction (Section 2.2.2), and decoding (Section 2.2.3).

### 2.2.1 Data model, projection and compression for storage on the server

For interactive visual media, the data $\Gamma$ is first mapped onto a Euclidean space. One reason for this mapping is that available coding schemes are mostly optimized to compress 2D image/video formats, so the data is mapped onto 2D to be able to use these compression tools. The other reason is that this 2D Euclidean space can be used as a common representation to deal with various types of interactive media. We denote this Euclidean 2D representation, also called the *2D image*, by $I$. For instance, the

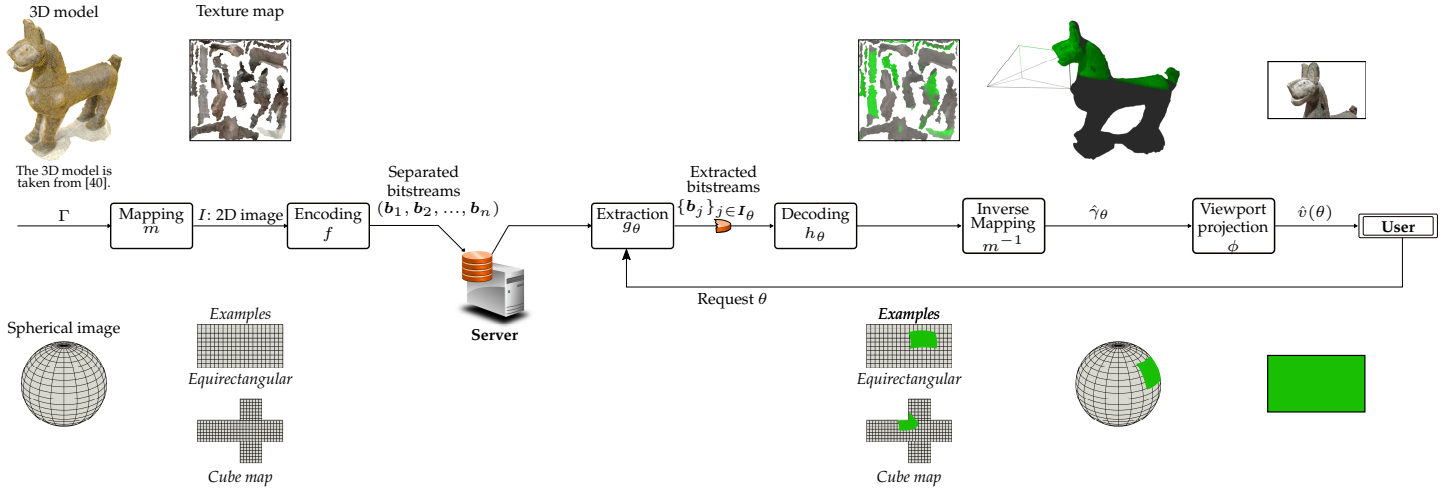



Figure 2.1 – Processing pipeline of compressing an interactive medium $\Gamma$. Mapping and encoding operations are performed offline whereas extraction, decoding and viewport construction are performed online, upon user's request.

equirectangular projection of spherical images consists in uniformly sampling the spherical coordinates (latitude and longitude values) and replacing them by euclidean coordinates: $(u, v) \approx \theta$, or texture mapping of a triangular mesh consists in piecewise affine mapping of the 2D texture image to the 3D mesh triangles defined by assigning $(u, v)$ coordinates of the texture image to the vertices of the 3D mesh. The mapping is denoted by

$$m : \Gamma \mapsto I. \tag{2.1}$$

Then, compression is performed with the encoding function $f$. To allow interactivity, the compression leads to a set of independently extractable bitstreams $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$:

$$(f \circ m)(\Gamma) = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \tag{2.2}$$

where $\circ$ stands for the composition of functions. The *storage cost*, i.e., the size of the data stored on the server is defined as:

$$S = |f \circ m(\Gamma)| = \sum_{i=1}^{n} |\boldsymbol{b}_i| \tag{2.3}$$

### 2.2.2 Bitstream extraction

A user's observation with parameter $\theta$ triggers a request of a subpart of $\Gamma$ denoted by $\gamma(\theta)$. For instance, $\theta$ for spherical images denotes the user's viewing direction determined by latitude and longitude values of spherical angles, while $\theta$ for 3D models denotes the user's position and viewing direction (6 DOF). Upon request, the server extracts parts of the stored bitstreams with indices $i \in \boldsymbol{I}_\theta$ related to $\gamma(\theta)$

$$g_\theta : (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \mapsto (\boldsymbol{b}_i)_{i \in \boldsymbol{I}_\theta}$$

and sends it to the user at rate

$$R_\theta = |g_\theta(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)| = \sum_{i \in \mathcal{I}(\theta)} |\boldsymbol{b}_i| \tag{2.4}$$

where $R_\theta$ is called the *transmission rate* and $R_\theta \leq S$.

### 2.2.3 Bitstream decoding

At the user's side, the received extracted bitstreams allow to reconstruct $\gamma(\theta)$, i.e., the requested subset of the spherical content, with the decoding function $h_\theta$:

$$\begin{aligned} h_\theta &: (\boldsymbol{b}_i)_{i \in \mathcal{I}_\theta} \mapsto \widehat{(m \circ \gamma)}(\theta) \\ m^{-1} \circ h_\theta &: (\boldsymbol{b}_i)_{i \in \mathcal{I}_\theta} \mapsto \hat{\gamma}(\theta), \end{aligned} \tag{2.5}$$

Note that due to lossy compression $\gamma(\theta) \neq \hat{\gamma}(\theta)$.

Finally, the *viewport* $v(\theta)$ is displayed to the user (Fig. 2.1). This image is generated by assuming a virtual camera positioned at the center of the sphere (for spherical images) or the position determined by the user (for 3D models) which orients towards user's viewing direction. The viewport is the projection result (denoted by $\phi$) of $\gamma(\theta)$ onto this virtual camera's image plane. Note that the size and resolution of the viewport determine the subset $\gamma(\theta)$ of pixels requested.

To measure a realistic quality of experience at the user side, the distortion is measured on the image displayed to the user [110]. Formally, *the distortion* is

$$D_\theta = ||v(\theta) - \hat{v}(\theta)||_2^2 = ||\phi(\gamma(\theta)) - \phi(\hat{\gamma}(\theta))||_2^2. \tag{2.6}$$

This distortion is usually presented in terms of PSNR.

Both distortion $D_\theta$ and rate $R_\theta$ vary significantly with $\theta$. Therefore, to be able to compare performance between different methods, the *expected* values over all users' requests are computed for these quantities. Compared to classical 2D video/image coding, the use of expected values for these quantities is new and derives from user-dependent navigation. In general, for the same amount of storage $S$ and the same expected transmission rates $\mathbb{E}_\theta(R_\theta)$, a method which has lower expected distortion $\mathbb{E}_\theta(D_\theta)$ is preferred. Therefore, unlike conventional 2D image/video evaluation methodologies in which 2D RD plots are used for comparison, here we need to compare different interactive compression schemes with 3D Storage-Rate-Distortion (S-R-D) curves. For simplicity from now on we omit the term *expected* and simply call S-R-D.

For 2D images, the lossy compression is usually achieved by quantizing the transform values of the signal and the quantization indices are then coded with a lossless compression algorithm. By changing the Quantization Parameter (QP) of the encoder, different qualities of the content are generated. It is worth noting that since the only free parameter that is adjustable is the QP value, the S-R-D values generate 3D curves rather than surfaces.

## 2.3 Proposed evaluation framework

Performing a formal comparison between 3D curves is a difficult task especially when the curves have different domains as is the case for typical S-R-D curves (see Fig. 2.2). We propose three ways to compare methods based on the constraints imposed by the system.

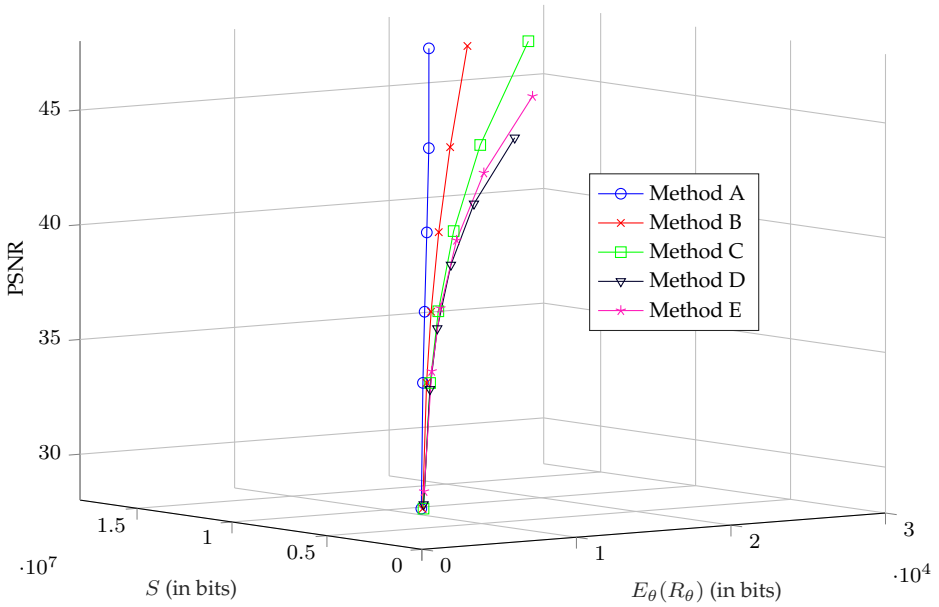



Figure 2.2 – Examples of S-R-D curves. Comparison between different methods is difficult in 3D. We propose three solutions to compare between methods.

### 2.3.1 Average storage and transmission rate saving

Bjontegaard-Delta (BD) rate [8] is a commonly used metric as it performs a fair comparison between two coding methods. More precisely, in this metric, a common PSNR range is determined, and the average bitrate reduction/increase is computed for this PSNR range. We extend the BD measure for storage and transmission rate by projecting S-R-D curves to R-D and S-D planes, and we name them BD-R and BD-S respectively. The BD-S represents the average reduction/increase in the amount of data stored on the server for the same PSNR range while the transmission rate is omitted. BD-R is equivalent to classical BD-R and represents the average bit rate saving/increase in the same PSNR range transmitted to each user while the storage is not taken into account. To evaluate different methods, the pair (BD-R, BD-S) must be considered jointly. The geometrical interpretation corresponding to BD-R (BD-S) is shown in Fig. 2.3 and it is computed based on the projection of the S-R-D 3D curve onto a plane which is perpendicular to the ignored axis $S$ ($R$).

It is worth noting that BD-R and BD-S have not been considered simultaneously so far. For instance, the BD-R and the sum of storage over all stored data are considered in [112] to compare different interactive coding schemes. However, although the storage is considered, the sum of the stored data size may not cover the same range of distortion between different methods, which leads to an unfair comparison.

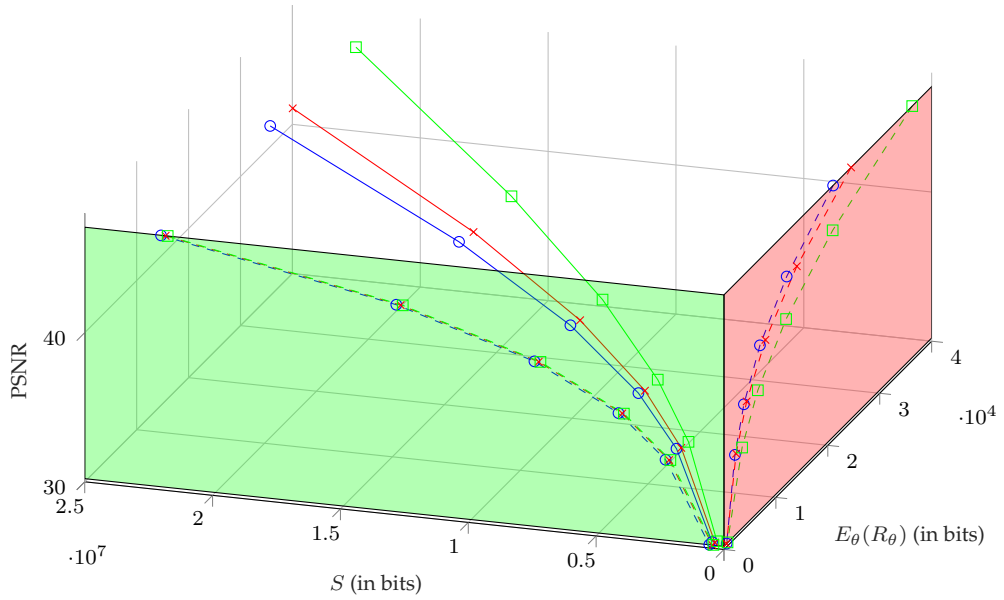



Figure 2.3 – The geometrical interpretation corresponding to BD-R and BD-S. The S-R-D 3D curves are projected on rate-distortion (in red) and storage-distortion (in green) planes, respectively. Then the Bjontegaard-Delta (BD) rate is computed separately on each plane.

Introducing BD-S solves this issue because it considers the same range of PSNR for both methods.

Despite the benefits of (BD-R, BD-S) pairs, two problems can happen with these metrics. The first one is due to the fact that sometimes it is not clear how to compare between pairs. For instance, assume (-20%, 10%) and (15%, -10%) for (BD-R, BD-S) pairs between two different methods (negative values represent percentage saving w.r.t. the reference). Here the former performs better on average in terms of transmission rate and the latter consumes less space on the server for the same PSNR. This problem will be tackled in Section 2.3.3. The second problem is that BD represents the average performance, but in some cases, constant metric bitrate rather than average bitrate must be satisfied. As a solution to this problem, we propose iso values in the next subsection.

### 2.3.2 Iso values

Iso values can be used as an evaluation tool when the system must satisfy some constraints. For instance, the service might be constrained to serve the clients with a minimum distortion level $d$. To choose and compare the coding schemes, we introduce the iso-distorsion values:

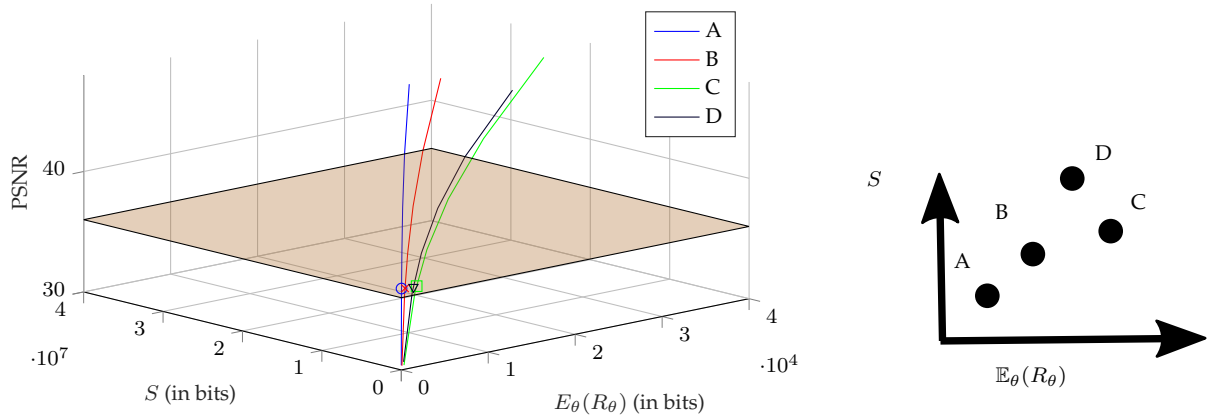
$$\{(S_i, R_i, PSNR_i) | PSNR_i = d \text{ and } i = \text{indices of method}\}$$

where $PSNR_i$ is the PSNR value computed from distortion $D_i$. To compute the isocurve, linear interpolation of the S-R-D curve is used.

Similarly, iso storage (iso transmission rate) can be proposed which shows the trade off between distortion and transmission rate (distortion and storage) at the same storage (transmission rate). De-

pending on what the constraints of the streaming system are (distortion or transmission rate or storage), one can plot iso points and compare methods in iso values domain. This metric can be geometrically interpreted as observing the 3D curves in a plane that is perpendicular to the coordinate axis of the entity chosen as a constraint.

An example of iso distortion values is shown in Fig. 2.4. Comparing method A and B, which have iso distortion values, reveal that method A consumes less bandwidth than method B and it needs lower space on the server to store data. Therefore, method A is preferred over method B. Sometimes iso curves do not help to determine which method is better than the other. For instance, in Fig. 2.4b method C takes lower space on the server and method D performs better in terms of transmission rate.




(a) (b)

Figure 2.4 – Iso distortion metric for the analysis of immersive coding schemes. (a) Geometrical interpretation of the metric can be interpreted as a plane perpendicular to the distortion axis. (b) Iso distortion points: each point corresponds to the R,S performance of a compression method for the same fixed distortion. Therefore, all points have the same distortion.

### 2.3.3 Weighted BD

To solve the ambiguities occurred with iso curves and (BD-R,BD-S) pairs, there is a need to adjust the tradeoff between rate and storage. This can be done by introducing a weighted combination:

$$\mathbb{E}_\theta(R_\theta) + \lambda \cdot S \tag{2.7}$$

where $\lambda$ balances the importance between storage and transmission rate and is determined by application constraints (a high value of $\lambda$ means that storage is more important than transmission rate). As shown in Fig. 2.5a, this mapping yields a projection of the 3D S-R-D curve onto the 2D plane defined by the distortion D axis and the line $S = \lambda \cdot R$. With the definition of (2.7), all points on a line perpendicular to $S = \lambda \cdot R$, are projected on the same value on $\mathbb{E}_\theta(R_\theta) + \lambda \cdot S$ axis. This projection is shown in Fig. 2.5b where the axis corresponding to $\mathbb{E}_\theta(R_\theta) + \lambda \cdot S$ is depicted in blue. Examples of such curves with different $\lambda$s are shown in Fig. 2.5c and Fig. 2.5d. Then, from two such curves, a BD measure can be computed. We call this metric weighted BD for the rest of this dissertation.

The choice of the weighting parameter $\lambda$ is crucial, and may yield opposite conclusions when comparing compression methods (see Figs. 2.5c and 2.5d). We can also give an objective interpretation to the regularization parameter. For instance, (2.7) can result from

$$\alpha \cdot \mathbb{E}_\theta(R_\theta) + \beta \cdot S \tag{2.8}$$

where the regularizers $\alpha, \beta$ can be used to impose another criterion like delay time for each method. For example, if $\beta$ represents the time per byte that it takes to read the data from the hard drive on the server and $\alpha$ represents the inverse of the bandwidth, then (2.8) represents the overall delay time which consists of delay time for the server to read data from its hard drive plus the delay time for the data to be received by the users through the network. By performing BD over this new metric we can compare the delay time between two approaches over the same PSNR range.

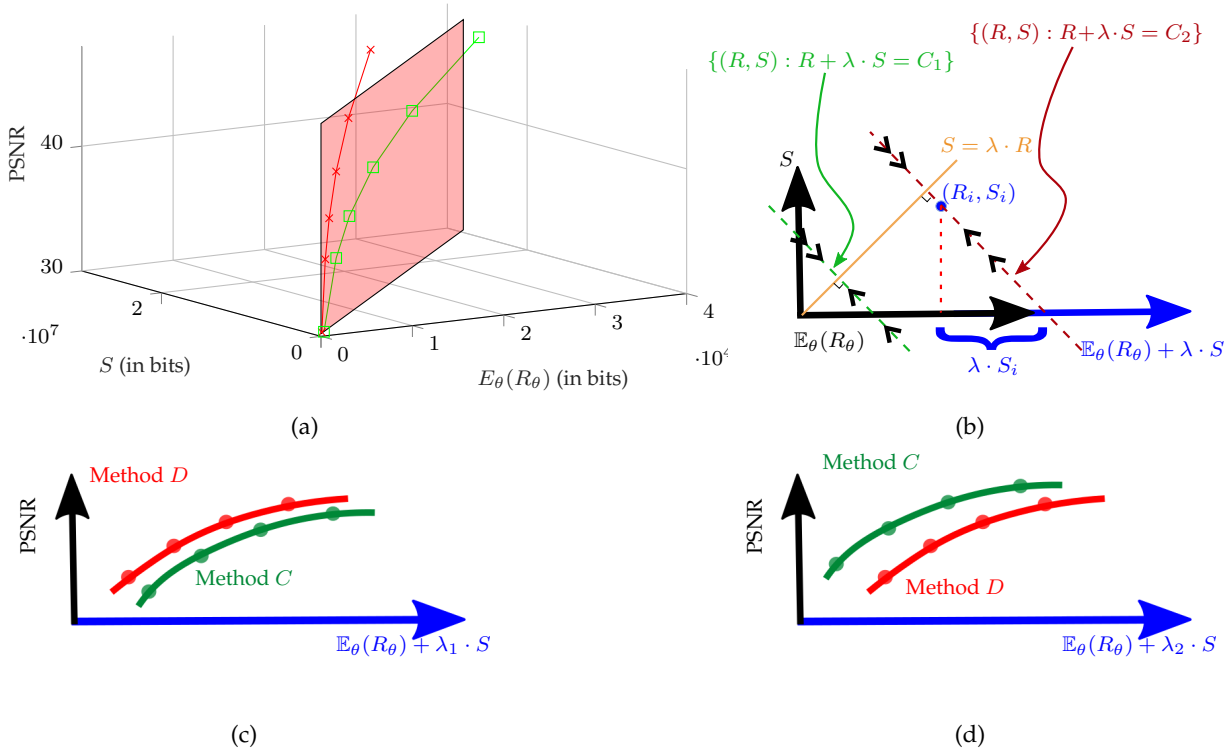



Figure 2.5 – Interpretation of weighted BD metric for the analysis of immersive coding schemes. (a) The plane where S-R-D values are projected on. This plane is defined by the distortion D and the weighted combination (2.7). (b) Projection of the S-R plane onto the new axis $R+\lambda S$ (in blue). (c) and (d) Projected S-R-D curves for different $\lambda$ where $\lambda_1 < \lambda_2$.

## 2.4 Experimental illustration

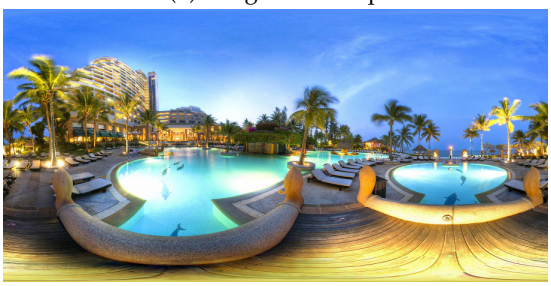
In this section, we illustrate the usefulness of the proposed framework with current solutions proposed for interactive compression of 360° images. Four large size 8960x4480 equirectangular images

from the SUN360 database [106] are considered in our experiments: two from indoor environments and two from outdoor environments. Images are shown in Fig. 2.6.

(a) Image church

(b) Image workshop

(c) Image plaza



(d) Image pool

Figure 2.6 – 360° images used to illustrate the proposed evaluation framework. The first row shows the indoor images and the second row shows the outdoor images.

As for the compression methods, we consider tile-based coding of equirectangular images with three tile sizes (640x320, 1280x640 and 4480x2240) which partition the panoramic image into 14x14, 7x7 and 2x2 tiles respectively. We denote them by *E. 14x14*, *E. 7x7* and *E. 2x2* respectively in the following (*E.* stands for the equirectangular projection because the tiling is defined on the equrectangular image). Each tile is intra-coded independently using High Efficiency Video Coding (HEVC) compression tool [86]. Upon user request, the server sends only the tiles related to the requested viewport. If any tile has been sent already in a previous request, the server avoids sending the tile again. We also implemented the cube map representation with two different face resolutions: 2560x2560 and 3008x3008 (the sum of six cube faces of size 2560x2560 is almost equal to the panoramic image size). We denote them by *C. lower* and *C. higher* respectively. Each cube face is encoded separately with HEVC intra-coding. Then, the server sends only the cubes faces which participate in the production of the requested viewport.

Our proposed evaluation framework does not rely on a specific user navigation model: it simply needs a set of user head position no matter how they are generated. We restrict our experiments to images, but one can perform similarly with videos. Since there is no long duration navigation dataset with large number of users for this image database, we simulate close-to-reality users' navigation using the probabilistic model explained in the following (Section 2.4.1).

### 2.4.1 Simulation of user navigation for 360° contents

In particular, it is observed in [110] that users more frequently view areas around the equator (up to latitude $\pm\pi/6$) than the poles. Moreover, when users are moving their head by changing the longitude and latitude angles, it is more likely for them to continue their previous direction rather than changing the head movement to the opposite direction.

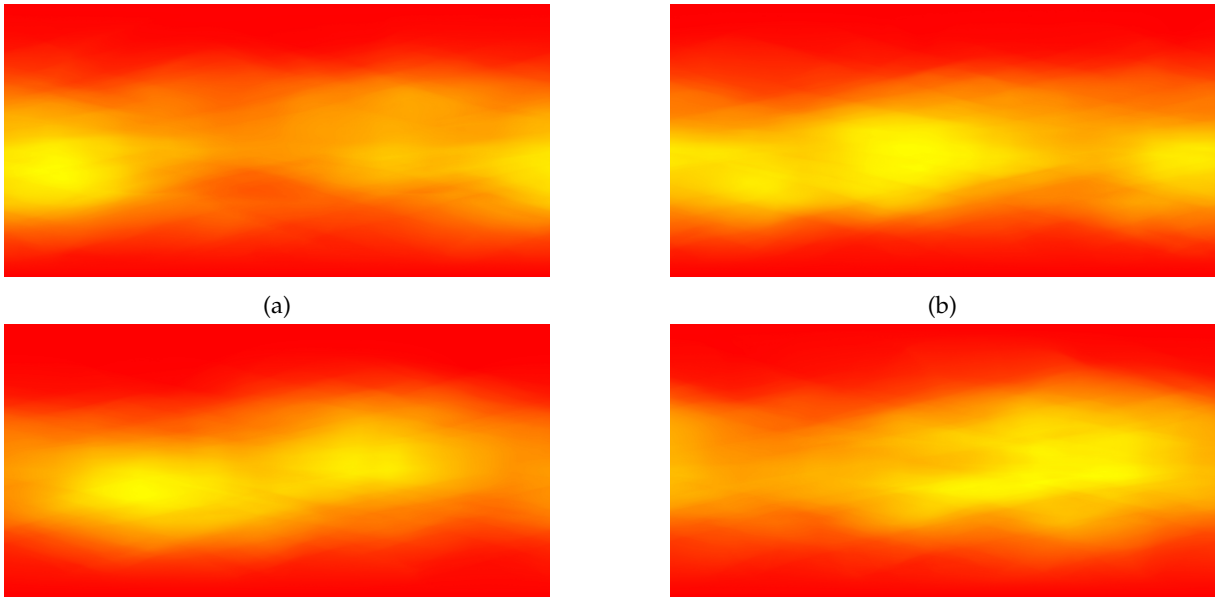
We simulate the user's navigation using a markov chain probabilistic model of order one. For the first request, the user's viewing direction is initialized by sampling from a uniform distribution in the interval $[-\pi/6, \pi/6]$ and $[-\pi, \pi]$ for latitude and longitude values, respectively. We then continue the navigation by defining three probabilities $p_1, p_2, p_3$, and a step $\Delta$ per longitude and latitude random variables. $p_1$ is the probability of continuing the previous movement, $p_2$ is the probability of stopping at the current state and $p_3$ is the probability of changing movement to the opposite direction while $p_1 > p_3$. Note that $\Delta$ can be positive or negative depending on in which direction the user is continuing his previous movement. For example if for the latitude (longitude) angle, the previous direction is towards increasing latitude (longitude), then $\Delta^{lat}$ ($\Delta^{lon}$) is positive, otherwise, it is negative. We then sample from this discrete distribution for each longitude and latitude value. If the sample is generated from $p_1$ we continue the previous movement by adding $\Delta$ to the previous value. If it is sampled from $p_2$, then the current value would be equal to the previous value. If the sample is generated from $p_3$ we first change the sign of step direction $\Delta \leftarrow -\Delta$ to change the previous movement direction and then add it to the previous value.

We chose probabilities with trial and error to make navigation of the users look natural. For that, for longitude values we used $(p_1^{lon}, p_2^{lon}, p_3^{lon}) = (0.983, 0.01, 0.007)$ with step $\Delta^{lon} = 0.5°$ and for latitude values we used $(p_1^{lat}, p_2^{lat}, p_3^{lat}) = (0.405, 0.59, 0.005)$ with step $\Delta^{lat} = 0.3°$. We generate navigation for 20 users where each has 600 requests (in total 12000 requests per image). The corresponding color-coded heatmaps are shown in Fig. 2.7 in which colors represent the aggregated frequency of each pixel (in the equirectangular domain) to be inside the viewport for the whole users' navigations. One can see that the proposed simulated navigations are similar to the real user's navigation collected in [19]. Finally, for each image, $\mathbb{E}_\theta(R_\theta), \mathbb{E}_\theta(D_\theta)$ are computed by averaging over all users' requests.

### 2.4.2 Storage and transmission rate saving

Intuitively, approaches with smaller tile size should perform better in terms of transmission, but be less efficient in terms of storage. This is because small tiles bring more flexibility to send only the parts which are requested, but larger tiles result in better encoding performance. The proposed (BD-R, BD-S) criterion allows to assess quantitatively to what extent the tile size influences the tradeoff between transmission rate and storage. For instance, the first two rows of Table 2.1 give the (BD-R, BD-S) pair averaged over all images, where 7x7 tiling is used as a reference (positive values show they perform worse than 7x7 tiling and negative values show they perform better). Interestingly, we observe that when tiles are intra-coded the tile size influences much more the transmission rate (from ∼-6% to ∼32%) than the storage (∼-4% to ∼10%).

Our proposed S-R-D characterization of the performance of an immersive coding scheme can also be plotted in terms of joint R-D and S-D curves, as shown in Fig. 2.8 for two images church and plaza.



(a) (b)

(c) (d)

Figure 2.7 – Color-coded heatmaps of the frequency of pixels observed in the user navigation simulated by the proposed probabilistic model. Colors represent aggregated frequency of each pixel to be inside the viewport. The frequency is increasing from red to yellow, i.e., yellow pixels participate more in viewport generation. (a) Image church heatmap. (b) Image workshop heatmap. (c) Image plaza heatmap. (d) Image pool heatmap.

In this representation, we observe that the cube map approach saturates at high bit rates in both R-D and S-D curves. This can be explained by the fact that the original images are stored in equirectangular format, and there is a loss in quality in the cube map approach, due to the extra projection from the equirectangular to the cube map representation.

### 2.4.3 Iso distortion values

The proposed iso distortion values allow to compare schemes at a given operating point and can help guiding the design of the compression scheme. For instance, one can evaluate the influence of the face resolution on the performance of the cube map approach in Fig. 2.9 for two images church and

Table 2.1 – BD measures averaged over all four images. 7x7 tiling is used as a reference. The first 2 rows are (BD-R, BD-S) pair. The last three rows are weighted BD computed over the same PSNR range for (2.7).

| | E. 14x14 | E. 2x2 | C. lower | C. higher |
|---|---|---|---|---|
| BD-R | -5.67 % | 27.71 % | 28.74 % | 32.03 % |
| BD-S | 9.84 % | -3.94 % | 2.97 % | 5.68 % |
| $\lambda = 0.01$ | 8.10 % | -0.52 % | 5.77 % | 8.55 % |
| $\lambda = 1e^{-3}$ | 1.21 % | 13.37 % | 17.10 % | 20.14 % |
| $\lambda = 1e^{-4}$ | - 4.52 % | 25.28 % | 26.77 % | 30.02 % |

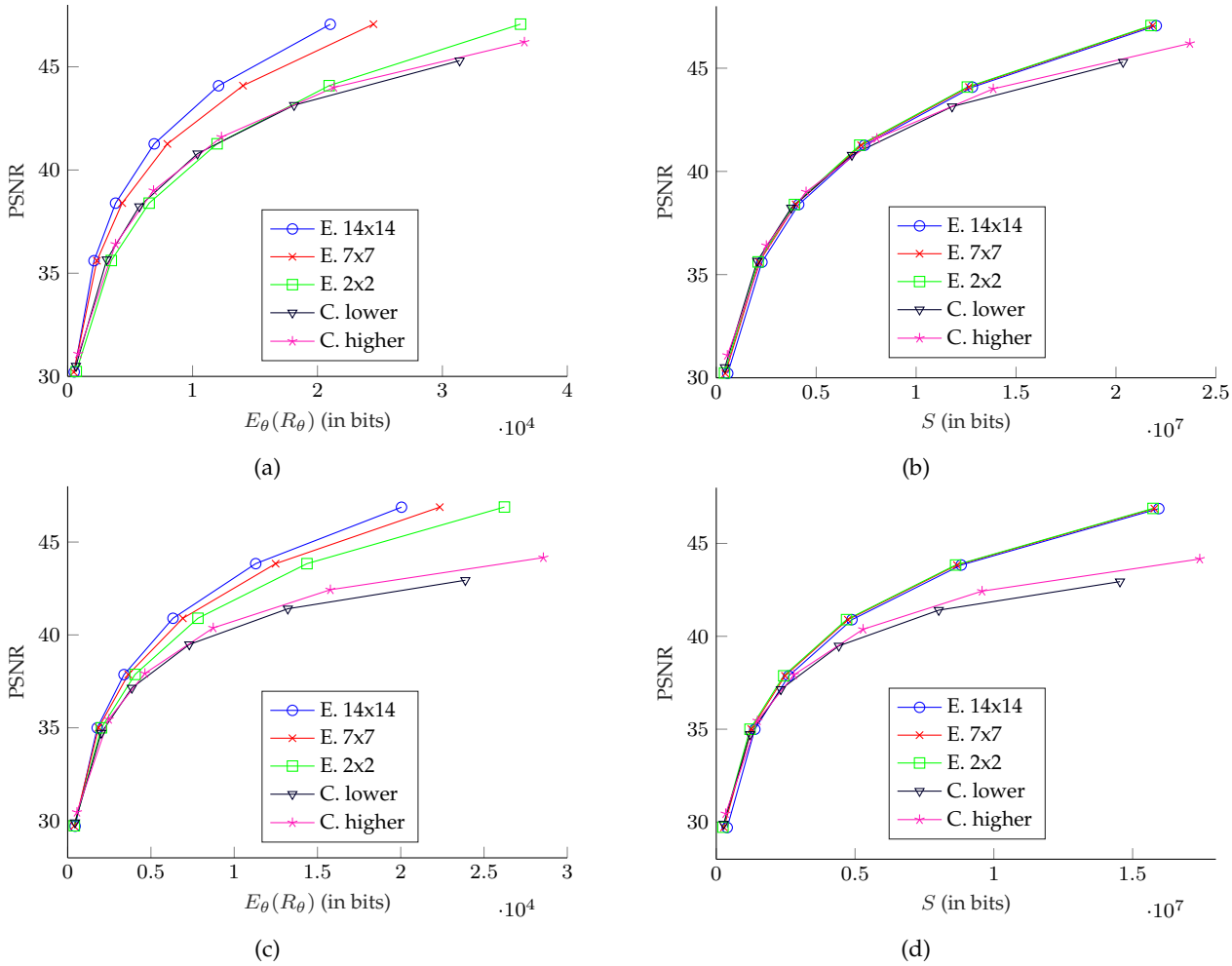



Figure 2.8 – R-D and S-D curves of interactive compression using tile-based approach in which each tile is intra-coded with HEVC. First row shows the results for image church and second row shows the result for image plaza. (a) R-D curves (church). (b) S-D curves (church). (c) R-D curves (plaza). (d) S-D curves (plaza).

plaza. At low PSNR (33-34 dB), lower face resolution achieves a smaller transmission rate and storage than higher face resolution, whereas, at high PSNR (39 dB), higher face resolution performs better in both storage and transmission rate. This highlights the strategy that for cube map projection, instead of playing with QP values, it is better to change the resolution of the cube faces. More precisely, we should consider a multi-resolution strategy where at lower bitrates (low PSNR values) cube maps with lower face resolution perform better in terms of both storage and transmission rate. At high bitrates (higher resolution) by increasing the resolution of the cube faces we can expect better performance again in both transmission rate and storage.

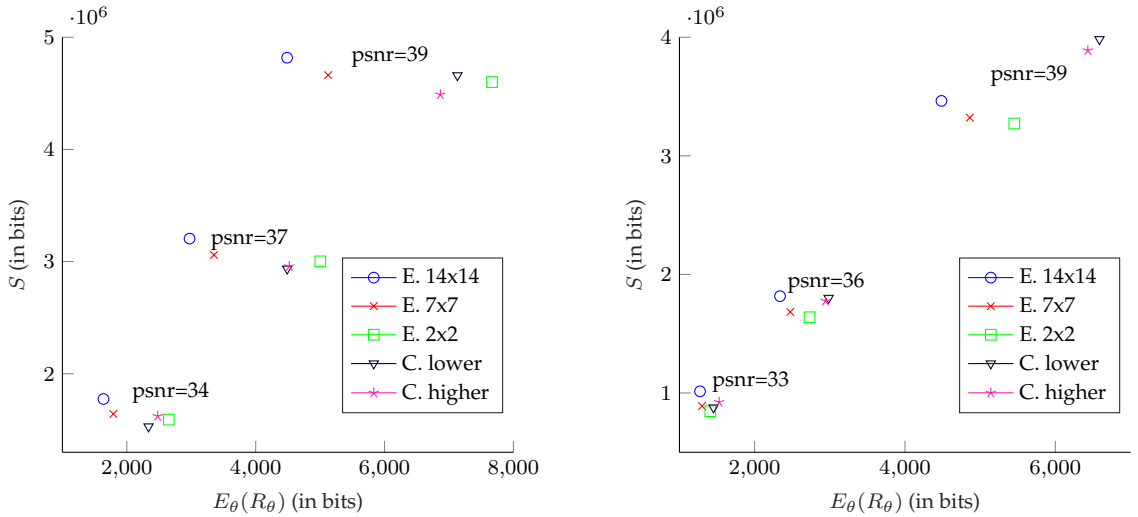



Figure 2.9 – Iso distortion values for interactive compression of images using tile-based approach while each tile is intra-coded with HEVC. (a) Image church. (b) Image plaza

### 2.4.4 Weighted BD

Weighted BD was introduced in this dissertation to deal with cases where the comparison between two coding schemes is not straightforward. This occurs in particular, when one scheme achieves a lower transmission rate but at the cost of higher storage rate with respect to another scheme. In contrast, the new weighted criterion (2.7) used to compute weighted BD can help to discriminate between these schemes by fixing $\lambda$. Depending on the application, this weighting coefficient balances the cost between storage and transmission rate, as shown in the last three rows of Table 2.1. For example, in our experiment the storage is about 700-1000 times larger than the transmission rate. Imposing $\lambda = 1e^{-3}$ means that storage and transmission rates are both of the same importance. In this scenario, 7x7 tiling performs better than the other ones. In other words, when storage and transmission rates are both of the same importance, tiling the equirectangular image into 7x7 tiles brings a good compromise between storage and transmission rate. For smaller $\lambda = 1e^{-4}$, transmission is more important and 14x14 tiling should be chosen. Weighted BD for different $\lambda$s are plotted in Fig. 2.10, where the lowest weighted BD corresponds to the best method. More precisely, when $\lambda < 0.0008$, 14x14 tiling achieves the best performance, whereas the reference 7x7 tiling is better in $0.0008 \leq \lambda < 0.009$ and 2x2 tiling should be chosen for $\lambda \geq 0.009$.

## 2.5 Summary

In this chapter, we proposed an evaluation framework to compare different interactive compression schemes. First, we reformulate the compression problem of interactive coding schemes (previously defined in Section 1.2.1) where 2D images are used to represent the interactive imaging modality (projection of 360° images or texture map of 3D models). More precisely, we considered storage and transmis-

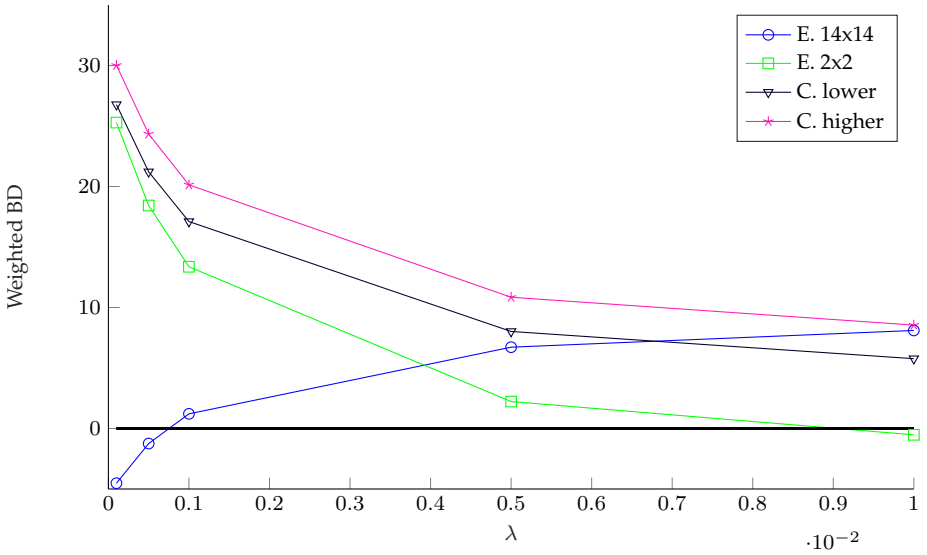



Figure 2.10 – Weighted BD for different $\lambda$s when church is coded using tile-based approach while each tile is intra-coded with HEVC.

sion rate along with distortion sensed on the user side altogether, as all of them affect decision making in the streaming system. Three different metrics were proposed: pair of (BD-R, BD-S), iso values, and weighted BD. The (BD-R, BD-S) pair represents the average bit rate saving/loss in the transmission rate and storage rate for the same range of PSNR qualities displayed to the users. The iso values are useful when we want to evaluate the performance under fixed constraints rather than average performance. However, ambiguity in decision can happen in both approaches. The weighted BD solves the ambiguity that can happen in the previous two metrics. Finally, we illustrated the proposed framework experimentally using classical tile-based approach used for 360° images.

CHAPTER 3

# CODING FOR GENERIC 2D REPRESENTATIONS OF INTERACTIVE IMAGE MODALITIES

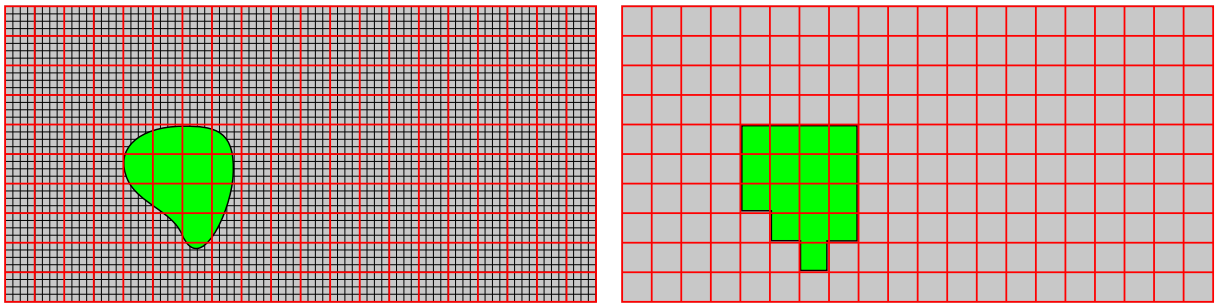
A generic representation of many interactive image modalities can be a 2D image. For instance, it was shown in the previous chapter (Section 2.2) that 360° images and texture maps of any arbitrary 3D models are projected on a 2D image to represent the content. In this chapter we propose a new interactive compression scheme for this 2D representation of interactive image modalities. Therefore, it can handle different kinds of interactive image modality that is projected on a 2D image. For the rest of this dissertation we assume that the interactive media is represented in 2D.

An interactive coding scheme must fulfill two characteristics: efficient compression of data, to lower the storage cost, and random access ability to extract part of the compressed stream requested by the user (for reducing the transmission rate). For efficient compression, classical schemes perform a prediction by a series of references that have been pre-defined and compressed. This introduces a coding and decoding order of the data which contrasts with the spirit of random accessibility. However, it is proven theoretically that with the SMRA approach and incremental codes we can relax this fixed ordering [70, 28, 30] (see Section 1.2.2.2 for more details). In this chapter, we propose solutions to put these theoretical results into practice. This leads to a generic coding solution based on incremental codes implemented by rate-adaptive channel codes. This scheme encodes the image while adapting to any user's request and provides efficient coding that is flexible in extracting data depending on the available information at the decoder. Therefore, when the user requests part of the image, only the information which is needed to be displayed at the user's side is transmitted. This allows achieving the same transmission rate as if the requests were already known at the encoder (the oracle coder).

## 3.1 Modeling interactivity through a navigation graph

Following the formal definition of the interactive coding given in Section 1.2.1, interactivity in these 2D images can be modeled by a *navigation graph*. To do so, first the granularity of the data has to be defined. Usually, in all compression schemes, the granularity is at the block level and as a result the image is split into non-overlapping blocks. Therefore, a *block* corresponds to the *smallest entity (source)* that can be requested, and it is represented by a node in the graph.

Depending on the application, the interactive system authorizes the user to ask a given set of *permitted requests* where *each request* corresponds to a *set of blocks*. For instance, as shown in Fig. 3.1, when a user observes a set of pixels, this observation corresponds to a request which consists of a set of blocks. One way to model this permitted set of requests is to first order each request. This is compliant with the *sequential* way to classically serve the requests. Then, the navigation graph is built such that all sequences of permitted requested can be represented by paths in the graph. In this case, the graph nodes represent the blocks in the image, and the edges between the nodes are defined such that starting from a block in the image one can traverse all requested blocks.



(a) (b)

Figure 3.1 – In interactive 2D image modalities, a user observes a set of pixels which corresponds to a request of a set of blocks. Image pixels (in black) are grouped into non-overlapping blocks (in red). (a) The observed area by a user (in green). (b) The corresponding request (in green), which consists of a set of blocks.

An example of a navigation graph is shown in Fig. 3.2. It is worth noting that the blocks of a request are not necessary adjacent, and a request may consist of several disconnected regions as shown in the figure. As explained, the edges between the nodes should make it possible to cross all the nodes representing the requested blocks. For the example shown in Fig. 3.2, this implies that there exist some links (edges) between some blocks (nodes) of these separated connected blocks (nodes). Therefore, the connection between blocks is not necessary a 4-connected neighboring. For this example, edges such as links A, B, and C are common in the texture map of 3D models because although these parts are disconnected on the 2D image, they are connected on the 3D surface (this will be explained later in Section 4.3.1). Edges like D and E occur in spherical images that are mapped to the 2D equirectangular plane, again because although the two vertical boundaries are far apart in 2D, they are actually connected on the sphere.

Navigation between the requested blocks can only start from specific blocks called *access blocks* whose indices are kept in a set $\mathbb{A}$. Any request of a set of blocks that do not contain an access block inside itself is an invalid request. Finally, for each request, starting from an access block, the remaining requested blocks are traversed one after the other following the *neighborhood* defined for each block within the navigation graph. More precisely, the *neighborhood* $\mathbb{J}_{(k)}$ represents the set of *possible* blocks from which it is possible to navigate directly to the current block $k$ (the notations and definitions are summarized in Table 1.1 of Chapter 1). Based on these assumptions, our user's navigation definition is compliant with i) the theoretical definition introduced in Chapter 1, and ii) the practical user's navigation that occur in

interactive image visualization.

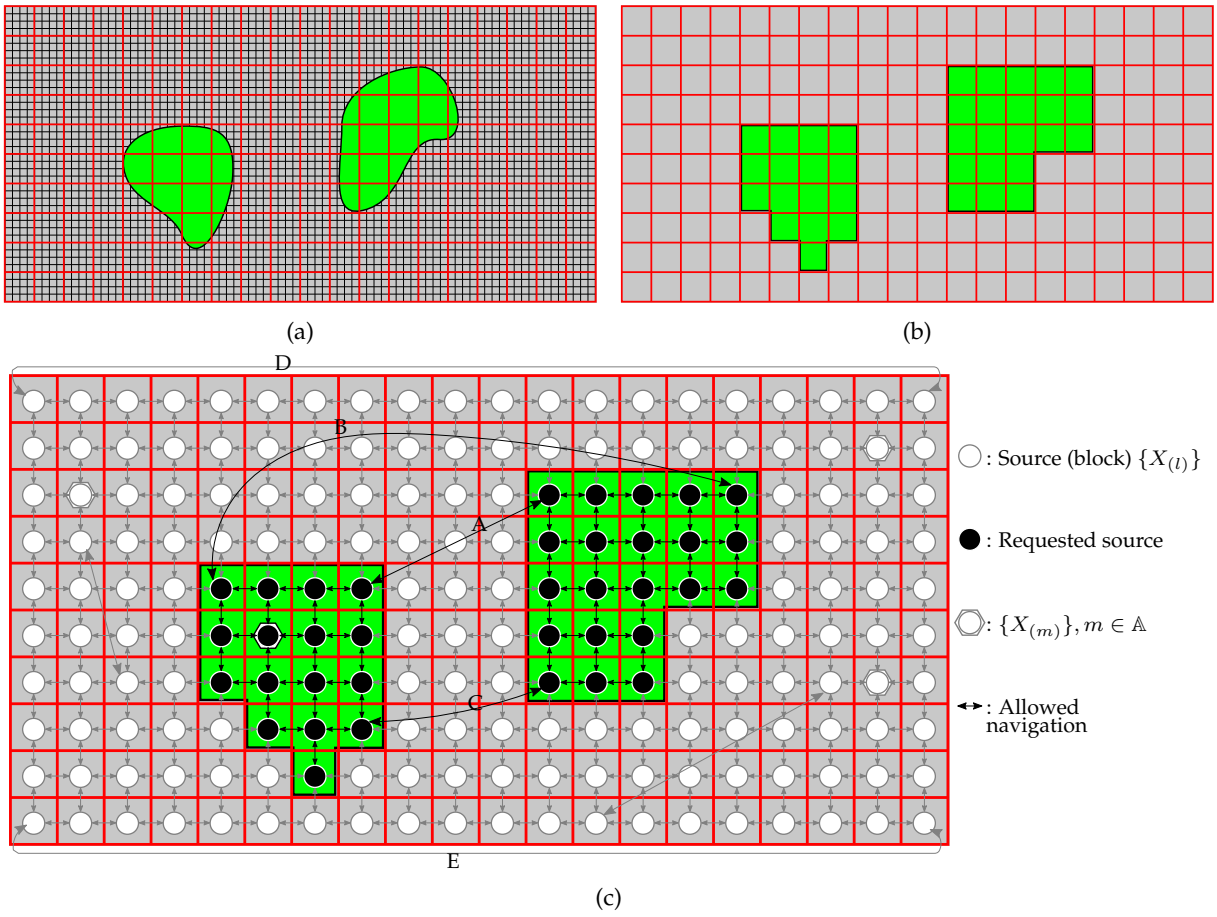



Figure 3.2 – Navigation graph for interactive 2D image modalities. A request (in green) may consist of one or several connected components. (a) Observed region in pixel domain. (b) Corresponding requested blocks. (c) The navigation graph $\mathbb{G}^{nav}$ where the image blocks are represented as source.

## 3.2 Existing schemes to compress interactive 2D image modalities

Whenever the decoder wants to decode a block, one or a combination of several previous decoded blocks can be used to create a prediction of the current block to capture the correlations between blocks. These predictions are used as SI sources which help to store and transmit the blocks more compactly. The better the prediction is, the more compact representation can be achieved. Therefore, the choice of scanning order impacts the performance of the interactive system.

In conventional coders, this correlation is captured thanks to the *predictive coding* scheme explained in Section 1.2.2.1. Let source $X$ model the image block to be compressed, and $Y$ model the *prediction* of $X$ using another block(s). Assuming $X$ and $Y$ are independent and identically-distributed (i.i.d.) sources, with the help of prediction $Y$, the achievable compression rate is the conditional entropy $H(X|Y)$ [21], which is smaller than the rate $H(X)$ achieved when no SI is available.

Despite the efficiency of predictive coding, this method is not suitable for random access compression because it imposes a *fixed pre-defined scanning order* of the blocks. To allow random access, or equivalently separable bitstreams defined in (2.2) (see Section 2.2), state-of-the-art coders divide the 2D image $I$ into sub-elements. Finally, each sub-element is encoded independently with $f$, the encoding function of a classical 2D video codec, such as HEVC [104].

For instance, in the tile-based approach the image is partitioned into several tiles in which each tile consists of multiple blocks located in a particular rectangular area of the image. Each tile is compressed separately. More precisely, for each tile an encoding graph is created which imposes the ordering for encoding/decoding of the blocks within a tile. Note that the encoding graph differs from the navigation graph and determines which of the already encoded/decoded blocks are used to generate a prediction of the current block as SI. An example of tile-based approach is shown in Fig. 3.3 where the tiles are depicted by blue rectangles. In each tile, there is a *reference block* (red block with yellow border) that is coded independently of the other blocks as the starting block. Remaining blocks are encoded/decoded in a fixed order depicted by numbers within circles. Then, upon request, all tiles containing part of the requested green area are sent and decoded. The disadvantages of this method is that the correlation between the tiles is not exploited and also potentially more blocks are needed to be sent than required.

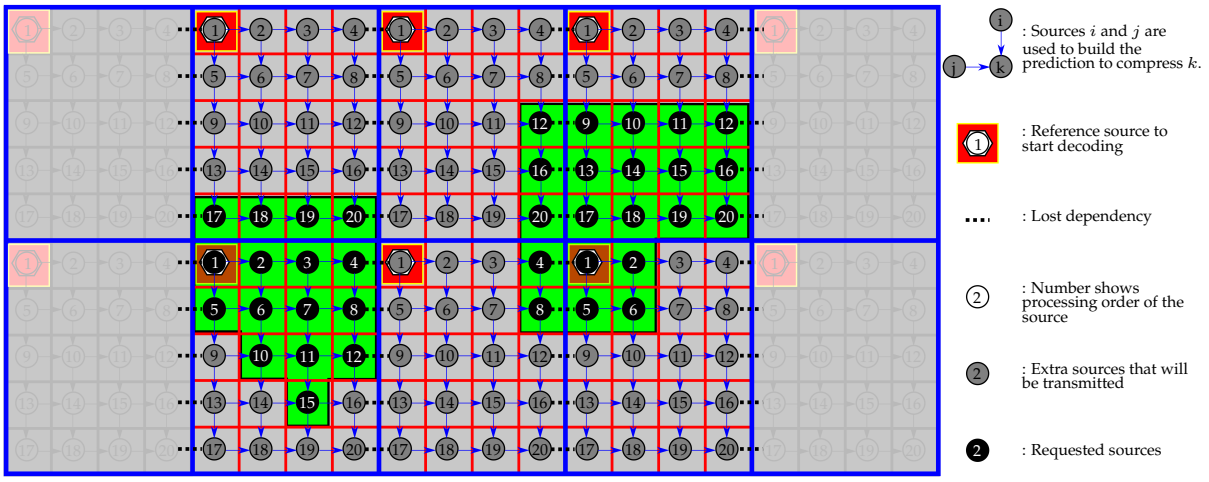



Figure 3.3 – Tile-based approach for compression of interactive 2D image modalities and the corresponding encoding graph $\mathbb{G}^{enc}$. Tiles are shown in blue and each one is coded separately. For the tiles that are transmitted to the user, the blocks are shown in red. The non-transmitted tiles and their corresponding subgraph are shown transparent. The filled red block with yellow border corresponds to the reference blocks. The green region is the requested blocks of the 2D image. The encoding/decoding order of the blocks are shown with numbers within circles. Any tile that part of it is requested must be transmitted. The arrows depict the edges of the encoding graph. More blocks are sent than needed. Moreover, dependencies between tiles are not exploited in the compression.

Considering the works presented for interactive 2D image modalities in Section 1.3, one can notice that all state-of-the-art approaches use the same predictive coding function $f$ (like HEVC), and they only differ in the mapping $m$ that maps the data into 2D images. For example, for 360° images different variants of QER (like pyramid) and tile-based approaches are proposed, or for texture mapping of the

meshes, different mapping are optimized to have less background area (see Fig. 1.15 (c)). Therefore, these methods attempt to solve

$$\begin{aligned} &\min_{m} \lambda S + \mathbb{E}_\theta(R_\theta). \\ &\text{such that } \mathbb{E}_\theta(D_\theta) \leq \delta \end{aligned} \tag{3.1}$$

where the storage $S$, the request $\theta$, the transmission rate $R_\theta$, and the distortion $D_\theta$ have been defined in Section 2.2. $\mathbb{E}_\theta$ is the expectation over all users' requests. The optimization is performed over the mapping function (defined in (2.1)). Ideally, to achieve the oracle compression rate of $R^{oracle}$, an interactive system should be able to send only the blocks that a user requests while the redundancies between blocks are captured. However, these predictive coding schemes either cannot achieve $R^{oracle}$ or even if they can, they will sacrifice the storage size with a huge overhead cost (exhaustive storage).

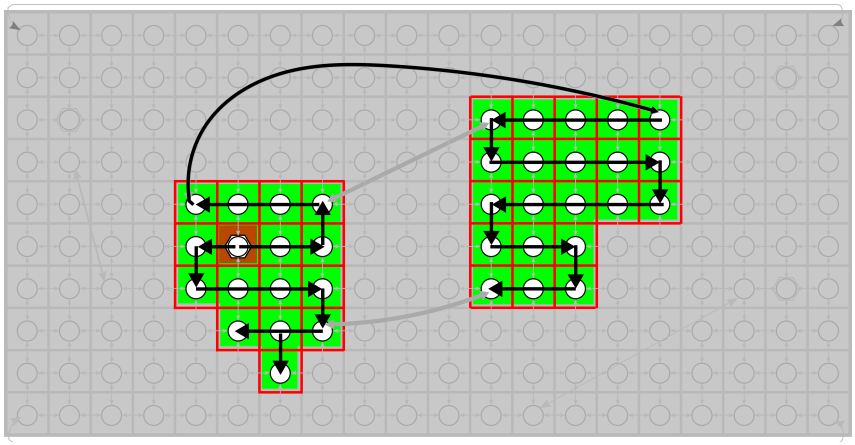
## 3.3 Principle and overview of the proposed interactive coding scheme

In contrast to these classical schemes, here we propose a new 2D interactive coding scheme in which the fixed scanning order constraint is relaxed. In this case, the encoding graph is equal to the navigation graph (see Section 1.2.2.2). Therefore, as long as the scanning order respects the navigation graph, the system can decode the blocks. Fig. 3.4 shows two examples of scanning blocks. This coder differs from state-of-the-art methods in several ways. First, our goal is to achieve the same transmission rate as the oracle scheme $R^{oracle}$ which knows the request in advance (i.e., upon encoding). Second, the storage should not be significantly penalized. Finally, to achieve this goal, the encoding and decoding functions $(f, g_\theta, h_\theta)$ are optimized for a given mapping $m$. So, our method attempts to solve

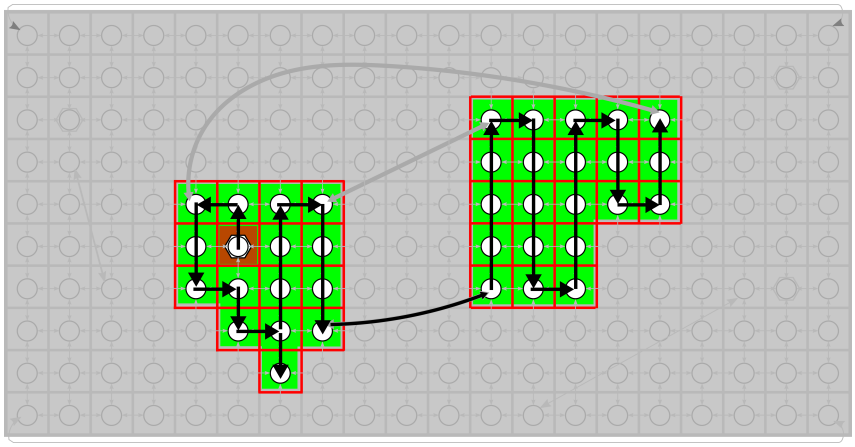
$$\begin{aligned} &\min_{f, g_\theta, h_\theta} S \\ &\text{such that } \mathbb{E}_\theta(D_\theta) \leq \delta \\ &\qquad\qquad \mathbb{E}_\theta(R_\theta) = R^{oracle}. \end{aligned} \tag{3.2}$$

Here, similar to the predictive coding approach, blocks represent sources that we want to encode, but a block (rectangle with a red border in Fig. 3.4) can be decoded whatever neighboring block(s) are available at the decoder. Then, for each user's request, only the needed blocks are transmitted, and the correlation between the blocks is *optimally* taken into account. Therefore, our scheme can reach the oracle compression rate $R^{oracle}$, while maintaining the constraint that the request is not known in advance.

The aforementioned optimality can be explained from [70], which studies lossless coding of a source with several SI sources available at the encoder. The source is encoded in a single bitstream and once it is known which SI source is available at the decoder the corresponding part of the compressed bitstream is extracted. More precisely, let $X_{(k)}$ model the image block $k$ to be compressed and each $Y_j$ correspond to the prediction for $X_{(k)}$ using one or a combination of the neighboring blocks whose indices are in the set $\mathbb{J}_{(k)}$ representing the set of neighboring blocks (see Section 1.2.1 for the formal definition of $\mathbb{J}_{(k)}$). In [70], it is shown that *incremental* coding *optimally* exploits the dependence between the sources because the extraction can be made at rate $H(X_{(k)}|Y_j)$, *i.e.*, at the same rate as if the encoder knows in advance the index of the SI.



(a)



(b)

Figure 3.4 – Examples of different allowed block scannings in the proposed method for the navigation graph given in Fig. 3.2. The decoder can decode each block with any scanning order that respects the navigation graph. Transmitted blocks are shown in red (including the filled one that corresponds to the access block). Compared to the tile-based approach, for the same request, the proposed scheme reduces the amount of information sent but not displayed. Moreover, the correlation among the blocks is still taken into account with our coder.

## 3.4 Detailed description of the proposed interactive coding scheme

We now detail our interactive coding scheme. We first present the overall architecture, and then we explain the strategy proposed to start the decoding process (first block). We then detail the practical implementation of the incremental entropy coder and its integration within an interactive image coder in Section 3.4.3. Finally, we formulate the scanning order problem.

### 3.4.1 Proposed architecture

The encoder first splits the input image into several blocks $X_{(k)}$. For each block, several predictions are computed. There is one prediction per possible set of neighboring blocks available at the decoder.

We will explain how these predictions are produced for different modalities in Chapter 4. For some of these blocks, the so-called *access blocks* ($\{X_{(a)}\}_{a\in\mathbb{A}}$), i.e., an empty prediction, is added to the prediction set. These access blocks can be decoded independently to start the decoding process. These blocks need to be carefully located to ensure that any possible request of the user can be decoded, but their number should not be too high to minimize the storage. Then, each block $X_{(k)}$ is encoded with an incremental coder as a function of different predictions. This incremental coder, together with the quantizer and the transform must be carefully integrated in the image scheme to avoid the error propagation problem (see Section 3.4.3).

At the decoder, given a requested viewport, one access block is first decoded. Then the remaining blocks are sequentially scanned one after the other. This block decoding order should be optimized to reach a better transmission efficiency. Each block is consecutively decoded by first generating a prediction based on the already decoded neighboring blocks. Then, the necessary bitstream is extracted, and the incremental decoder is executed (see Section 3.4.3).

### 3.4.2 The access blocks

Access blocks, $\{X_{(a)}\}_{a\in\mathbb{A}}$, are blocks that are coded with an incremental coder as the other blocks, but can be decoded independently of the others *if necessary*. They act similar to reference blocks in predictive coding (red shaded blocks with yellow borders in Fig. 3.3) to serve as the starting point for the decoding process, but only if needed. If a block $X_{(a)}$ is an access block, its corresponding set of prediction is complemented by a $Y_{\emptyset}$, being an empty prediction. Therefore, this block can be decoded independently or as a function of the neighboring blocks if they are already decoded. In other words, an access block $X_{(a)}$ is stored with a rate $H(X_{(a)})$ but can be transmitted at the proper rate, i.e., $H(X_{(a)}|Y_j)$ if a prediction can be generated for it from the already decoded blocks (meaning that it is not the first block to be decoded) and $H(X_{(a)})$ if the block is the first block to be decoded.

Since access blocks are more costly to store than other blocks, their number should be as limited as possible. On the other hand, to avoid sending unrequested blocks and therefore minimize the transmission rate, they must be located such that with any request of users (for instance, rotation angles $\theta$ of the HMD device), there exists at least one access block belonging to the mapped visible region $(m\circ\gamma)(\theta)$ (see Section 2.2). The problem is thus formulated as follows:

$$\min_{\mathbb{A}} |\mathbb{A}| \text{ while } \forall\theta,\ \exists\, k\in\mathbb{A} \text{ s.t. } X_k\in(m\circ\gamma)(\theta). \tag{3.3}$$

where $|\mathbb{A}|$ represents the cardinality of the set $\mathbb{A}$. A solution to this problem is proposed in Section 4.1.1 for 360° images and in Section 4.3.2 for texture maps of 3D models.

### 3.4.3 Practical implementation of incremental source code

A practical solution to implement incremental codes can be achieved by rate-adaptive channel codes such as LDPCA codes [97]. Channel codes have been used in other interactive coding schemes such as stream-switching [15, 16] where DSC is used to merge multiple SI frames. More precisely, and similar to our context, a set of SI frames is computed at the encoder and only one of them is available at the

decoder. However, the current frame is encoded by exploiting the worst-case correlation between a set of potential SI frames and the target frame using compound conditional coding (see Section 1.2.2.2 for more details). Here, thanks to the theoretical work obtained in [70] for the incremental code, we propose to store the data with respect to the worst-correlated prediction (or equivalently SI), but transmit optimally by extracting the amount of data required to decode based on what is available at the decoder.

The proposed incremental coding scheme for encoding and decoding a block $X$ is depicted in Fig. 3.5. The input block $X$ is first transformed, for example, with Discrete Cosine Transform (DCT), and then quantized. This results in a signal $\bar{X}$ that is split into several bitplanes $\bar{X}^1, \ldots, \bar{X}^P$. We now assume that the predictions $Y_1, \ldots, Y_N$ are sorted from the best to the worst, i.e., $Y_1$ is more correlated with $X$ than $Y_N$. These predictions are also transformed and quantized resulting to $\bar{Y}_1, \ldots, \bar{Y}_N$. Note that doing so, encoder and decoder are consistent, which avoids the error propagation phenomenon.

Each bitplane $\bar{X}^p$ is encoded thanks to a binary rate adaptive channel coder, with $\bar{Y}_1, \ldots, \bar{Y}_N$ as SI sequences. It is worth noting that since the correlations between the source and its SI sources are known in advance at the encoder, the symbol-based ($Q$-ary) and binary-based rate-adaptive coder (with bitplane extraction) perform equally [103]. However binary-based encoding is less complex than its symbol-based version. This generates $N$ bitstreams $(\boldsymbol{b}_1^p, \ldots, \boldsymbol{b}_N^p)$ that are stored on the server. For decoding of $X$, if the prediction $Y_j$ is available, the bitstreams that need to be transmitted are $(\boldsymbol{b}_1^p, \ldots, \boldsymbol{b}_j^p)$. It is noticeable that a bitstream is not only dedicated to one particular prediction, but can serve the decoding of several predictions. This is the reason why the storage cost remains limited. Indeed, the stored bitstream does not depend on the number of possible SI sources $N$ but only on the correlation of the worst SI source.

For the access blocks, an additional empty prediction $Y_\emptyset$ (a source of zero entropy) is considered as possible SI source. Then, the encoding process is identical, which means that on top of $(\boldsymbol{b}_1^p, \ldots, \boldsymbol{b}_N^p)$, a bitstream $\boldsymbol{b}_\emptyset^p$ is added to complement the decoding in the case where the block has to be decoded independently. Note that the bitstream $\boldsymbol{b}_\emptyset^p$ serves as a complement of the other bitstreams in case of independent decoding, which means that $\sum_{j=1}^{N} |\boldsymbol{b}_j^p| + |\boldsymbol{b}_\emptyset^p| = H(\bar{X}^p)$.

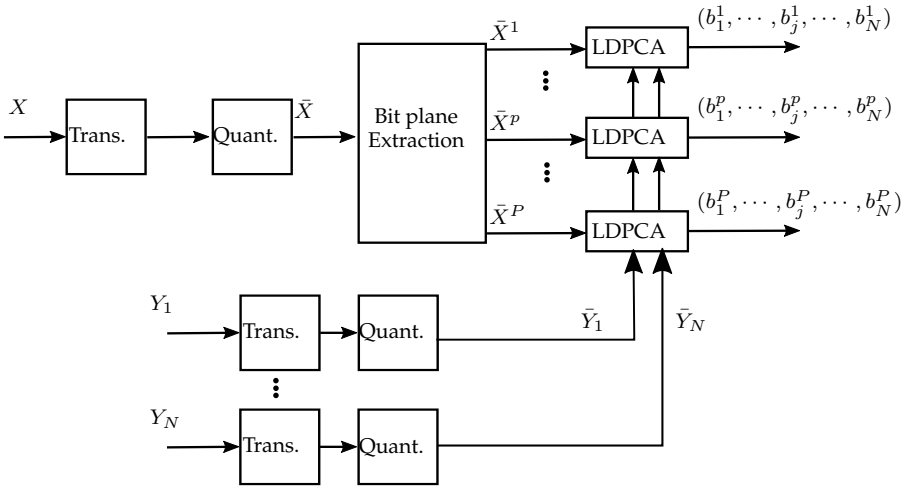
### 3.4.4 Decoding order and navigation

Blocks are decoded one after the other until all blocks in the requested area are processed. The choice of the decoding order is important since it impacts the quality of the predictions that are made, and therefore the transmission rate. More formally, let $\mathbb{H}(\theta)$ denote indices of the blocks in the user's request $\theta$, i.e.,

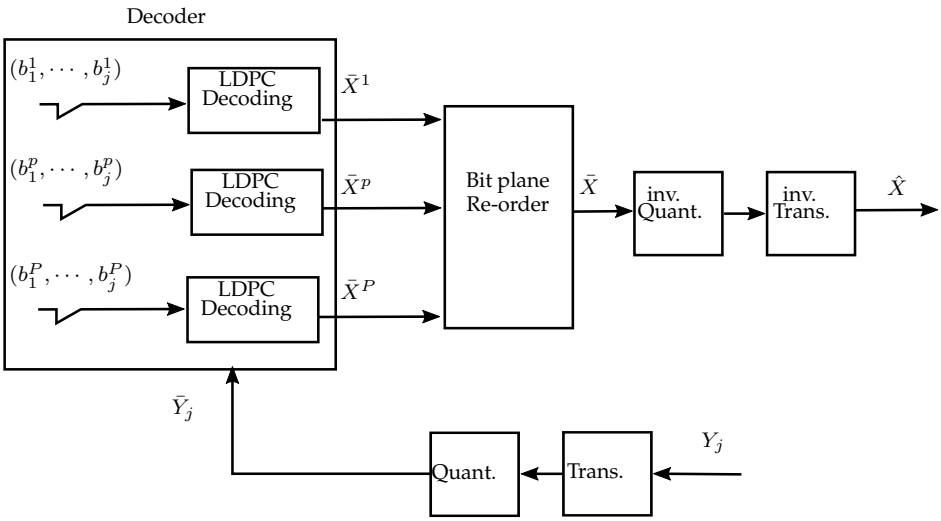
$$\mathbb{H}(\theta) = \{j \mid X_{(j)} \in (m \circ \gamma)(\theta)\} \tag{3.4}$$

where $(m \circ \gamma)(\theta)$ represents the requested region in the 2D mapped domain (see Section 2.2 for the related notations). We need to seek for a scanning order for decoding, i.e., a permutation $\tau$ of this set such that the transmission rate is minimized. $\tau(j)$ stands for the position of $j$ in the new arrangement.

To solve this optimization problem, one needs to know the encoding rates of each block for each possible SI source. This can only be solved at the encoder, and the scanning order must be sent to the decoder. However, transmitting the scanning order may impose overhead on the transmission rate. Instead, to avoid this overhead, one can use heuristics at both encoder and decoder to determine the




(a)




(b)

Figure 3.5 – Proposed lossy coding scheme of an image block $X$ as a function of its predictions $Y_1, \ldots, Y_N$. (a) Offline encoding function. (b) Decoding procedure.

processing order. Heuristics will be proposed in Section 4.1.3 for 360° images and in Section 4.3.3 for texture maps of 3D models.

## 3.5 Summary

In this chapter, we proposed a novel interactive coding scheme for 2D representation of interactive images modalities. This scheme achieves the oracle transmission cost while keeping the storage cost reasonable. More precisely, the storage size is far less than the exhaustive storage solution and close to the solution without random access.

Classical compression schemes work block-wise, and use a fixed block scanning order for compression and decompression which is not suitable for interactive compression. However, it is proven theoretically that with incremental codes (the SMRA approach), we can relax this fix ordering and encode the data as it is defined in the navigation graph. This way a source is encoded once, but it is decoded in several ways depending on which SI is available at the decoder. We proposed solutions to put these theoretical results into practice. More precisely, we defined a navigation graph in the 2D image. Using the graph, blocks are encoded such that any decoding order is possible at the decoder's side. As a consequence, i) only the useful blocks are sent and decoded, ii) the correlation between the blocks is still taken into account. Therefore, i) and ii) enables our coder to reach the same transmission rate as the oracle coder.

In practice, for each block to be encoded, several predictions are generated (corresponding to different block decoding order), and a bitstream is built based on the incremental entropy codes studied in [70]. This bitstream compensates for the worst prediction, and a substream can be extracted from it to compensate for any other prediction. For some blocks, called access blocks, an empty prediction is added to the prediction set that brings the ability to decode the block independently. These blocks are spread over the image so that the system can initiate the decoding operation at any request. Upon request of a set of blocks from the user, the system starts the decoding from an access block and then the remaining blocks are decoded one after the other with a specific scanning order. This way, only the requested blocks are transmitted.

CHAPTER 4

# CODING DESIGN FOR OMNIDIRECTIONAL IMAGES AND TEXTURE MAP OF 3D MODELS

In the previous chapter, we proposed a generic 2D image coder based on incremental codes that provide the ability to access part of the image. With this incremental coder, the system can decode any scanning order that respects a given navigation graph. However, a scanning that provides a better prediction is preferred because it reduces the transmission rate. Therefore, the choice of scanning order is critical and has an impact on the performance of the interactive system. Moreover, the existence of access blocks that can be decoded independently is undeniable, but the larger the number of access blocks, the larger the storage size. Therefore, the position of these blocks should be such that they can respond to every request, but at the same time, their number should be as low as possible.

In this chapter, according to the proposed compression scheme, we consider the interactive compression of two imaging modalities that are interactive by their nature: 360° images and texture map of 3D models. Each of these images has its own characteristics that need to be studied separately. More precisely, we study the access block placement in each of these modalities. We also explain how the predictions of each block are generated depending on different scanning order. Finally, we propose solutions to optimize the scanning order based on the properties of each modality.

## 4.1 Interactive compression of 360° images

As stated in Section 3.3, the proposed coder is compatible with any mapping $m$ of 360° images. Here, without loss of generality, we adopt the equirectangular projection as this projection is the most common projection used in the community. To build an efficient end-to-end coder, a new strategy is developed to spread the access blocks over the image such that the decoding process can be triggered whatever the requested viewport is. We then propose novel prediction modes to adapt not only to interactivity but also to the geometrical properties and discontinuities of the 360° equirectangular mapped data. Finally, a new scanning order is proposed to further lower the transmission rate.

### 4.1.1 The access block placement problem

Recalling the access block placement formulation stated in (3.3), the access blocks must be located such that with any rotation angle $\theta$ of the HMD device, there exists at least one access block in the corresponding requested region of the equirectangular image.

We apply a greedy algorithm to find the location of access blocks. For that, we sweep the sphere from the north pole to the south pole by changing the viewing direction $\theta$ and generating the corresponding viewport. At each step, we check if an access block exists in the requested region of the equirectangular image corresponding the the viewport, i.e., $(m \circ \gamma)(\theta)$. If not, the central block of the corresponding area is added to the $\mathbb{A}$ set. The details of the algorithm can be found in Algorithm 1. The obtained access blocks located in the equirectangular image are shown in Fig. 4.1. The advantage of this method is that it depends only on the size of the blocks, size of the equirectangular image, and the FOV of the HMD device. Therefore, once they are set, this greedy algorithm can be performed once, and we can use the same placement of access blocks for all similar settings.

**Algorithm 1** Algorithm for placing access blocks

**Input:**
$\theta = (\theta_1, \theta_2)$
$\theta_1$: longitude, $\theta_1 \in [-\pi, \pi)$
$\theta_2$: latitude, $\theta_2 \in [-\pi/2, \pi/2]$
$\Delta\theta_1, \Delta\theta_2$: predefined steps
$center(\theta)$: returns the block index in $m(\Gamma)$ which contains the center of $\gamma(\theta)$
**Output:** $\mathbb{A}$
*Initialization*: $\mathbb{A} = \emptyset$
**for** $\theta_2 = -\pi/2$ **to** $\theta_2 \leq \pi/2$ **do**
  **for** $\theta_1 = -\pi$ **to** $\theta_1 < \pi$ **do**
    $\theta = (\theta_1, \theta_2)$
    **if** $\nexists\, k \in \mathbb{A}$ s.t. $X_k \in (m \circ \gamma)(\theta)$ **then**
      $\mathbb{A} \leftarrow \mathbb{A} \cup \{center(\theta)\}$
    **end if**
    $\theta_1 \leftarrow \theta_1 + \Delta\theta_1$
  **end for**
  $\theta_2 \leftarrow \theta_2 + \Delta\theta_2$
**end for**
**return** $\mathbb{A}$

### 4.1.2 Set of possible side information and prediction functions

Recall from Chapter 3 that for each block, several predictions are generated corresponding to different block decoding orders. In 360° images the adjacent blocks can be used to produce the prediction. The prediction of a given block by one or several blocks of its neighbors is performed by adapting the intra-prediction module of conventional video coding standards [104]. More precisely, the boundary row/column pixels from the adjacent blocks are combined to form a directional or planar prediction. By construction, the prediction depends on which adjacent blocks are already decoded and available to serve as a reference. In conventional video coding, the blocks are encoded and decoded with a raster

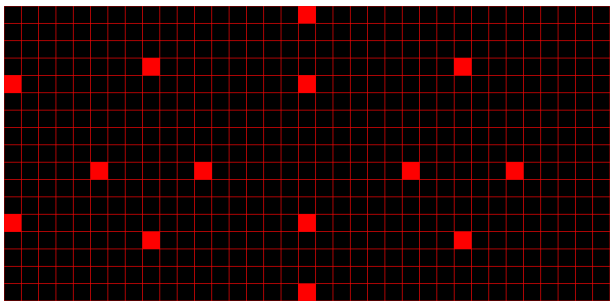


Figure 4.1 – Position of the access blocks (in red) in the equirectangular image using the proposed algorithm. The equirectangular image is of size 8960x4352 with the horizontal FOV of 110° and vertical FOV of 77.5°. Blocks are of size 256x256.

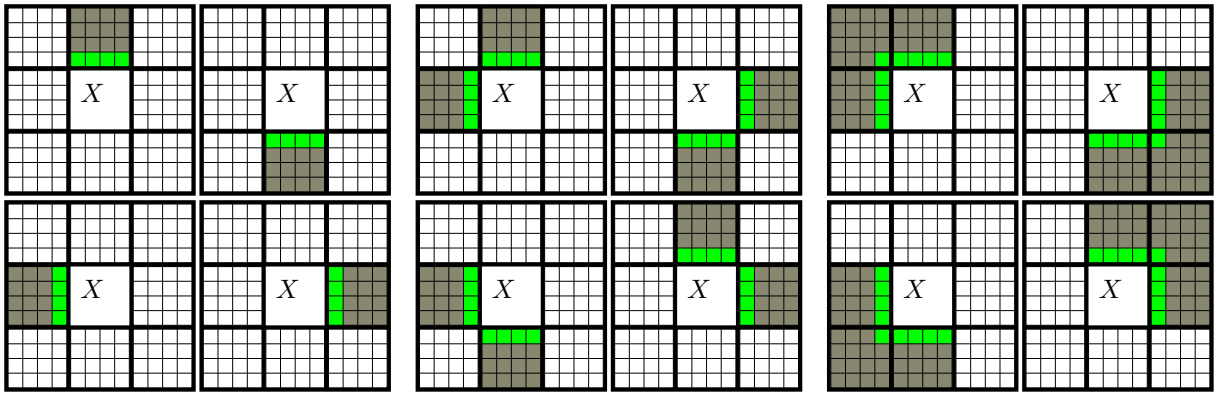
scanning order, which means that the reference blocks are always the left and/or the top blocks.

In our case, the order depends on the requested viewport and therefore, the decoder should be able to address any possible order. This decoding order influences which blocks are used as references (also called context or causal information). Given these references, new prediction functions have to be defined. We categorize the reference blocks used to generate prediction according to the number of neighboring blocks available at the decoder. This leads to three types, as illustrated in Fig. 4.2:

— *Type 1*: Only one of the adjacent blocks of 4-connected neighboring has been decoded.
— *Type 2*: A vertical and a horizontal border at the boundaries are available to perform the prediction. This means that a block at the top or bottom and a block at the right or left side of the current block have been decoded already.
— *Type 3*: This type is similar to *type 2* except that, in addition to the vertical and horizontal border, the corner block between them has also been decoded.

For each type, there exist 4 possibilities, which lead to 12 possible contexts. Each context gives rise to a possible prediction (denoted by $Y_1, Y_2, \ldots, Y_N$). Prediction functions are deduced from the prediction modes of conventional video coding standards by a rotation of multiples of 90°. Each prediction corresponds to one possible SI. In order to be consistent both at encoder and decoder, and avoid error propagation, the predictions are generated from the lossy version of the neighboring blocks.

In general, if there is no limitation for the navigation, all of these 12 predictions are generated and used to encode the block $X$. For the blocks which are on the top (bottom) row of the equirectangular image, since there exists no block on top (bottom), the non-available side information predictions are removed from the possible set. By contrast, for the blocks which are on the left and right boundaries of the equirectangular image, since when they are wrapped to the surface of the sphere they are adjacent to each other, they can be used as a reference to predict the opposite side. For example, for a block which is located at the rightmost part of the equirectangular image, the leftmost block at the same row can be used as a reference for its prediction and vice versa. This corresponds to edges D and E of the navigation graph depicted in Fig. 3.2c.




(a) (b) (c)

Figure 4.2 – Reference pixels used to predict the block $X$ from the neighboring blocks in the proposed 360° interactive compression. The prediction serves as SI for the entropy coder. The available neighboring blocks are depicted in dark gray. (a) Type 1 references (4 predictions). (b) Type 2 references (4 predictions). (c) Type 3 references (4 predictions).

### 4.1.3 Decoding order and navigation

As noted in Section 3.4.4, transmitting the scanning order can waste the network bandwidth. We instead propose a heuristic to estimate the decoding order at the decoder from its already available data. Indeed, the quality of the prediction, which impacts the transmission rate, increases with the number of neighboring blocks that are served as reference (Type 1, 2 or 3 in Fig. 4.2). Therefore, we propose to find a decoding order that maximizes the overall number of blocks used for predictions i.e.

$$\max_{\tau} \sum_{k \in \mathbb{H}(\theta)} |\{j' \mid \tau(j') < \tau(k), X_k \sim X_{j'}\}|, \tag{4.1}$$

where as explained in (3.4), $\mathbb{H}(\theta)$ represents the set of requested blocks, function $\tau$ sorts the blocks to determine the ordering, and $\sim$ means that two blocks are neighbors ($j' \in \mathbb{J}_{(k)}$).

Moreover, according to the experiments done, we found that the correlation between horizontal blocks are higher than vertical blocks, i.e. the prediction that is generated by the left or right block generates less distortion than the prediction generated from the top or bottom block. Therefore, we augmented the criterion to favor horizontal predictions

$$\begin{aligned} &\max_{\tau} \sum_{k \in \mathbb{H}(\theta)} |\{j' \mid \tau(j') < \tau(k), X_k \sim X_{j'}\}| + \mathbb{1}_{\mathcal{E}}(k) \\ &\mathcal{E} = \{k | \exists j', X_{j'} \sim X_k, X_{j'} \in \text{ same line as } X_k\} \end{aligned} \tag{4.2}$$

where $\mathbb{1}_{\mathcal{E}}(k)$ is the characteristic function of the set $\mathcal{E}$.

Solving the problem (4.2) requires an exhaustive search among all admissible decoding orders. We propose instead to directly implement a solution inspired by the one-scan connected component labeling technique [2]. The solution is a snake-like scan that is adapted to the characteristics of a requested area. The algorithm is detailed in Algorithm 2. Indeed, the proposed solution can start at any position of

a block inside the requested region (not necessarily at a corner) and is adapted to any shape (not necessarily blocks arranged in the shape of a rectangle). These latter adaptations are made possible through a recursive implementation of the scanning order. The scanning is well adapted to our problem because, first, almost all blocks have at least two causal neighbors, except for the first scanned line, and second, the number of horizontal transitions is significantly larger than the number of vertical transitions. Moreover, the algorithm enables to traverse horizontally from a boundary block on one side to the boundary block on the opposite side of the image (at the same row) because these two regions are connected on the sphere surface (see line 10 of Algorithm 2). An example of such a scanning is shown in Fig. 4.3.

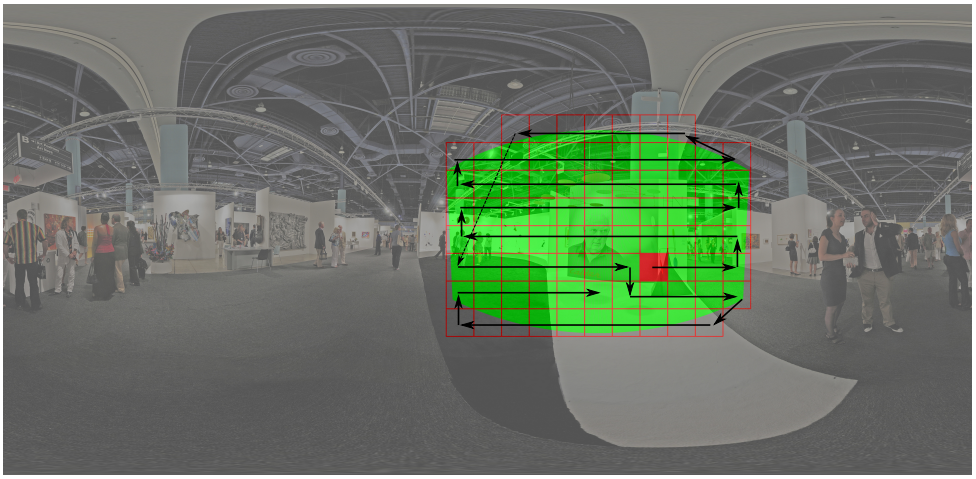


Figure 4.3 – An example of the proposed scanning of the blocks for interactive compression of 360° images based on Algorithm 2.

Navigation within a 360° image is also handled by our interactive coder. Indeed, navigation consists of successive requests of viewports caused by the continuous change of the user's viewing direction. Therefore, for any newly requested viewport, the already decoded blocks of this viewport and its boundary are determined. Then, one of these blocks is chosen as the starting point. In this case, the proposed method performs even better since it is no longer necessary to send a block as an access block independently, and one of the already decoded blocks in the previous viewport request is chosen as the starting point. A decoding order is computed for the remaining non-decoded blocks. In conclusion, our scheme efficiently handles navigation by sending only the unknown blocks.

### 4.1.4 Experimental results

For the experiment, 8 large panoramic images of size 9104x4552 are chosen from SUN360 database [106]. Four of them have already been shown in Fig. 2.6, and the other four are shown in Fig. 4.4. Blocks have size of $32 \times 32$. The images are resized to 8960x4480 to be divisible by the size of blocks and also have an aspect ratio of 2:1 as commonly adopted in the community [63].

Two rate-adaptive (RA) codes have been considered in our approach: LDPCA [97] and Protograph-Based LDPC codes [109]. The results presented here are based on LDPCA and Protograph-based LDPC performance evaluated by our collaborators [109]. For the sake of fairness, we adopted pessimistic

**Algorithm 2** Algorithm for decoding order when $m(\Gamma)$ is equirectangular projection. Comments are shown in *italic* between /* and */.

**Input:**
$\mathbb{H}(\theta)$: set of requested blocks
$\mathbb{A}$: set of access blocks
**Output:** permutation $\tau\big(\mathbb{H}(\theta)\big)$
*Initialization*:
$\mathbb{B} \leftarrow \mathbb{H}(\theta)$
$i \leftarrow 0$
**Find** $k \in \mathbb{A}$ s.t. $k \in \mathbb{B}$
recursive_processing_neighbor($k$)
**return** $\tau\big(\mathbb{H}(\theta)\big)$

**Function** recursive_processing_neighbor($k$):
$pos : k \mapsto (x, y)$, converts block $k$ index to position in $m(\Gamma)$
$n_{row}$: number of blocks per row in $m(\Gamma)$
$n_{col}$ : number of blocks per col in $m(\Gamma)$
*/* $dx, dy$ *give the priority to process the horizontal neighbors first* */*
$dx = [1, -1, 0, 0]$
$dy = [0, 0, -1, 1]$

$\tau(k) \leftarrow i$
$\mathbb{B} \leftarrow \mathbb{B} \setminus \{k\}$ */* *remove* $k$ *from set* $\mathbb{B}$ */*
**if** $\mathbb{B} = \emptyset$ **then**
  **return**
**end if**
$i \leftarrow i + 1$
$(x, y) \leftarrow pos(k)$
**for** $j = 0$ **to** $4$ **do**
  $x_{neighbor} \leftarrow x + dx[j]$
  */* *to enable circular traversing for 360° video in x direction:* */*
  $x_{neighbor} \leftarrow x_{neighbor} \mod n_{row}$
  $y_{neighbor} \leftarrow y + dy[j]$
  **if** $y_{neighbor} \geq n_{col}$ or $y < 0$ **then**
    continue
  **end if**
  $k \leftarrow pos^{-1}(x_{neighbor}, y_{neighbor})$
  **if** $k \in \mathbb{B}$ **then**
    recursive_processing_neighbor($k$) */* *recursive call* */*
  **end if**
**end for**
**return**

(a) Image Exhibition

(b) Image Museum

(c) Image Room

(d) Image Ruins

Figure 4.4 – The other four images used in our experiments to evaluate interactive coder for 360° images.

performance evaluation. The full coder integrating the channel code is being currently implemented.

We consider two coding schemes as baselines to evaluate the proposed coder, namely tile-based scheme and exhaustive storage. For the tile-based approach, we consider five different tile sizes which partition the equirectangular image into $7 \times 7$, $7 \times 5$, $5 \times 5$, $5 \times 4$, $4 \times 4$ and $2 \times 2$ non-overlapping tiles. We denote them by "T. $m \times n$" where $m$ and $n$ are the numbers of, respectively, vertical and horizontal tiles. We also consider the case when the whole equirectangular image is sent in the first request (T. $1 \times 1$). The Exhaustive Storage (ES) approach does not really exist in practice, but it illustrates the intuitions of some baselines adopted in other contexts [44, 57, 24, 14, 42]. As in our coding scheme, the ES scheme considers all the predictions for each block, but stores a residue for each of them. It is thus well adapted for transmitting at the oracle transmission rate $R^{oracle}$, but has a huge storage cost.

We simulate user navigation as explained in Section 2.4.1. We generate navigation paths of $U = 20$ users where each has $T = 600$ requests. For each image, the transmission rate per user and per request $r(\theta_t, u), u = 1, ..., U, t = 1, ..., T$ is computed together with distortion in the viewport according to (2.6).

#### 4.1.4.1 Achievability of the oracle transmission rate

We recall that the aim is to propose a coder that is able to achieve the *oracle* transmission rate $R^{oracle}$, obtained if the user head movement was known at the encoder. For that purpose, we compute the accumulated transmission rate for one user during successive requests, i.e.,

$$R(\theta_{T'}, u) = \sum_{t=1}^{T'} r(\theta_t, u).$$

We show in Fig. 4.5 an illustration of what we obtain at iso-distortion of PSNR 37 dB for two of the images. Similar behaviors were observed for other images and other PSNR values. We can see that, in theory, our proposed coder achieves the oracle transmission rate. In practice, the suboptimality of the adopted coders leaves a gap between the practical performance and the theoretical one. However, we can see that our implementation obtains much better performance than the tile-based ones, especially at the beginning of the request. Moreover, we can observe that our method has a behavior that is better suited to interactivity than the tile-based approach. Indeed, it has a smooth $R(\theta_{T'}, u)$ evolution during the user's head navigation, meaning that our scheme transmits only what is needed upon request. By contrast, the tile-based approaches have a staircase behavior, meaning that there is a burst of rate at the moment when a new tile is needed (in which some blocks are useless).

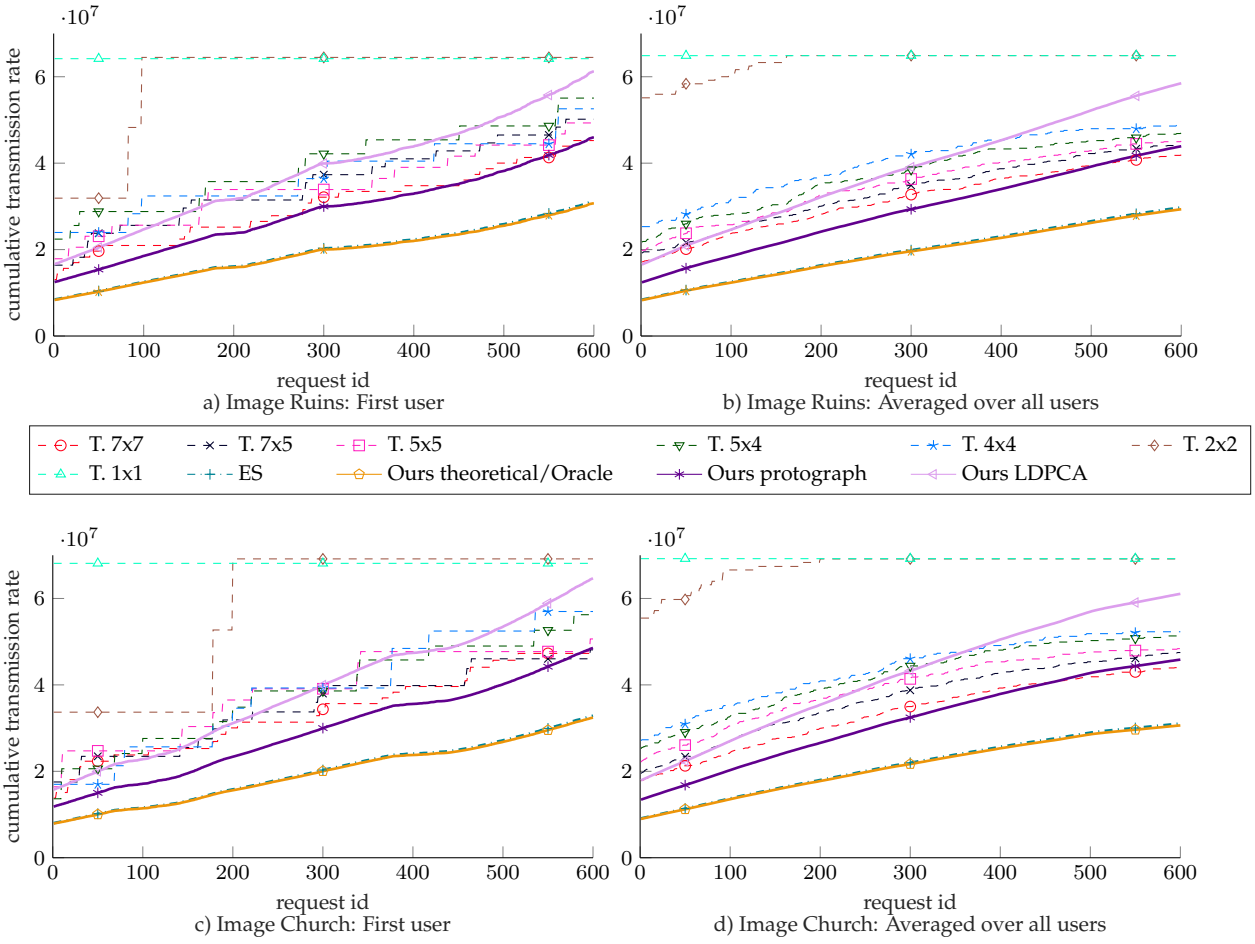



Figure 4.5 – Accumulated transmission per request at iso-distortion PSNR 37dB for image Ruins (top row) and image Church (bottom row). Left column shows the accumulated transmission for the first user. Right column shows the accumulated transmission averaged over all users' requests.

Another way to analyze the performance of the interactive coder is by means of the *usefulness* of the transmitted data. In the following, the usefulness is defined as the proportion of displayed pixels among

all the decoded ones for a request $\theta$:

$$\mathcal{U} = \mathbb{E}_\theta(\mathcal{U}_\theta)$$

where

$$\mathcal{U}_\theta = \frac{\sharp \text{ displayed pixels for a request } \theta}{\sharp \text{ decoded pixels for a request } \theta}. \tag{4.3}$$

The usefulness averaged over all users is shown in Fig. 4.6 and confirms that our scheme sends blocks that are all observed by the user, contrary to the tile-based approach.

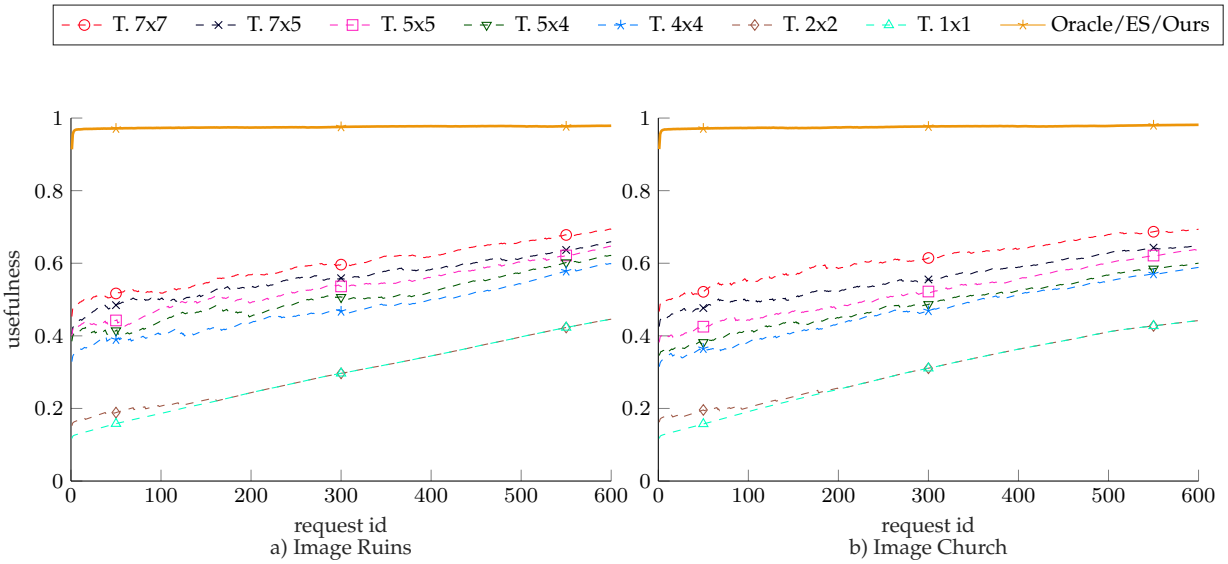



Figure 4.6 – Usefulness of the observed pixels per request average of all users' requests for 360° images Ruins (left) and Church (right).

In all these experiments, the ES scheme performs as good as our solution in terms of the transmission rate. More exactly, it performs as good as our theoretical performance when using an optimal rate-adaptive code. In practice, they use an arithmetic coder that does not suffer from some sub-optimality of rate-adaptive-based channel coder implementation. However, in the ES approach, the focus is on the transmission rate only, and this is why in the following we present an evaluation that includes the storage cost.

#### 4.1.4.2 Analysis of the Rate-storage trade-off

We now discuss the performance of different coders in terms of rate and storage performance. In Section 2.3.3, a weighted Bjontegaard metric is introduced, in which the weight $\lambda$ balances the relative importance between the rate and the storage, as a function of (2.7). In the following, we first study the two extreme scenarios where either the rate or the storage is neglected which result in storage-distortion (SD) and rate-distortion (RD) performances respectively. Then, we show results for more realistic scenarios where both matters.

**RD and SD performances**

In Fig. 4.7 we compare the performance when either the storage ($\lambda = 0$) or the transmission rate ($\lambda = \infty$) is neglected, for queries of length 10 averaged over all users. We see that the proposed method with protograph-based rate-adaptive codes performs better than tiling approach with only a small extra cost in storage. Compared to the ES approach, the ES performs better on the transmission rate, which is basically because rate-adaptive channel codes are not optimal for short-length sequences but, as expected, the ES approach performs poorly in terms of storage.

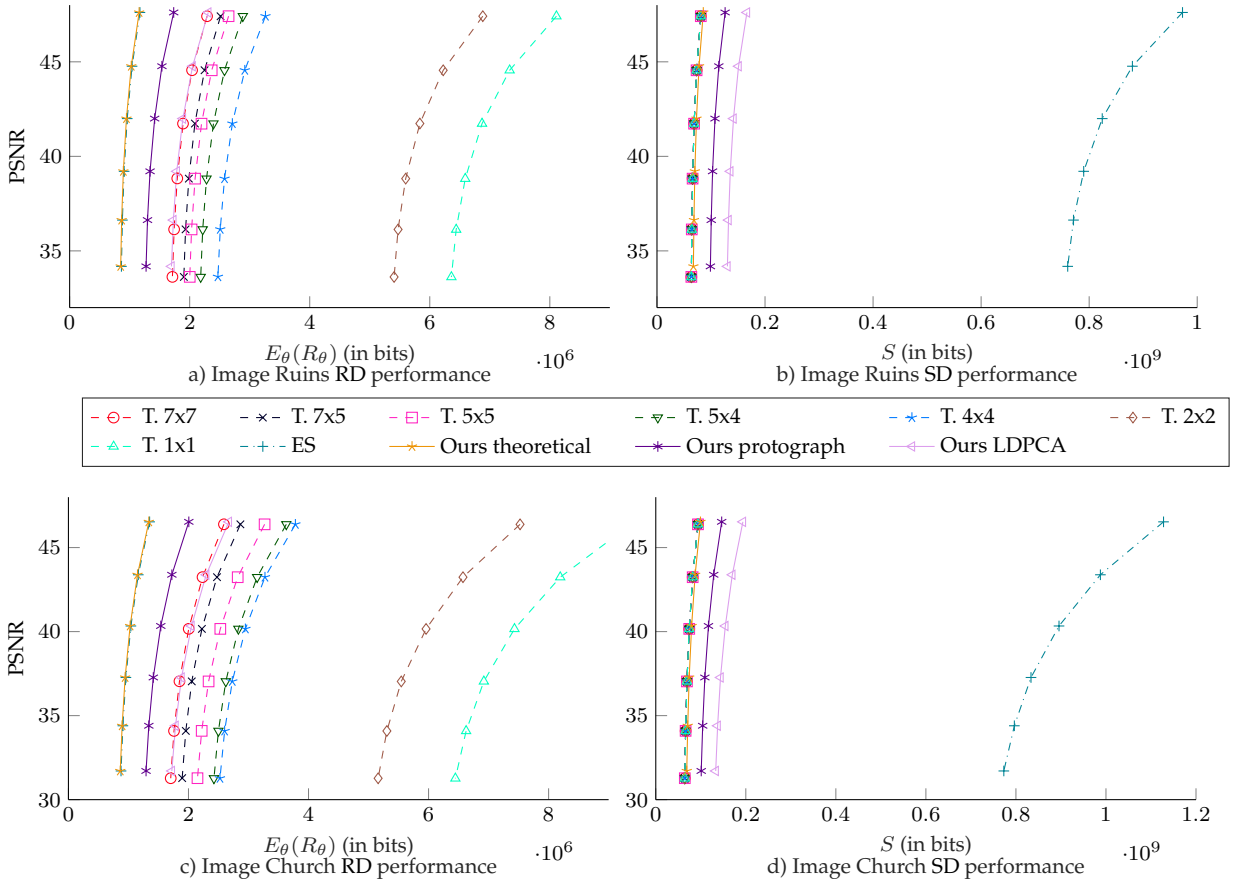



Figure 4.7 – Comparison of storage and transmission performance for requests of length 10 using the proposed interactive coder. (top row) Image Ruins. (bottom row) Image Church.

The averaged bitrate difference in transmission rate and storage is summarized in Table 4.1 showing the performance of each method relative to the no tiling (T. $1 \times 1$) approach. Negative values indicate bitrate saving with respect to no tiling approach. Due to space preservation, intermediary tile sizes are removed from the table and only the important baselines are shown. It is observed that while the performance of protograph-based rate-adaptive codes does not exactly reach its theoretical rates, they still perform better than the tile-based approach in terms of transmission rate even at a long duration of 400 requests per user: -44.80% for protograph-based rate-adaptive coder against -42.92% for T. $7 \times 7$ (the lower the negative value, the better the performance). In terms of storage performance, we observe

that the proposed method performs significantly better than exhaustive storage (the greater the positive value of BD, the worse the performance). In other words, the proposed method loses $56.26\%$ of storage with respect to no tiling approach while exhaustive storage loses $1097.56\%$. If we look at the theoretical results of the proposed coder, we see that in theory this coder should only lose $6.34\%$ in terms of storage and gain $86.22\%$ in transmission rate, which clearly shows there is a lot of room for improvement in designing rate-adaptive codes.

Table 4.1 – BD measures (in percent) averaged over all users relative to the no tiling approach (T. 1x1). BD-R $n$ is the BD metric for transmission rate for requests of length $n$. BD-S is the BD-storage measure.

| | | Plaza | Church | Exhibition | Museum | Room | Ruins | Workshop | Pool | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| BD-R 10 | T. 2x2 | -23.61 | -19.85 | -13.85 | -21.27 | -12.47 | -15.07 | -12.46 | -15.00 | -16.70 |
| | T. 7x7 | -73.08 | -73.09 | -72.17 | -72.92 | -72.31 | -72.57 | -73.43 | -73.22 | -72.85 |
| | ES | -85.64 | -86.11 | -85.81 | -86.24 | -85.81 | -86.15 | -86.27 | -86.32 | -86.04 |
| | Ours theoretical | -85.82 | -86.29 | -85.97 | -86.43 | -85.98 | -86.34 | -86.46 | -86.50 | -86.22 |
| | Ours protograph | -78.76 | -79.47 | -78.99 | -79.68 | -79.00 | -79.55 | -79.73 | -79.79 | -79.37 |
| BD-R 400 | T. 2x2 | 0.01 | 0.01 | -1.29 | 0.01 | 0.02 | 0.00 | 0.01 | 0.01 | -0.15 |
| | T. 7x7 | -42.22 | -42.84 | -42.41 | -44.28 | -42.61 | -42.97 | -41.77 | -44.26 | -42.92 |
| | ES | -62.88 | -62.64 | -60.75 | -63.50 | -62.17 | -63.90 | -62.17 | -63.15 | -62.65 |
| | Ours theoretical | -63.35 | -63.13 | -61.21 | -64.01 | -62.62 | -64.38 | -62.70 | -63.64 | -63.13 |
| | Ours protograph | -45.12 | -44.79 | -41.91 | -46.12 | -44.03 | -46.68 | -44.17 | -45.57 | -44.80 |
| BD-R 600 | T. 2x2 | 0.01 | 0.01 | -1.29 | 0.01 | 0.02 | 0.00 | 0.01 | 0.01 | -0.15 |
| | T. 7x7 | -34.18 | -35.84 | -36.39 | -38.32 | -35.61 | -34.69 | -35.72 | -36.73 | -35.93 |
| | ES | -54.32 | -54.82 | -51.58 | -55.83 | -54.60 | -53.49 | -55.02 | -55.75 | -54.43 |
| | Ours theoretical | -54.90 | -55.41 | -52.15 | -56.44 | -55.14 | -54.12 | -55.65 | -56.34 | -55.02 |
| | Ours protograph | -32.48 | -33.24 | -28.34 | -34.79 | -32.83 | -31.31 | -33.61 | -34.64 | -32.66 |
| BD-S | T. 2x2 | 0.01 | 0.01 | 0.01 | 0.01 | 0.02 | 0.00 | 0.01 | 0.01 | 0.01 |
| | T. 7x7 | 0.03 | 0.04 | 0.06 | 0.06 | 0.07 | 0.02 | 0.04 | 0.03 | 0.04 |
| | ES | 1097.19 | 1097.43 | 1099.70 | 1100.38 | 1102.33 | 1092.96 | 1095.41 | 1094.95 | 1097.54 |
| | Ours theoretical | 5.94 | 6.38 | 5.75 | 7.32 | 6.75 | 5.75 | 6.47 | 6.36 | 6.34 |
| | Ours protograph | 55.68 | 56.36 | 55.77 | 57.56 | 57.05 | 55.28 | 56.25 | 56.17 | 56.26 |

**Weighted BD**

The weighted BD is presented for more realistic scenarios, where both storage and transmission rates matter, with relative importance. Lower values of $\lambda$ given in (2.7) mean that the importance is given more to transmission rate and, on the contrary, higher $\lambda$ values give the importance to storage. Weighted BD of different methods is summarized in Table 4.2. In our experiments, for tile-based and our approach, the storage is $10 - 100$ times larger than the transmission rate and for the exhaustive storage, the storage is usually $800 - 1000$ times the transmission rate. Therefore, imposing $1e^{-3} \leq \lambda \leq 0.01$ gives the same importance between transmission rate and storage. It can be seen from Table 4.2 that for $\lambda = 0.01, 1e^{-3}$ the protograph-based RA code performs better than others which means that if storage and transmission rates are of the same importance, the proposed method performs better. Decreasing $\lambda$ to $1e^{-4}$ emphasizes the transmission rate where exhaustive storage is more suitable. Finally, if the storage is more important than the transmission rate ($\lambda \geq 0.1$) tiling approaches perform better.

### 4.1.5 Conclusion of the results for the proposed interactive coder

Experimental results show that our scheme balances the trade-off between transmission rate and storage. In addition, the usefulness of the transmitted blocks shows that the proposed scheme better

Table 4.2 – Weighted BD for requests of length 10 averaged over all users relative to the no tiling approach (T. 1x1).

| | | Plaza | Church | Exhibition | Museum | Room | Ruins | Workshop | Pool | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| $\lambda = 0.1$ | T. 2x2 | -11.80 | -9.92 | -6.92 | -10.63 | -6.22 | -7.53 | -6.23 | -7.50 | -8.34 |
| | T. 7x7 | **-36.52** | **-36.52** | **-36.05** | **-36.43** | **-36.12** | **-36.27** | **-36.70** | **-36.59** | **-36.40** |
| | ES | 505.90 | 506.04 | 506.89 | 507.21 | 508.13 | 503.39 | 504.69 | 504.49 | 505.84 |
| | Ours protograph | -11.51 | -11.50 | -11.61 | -11.04 | -10.99 | -12.13 | -11.72 | -11.79 | -11.54 |
| $\lambda = 0.01$ | T. 2x2 | -21.47 | -18.04 | -12.59 | -19.33 | -11.33 | -13.70 | -11.33 | -13.64 | -15.18 |
| | T. 7x7 | -66.43 | -66.44 | -65.60 | -66.29 | -65.73 | -65.97 | -66.75 | -66.56 | -66.22 |
| | ES | 21.92 | 21.55 | 21.96 | 21.66 | 22.18 | 21.04 | 21.18 | 21.10 | 21.58 |
| | Ours protograph | **-66.53** | **-67.11** | **-66.74** | **-67.20** | **-66.64** | **-67.29** | **-67.37** | **-67.42** | **-67.04** |
| $\lambda = 1e^{-3}$ | T. 2x2 | -23.38 | -19.65 | -13.72 | -21.06 | -12.34 | -14.92 | -12.34 | -14.85 | -16.53 |
| | T. 7x7 | -72.35 | -72.36 | -71.45 | -72.20 | -71.60 | -71.85 | -72.71 | -72.49 | -72.13 |
| | ES | -73.92 | -74.38 | -74.07 | -74.49 | -74.05 | -74.48 | -74.57 | -74.62 | -74.32 |
| | Ours protograph | **-77.43** | **-78.12** | **-77.65** | **-78.32** | **-77.66** | **-78.21** | **-78.39** | **-78.44** | **-78.03** |
| $\lambda = 1e^{-4}$ | T. 2x2 | -23.59 | -19.83 | -13.84 | -21.24 | -12.45 | -15.05 | -12.45 | -14.99 | -16.68 |
| | T. 7x7 | -73.00 | -73.01 | -72.09 | -72.85 | -72.24 | -72.50 | -73.36 | -73.15 | -72.78 |
| | ES | **-84.46** | **-84.93** | **-84.62** | **-85.05** | **-84.62** | **-84.97** | **-85.09** | **-85.14** | **-84.86** |
| | Ours protograph | -78.63 | -79.33 | -78.85 | -79.55 | -78.87 | -79.41 | -79.60 | -79.65 | -79.24 |

adapts to the queries of the users and the transmission rate increases gradually with the duration of the request. This avoids the staircase effect in the transmission rate that occurs in tile-based approaches and makes the encoder perfectly suited for interactive transmission.

## 4.2 Beyond the 2D representation: Towards a better representation for 360° contents

As mentioned, the methods available for compressing spherical images all rely on first presenting them with 2D rectangular images. However, as explained in Section 1.3.2.2, these 2D representations have major drawbacks. To solve these issues, in this section, we propose to compress and process the spherical content directly on the sphere. The goal is to avoid projecting the data onto 2D planes. Therefore, requirements (ii) and (iii) of Section 1.3.2.2 needed to represent the 360° contents are fulfilled, and the property (i) is no more required. To show the efficiency of the representation, we perform compression on the whole 360° image, but the proposed representation can be used in the interactive coder proposed in the previous section.

Moreover, we focus again on intra coding, where images are coded independently of the others to allow resynchronization. Intra coding is a key element in video compression, as the bitrates devoted to intra-coded frames occupy a significant portion of the coded video bitrate. Therefore, reducing the compression rate of intra coded frames will have a significant impact on the overall video compression performance. Nevertheless, we believe that this approach can be easily extended to inter-frame coding as already some efforts have been made to perform motion compensation directly on the sphere [25, 100, 99].

The proposed approach relies on a quasi-uniform sampling of the sphere (as illustrated in Fig. 4.8) to get a pixelization of the spherical content, called HEALPix (Hierarchical Equal Area isoLatitude Pixelization) [38]. Then, a complete image coder is built. This coder follows the key steps of 2D plane

image compression but is defined on the sphere. More precisely, from the obtained discrete representation of the spherical content, a partitioning of the content into Spherical Blocks (S-blocks) is performed. Both pixelization and S-block partitioning are presented in Section 4.2.1. Then the proposed coder is explained in Section 4.2.2. For that, a scanning order is proposed on the sphere. This order induces the causal information that can be used to process the current S-block. Then, prediction on a S-block level is proposed to exploit redundancies between S-block. Scanning order and prediction are presented in Section 4.2.2.1. Finally, the output of the prediction is further processed to remove redundancies within the S-block. To do so, a graph transform is used (see Section 4.2.2.2), similar to the transform used in [60].

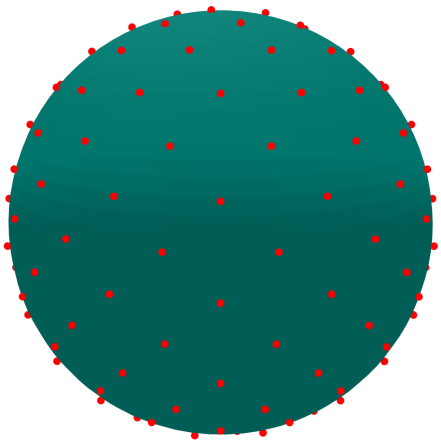
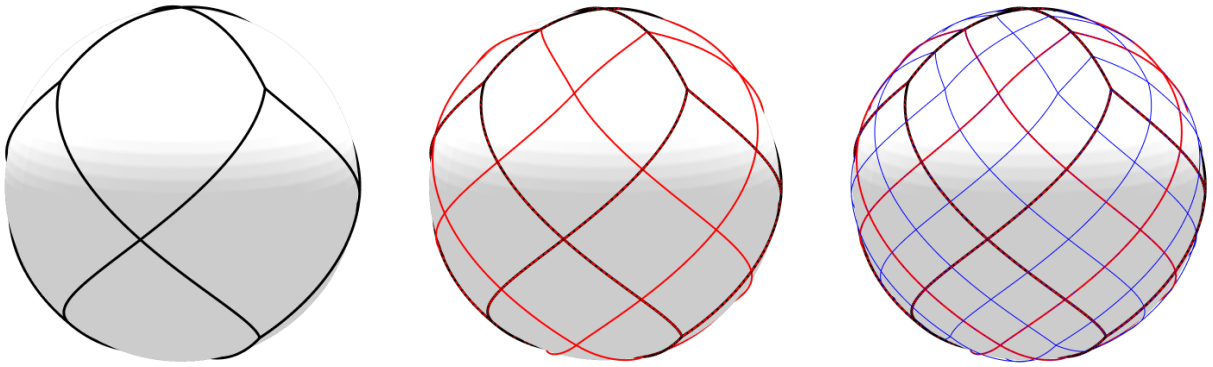


Figure 4.8 – Uniform sampling of the sphere.

### 4.2.1 Sphere pixelization and S-block partitioning

Uniform sampling of the sphere is obtained with the HEALPix scheme [38], a popular sampling scheme used in cosmology and astrophysics. The process starts with a tessellation of the sphere into 12 equal-area regions (base resolution). This tesselation is shown in Fig. 4.9a. Each region is a quadrilateral with curvilinear non-geodesic boundaries. The centers of the 12 regions are located on only three iso-latitude circles. Then, to increase the resolution, each region is further divided into 4 equal-area regions, see Fig. 4.9b. Finally, the partitioning is repeated to reach the desired resolution, see Fig. 4.9c. Once the desired resolution has been achieved, the pixelization is obtained by assigning one pixel to the center of each region.

The first and principal property of the HEALPix pixelization is to yield uniform sampling of the sphere. Second, the iso-latitude property allows fast and efficient processing of huge size data. These properties were the original requirements that motivated the construction of HEALPix and are also important in the context of image compression. Moreover, HEALPix offers other properties, which are of great interest in our context. First, the hierarchical division of pixels into 4 equal-area pixels to obtain higher resolution can be used to define blocks. More specifically, as conventional 2D coders, pixels can



(a) (b) (c)

Figure 4.9 – HEALPix hierarchical pixelization of the sphere. (a) Base-resolution decomposition into 12 pixels. (b) Second decomposition into 48 pixels. (c) Third decomposition into 192 pixels.

be gathered into blocks of pixels, where the number of pixels in each side is a power of two. The blocks are called Spherical Blocks (S-blocks) as they differ by construction from planar 2D blocks (see Fig. 4.10). Second, the iso-latitude property allows to define S-blocks which are spread on iso-latitude circles i.e. with horizontal neighbors. This is a figure of merit, as the correlation between neighboring horizontal S-blocks is greater than between vertical ones.

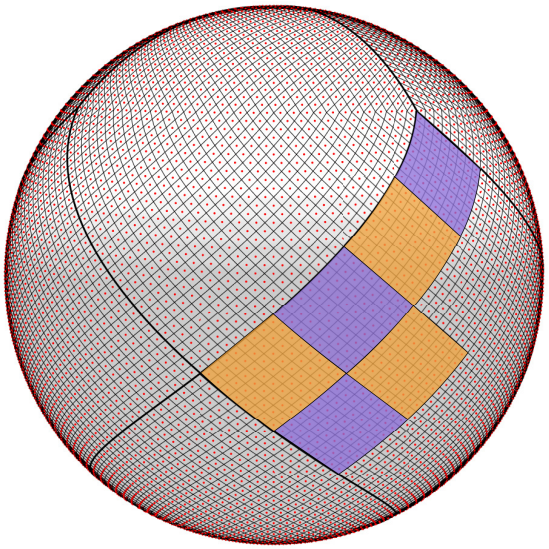


Figure 4.10 – Partitioning of the sphere into S-blocks of 8x8 pixels. An example is shown with a resolution of 12288 pixels. Pixel centers are depicted by red dots, and base-resolution tessellation is shown with black boundaries. S-blocks of 8x8 pixels which correspond to the third decomposition of the base-resolution tessellation are depicted in yellow or violet.

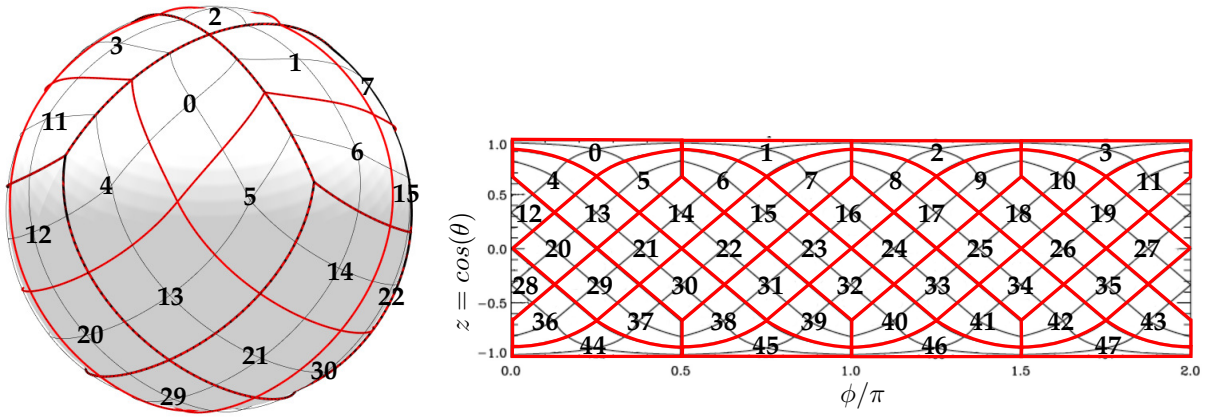
### 4.2.2 Proposed coder

Prediction and transform are two key steps in image compression. Their goals are to exploit the short and long range spatial redundancies of an image. More precisely, the prediction exploits the redundancies between S-blocks whereas the transform rather exploits local redundancies i.e., within a S-block. In this section, we present a novel image coder, where these two types of redundancy are exploited directly on the sphere, and not in a projected domain as classically done.

#### 4.2.2.1 Intra-prediction and scanning order

From the partitioning of the sphere into S-blocks, one can define a block-based prediction algorithm to remove redundancies between adjacent S-blocks. The principle is to predict the current S-block from neighboring S-blocks that have already been encoded/decoded and are therefore available at the decoder. This requires to define a scanning order of the S-blocks.

One property of omnidirectional images is that the correlation between S-blocks with the same latitude is higher than between S-blocks with the same longitude. Therefore, we propose a horizontal scanning order, where S-blocks are processed ring by ring. Moreover, at the beginning of the compression, less neighboring S-blocks are available. So it is important to start the compression with smooth S-blocks for which directional prediction is not needed. This motivates to start the scanning at the north pole since, in 360° content, smooth scenes such as sky or ceiling are usually depicted at the north pole. In a nutshell, the coder starts scanning at the north pole, then it moves down from the north pole to the south pole along each iso-latitude ring (Fig. 4.11). It is worth noting that the proposed scanning order is fixed and therefore does not require any signalization.




(a) (b)

Figure 4.11 – Proposed scanning order of the spherical order for the case when S-blocks are 2x2 and sphere is pixelized into 192 pixels. S-blocks are processed ring by ring starting from the north pole. Red lines are the border of the S-blocks and black thin lines are the border of the pixels. (a) S-blocks on the sphere. (b) Cylindrical projection of S-blocks and pixels.

The prediction consists in predicting the current S-block from neighboring S-blocks. The neighboring S-blocks are the closest (to achieve reliable prediction) and already decoded (to avoid error propagation)

S-blocks. We adapt the intra prediction method that exists in conventional 2D coders [45] with the difference that the neighboring S-blocks participate in producing the prediction are the ones placed north-east, north, and north-west of the current S-block (these blocks are already decoded based on our proposed scanning order). For instance, the S-block with index 13 in Fig. 4.11 is predicted from the S-blocks with index 4, 0 and 5, whereas the S-block with index 5 is predicted only from the S-block with index 0 because this is the only neighboring block which has been decoded so far. Note that since the neighboring S-blocks are defined directly on the sphere, the connectivity between S-blocks is preserved. This is different from the compression methods based on a 2D projection followed by the classical 2D plane intra prediction algorithm [86].

#### 4.2.2.2 Residual coding with graph transform

The residual signal is generated by subtraction of the prediction signal from the current S-block. To further decorrelate the residual signal, a transform is applied to the residual signal. When the signal is defined on a 2D-regular grid, the 2D DCT [86] is very efficient to exploit local dependencies. In our context, the signal is defined on the sphere. Therefore, we propose to apply a transform that takes the sphere geometry into account. Indeed, a Graph Fourier Transform (GFT) [80] is defined based on the geodesic distance between pixel samples.

More precisely, for each S-block we construct a weighted undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \mathbf{W})$, where the set of nodes $\mathbb{V}$ in the graph represents the set of pixels, $\mathbb{E}$ is the set of edges which represents the connectivity between pixels, and $\mathbf{W}$ is the weighted adjacency matrix. We use 8-connected neighbors (except for the boundary pixels of the S-block which have fewer neighbors) to define $\mathbb{E}$. We use the weighted adjacency matrix $\mathbf{W}$ suggested in [60]

$$W_{ij} = \begin{cases} exp\Big(-\frac{d(x_i,x_j)^2}{\rho^2}\Big) & \text{if pixels i and j are neighbors} \\ 0 & \text{otherwise,} \end{cases}$$

where $d(x_i, x_j)$ represents the geodesic distance between pixels $i$ and $j$, and

$$\rho = \frac{1}{|\mathbb{E}|} \sum_{(v_i,v_j)\in\mathbb{E}} d(x_i, x_j)$$

is the average geodesic distance over all connected pixels in the S-block.

Following [80], the combinatorial graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$ can be used to define a Fourier basis on the graph. The degree matrix $\mathbf{D}$ is a diagonal matrix where $D_{ii} = \sum_j W_{ij}$. By construction, $\mathbf{L}$ is symmetric positive semi-definite and has a complete set of orthonormal eigenvectors $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_N]$ with non-negative eigenvalues $\lambda_1, ..., \lambda_N$ where $N$ is the number of pixels in the S-block. The graph Fourier basis is defined as the Laplacian eigenvectors. The GFT of a signal $\mathbf{s} \in \mathbb{R}^N$ is its projection on the eigenvectors given by $\mathbf{U}$

$$\hat{\mathbf{s}} = \mathbf{U}^T\mathbf{s}.$$

The coefficients are then quantized with a uniform quantizer and compressed using an arithmetic coder.

Finally, to avoid error propagation, a Differential Pulse-Code Modulation (DPCM) encoder scheme

[75] is performed as in all classical compression schemes [86]. More precisely, encoded S-blocks are decoded and used as a reference to predict next S-blocks.

### 4.2.3 Experimental results

In this section, we compare the proposed spherical coder with several baselines. In the following, the rate of each coding scheme corresponds to the amount of bits that are necessary to describe the compressed image. Computing the quality loss due to 360° image compression is less straightforward since the input images differ between different coding schemes. We adopt the strategy proposed in [110]. Since 360° images are aimed to be partly visualized by users, the compression loss is estimated on the viewport images.

For that purpose, we take four high-resolution 360° images from [106]: Exhibition, Pool, Workshop, Plaza. These images are in equirectangular format. We generate several ground-truth viewports. Then, to perform a fair comparison of the compression performances of the schemes, we follow the recommendations of [110]. In particular, we down-sample the input equirectangular to a lower resolution such that each method has the same number of input pixels to process. Hence no method is favored over another. Then the same viewports are generated from the compressed images and the PSNR is calculated on each of them. The final given PSNR value corresponds to the PSNR averaged over all viewports.

We compare our proposed spherical coder with two mapping-based baselines: the equirectangular and the rhombic dodecahedron [35]. The number of pixels for the three methods have been set to be almost identical (equal to 3145728 pixels). For the equirectangular-based coder we keep the aspect ratio of the down-sampled equirectangular image to be equal to the ground truth. The rate-distortion plots are displayed in Fig. 4.12. We can see that our spherical coder clearly outperforms the equirectangular-mapping based coder.

More interestingly, the proposed spherical coder also outperforms rhombic-dodecahedron mapping-based coder which uses similar HEALPix pixelization technique. The main difference between our proposed coder and the rhombic-dodecahedron is that in rhombic-dodecahedron the sphere is mapped to four planar rectangular images which introduces distortion and discontinuities, but our coder performs everything directly on the sphere. In particular, Table 4.3 shows that the proposed method allows to reduce the rate by $2\%$ on average, and up to $5.6\%$, with respect to the method based on a 2D mapping of the spherical content onto the faces of the rhombic dodecahedron [35]. Whereas both approaches rely on the same HEALPix pixelization technique, we can conclude that processing the data directly on the sphere is more efficient than mapping and performing 2D processing. This validates the intuitions brought by our spherical coder.

Table 4.3 – BD-rate gain of the proposed method with respect to the the rhombic dodecahedron approach proposed in [35].

| | Exhibition | Pool | Workshop | Plaza | **Average** |
|---|---|---|---|---|---|
| BD-rate gain | **-5.63** % | **-2.89** % | 0.20 % | **-0.12** % | **-2.11** % |

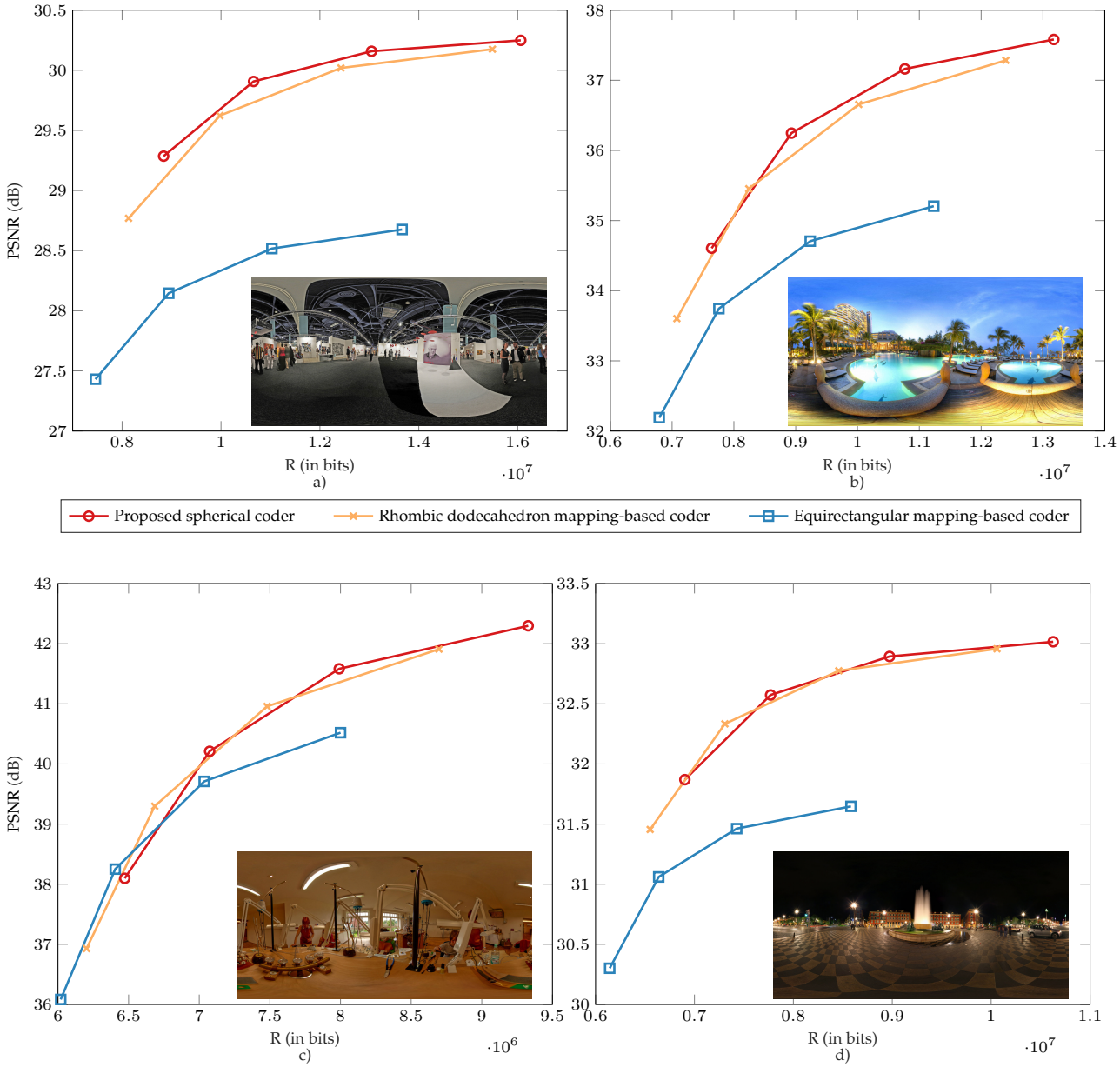



Figure 4.12 – RD curves of the proposed spherical coder against the baselines. (a) Image Exhibition. (b) Image Pool. (c) Image workshop. (d) Image Plaza.

### 4.2.4 Conclusion of the results for the proposed spherical coder

In this section, we proposed a new coder for 360° images which performs the compression directly on the sphere. Based on a uniform sampling of the sphere, our coder extends the regular tools of image compression to the spherical domain. Experiments demonstrate the benefits of such an approach, in particular with respect to conventional non-uniform sampling methods, and also based on a uniform sampling method which is mapped to several 2D image planes for compression. To show the efficiency

of the representation, we performed compression on the whole 360° image, but the proposed representation can be used in the interactive coder proposed in the previous section.

## 4.3 Geometry-aware compression of 3D mesh texture

The second imaging modalities we consider in this dissertation is the texture map of 3D models. The first challenge in this regard is that adjacent parts of the 3D mesh might not be adjacent in the associated 2D texture maps, even though the texture on the surface of the object seems continuous. We propose a geometry-aware prediction that takes into account the geometry information to predict a block from its neighbors in 3D space. We then explain the strategies used to place the access blocks, and the scanning order used to decode the requested blocks. *We assume that the mesh geometry is already coded and available at the decoder and we use it to find more relevant references for texture prediction.*

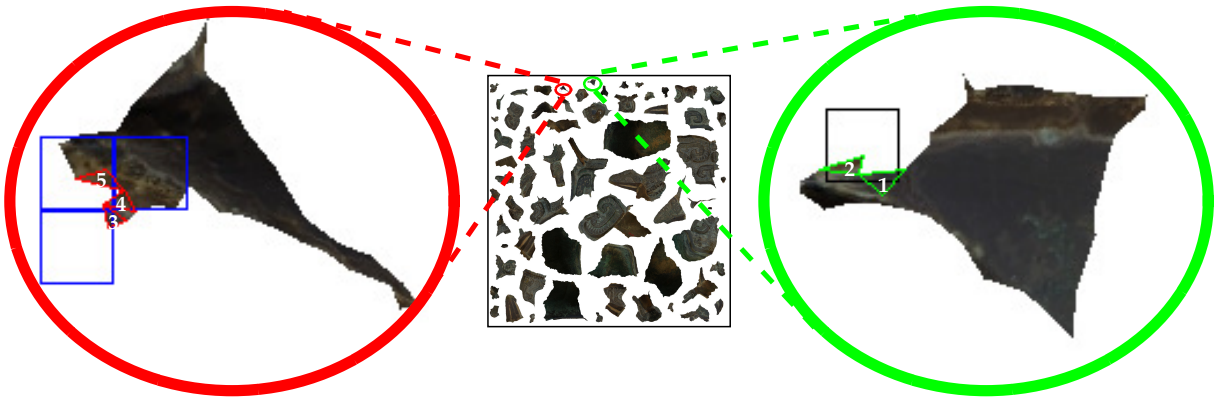
### 4.3.1 Neighborhood construction and prediction generation

Consider a block $k$. The neighborhood construction consists in finding the set $\mathbb{J}_{(k)}$ of the adjacent block's indices. In a classical 2D image, $\mathbb{J}_{(k)}$ is simply composed of the $4$-connected neighbor indices placed on the previous/next column/row. However, some textural information that are neighbors in the 3D model can be split into distant blocks in the atlas (see links A, B, and C of Fig. 3.2c). Thanks to the mesh topology, we retrieve this proximity and describe it in $\mathbb{J}_{(k)}$. This is illustrated in Fig. 4.13.

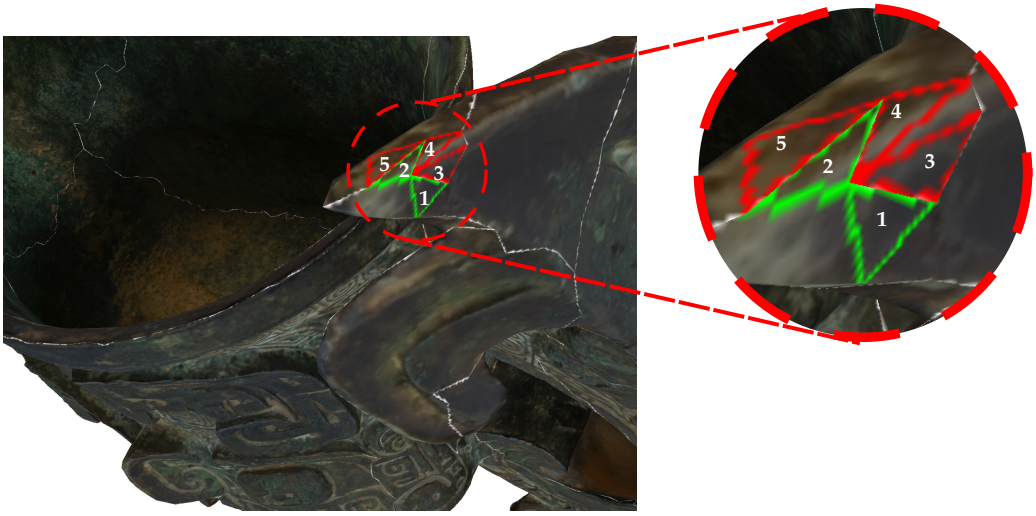
For the *boundary blocks*, i.e., blocks located on patch borders, the conventional intra prediction approach is not able to distinguish between informative and non-informative parts of a texture atlas. Therefore, its use for texture compression produces poor prediction due to the inaccuracy of non-informative references which results in increasing the compression rate. To address this problem, the topology of the mesh is used to provide more correlated references for intra prediction. For that, discrete patches on texture atlas are mapped to continuous 3D texture using mesh triangles, as shown in Fig. 4.14(a). The algorithm proposed here uses this information to fill the empty set of references when the block is positioned on a patch border (Fig. 4.14(b)).

For this purpose, first, the neighbors of a boundary block are detected using the mesh connectivity. Then, if they are already coded, their border pixels are copied to fill the references of the current block. Such neighbors may be far apart in the 2D texture atlas, yet, adjacent on the 3D mesh. Fig. 4.14(c) shows how the proposed algorithm copies the inner border of the reference triangles on the outer border of the other triangle. Barycentric coordinates [26] between the two triangles are used to find the corresponding points.

The references that were replaced on the border of the patches with such an approach are irregularly positioned. This is in contrary with the conventional intra prediction algorithm, where the references are regularly placed on the top row and left column. This makes the conventional block prediction challenging. For instance, the block prediction algorithm of HEVC performs a backward projection along with a linear interpolation on the reference pixels in order to produce the prediction values within the block [45]. The formulation of this process requires the reference pixels to be aligned in one row and one column.




(a)




(b)

Figure 4.13 – Retrieving the neighborhood information using the mesh topology. (a) Triangles 1 and 2 in the black block (the right patch in green circle) are neighbors to triangles 3, 4, and 5 in blue blocks (the left patch in red circle). As can be seen from the image in the middle, these two patches are far from each other in the texture map. (b) The corresponding triangles are neighbors of each other on 3D model (the seam between patches are shown in white). To predict the black block with triangles 3, 4, and 5, the blue blocks must be decoded before the black block. More precisely, if we denote the black block by $k$ the *combination* of the three blue blocks are in $\mathbb{J}_{(k)}$ as an element of this set.

To address the above problem, our algorithm integrates a forward reference projection, instead of the regular backward one. This process is shown in Fig. 4.15 and consists of two steps: 1) forward directional projection of reference pixels with a specified angle on the non-integer positions in the block, 2) interpolation of integer positions either with a simple nearest neighbor approach or with bilinear interpolation of four non-integer neighbors. For comparison, Fig. 4.15 also shows the resulting prediction with the conventional intra prediction algorithm on a boundary block.

In addition to the prediction function described above, for the sides of the block which are in the informative region, the corresponding SI prediction generated with reference pixels depicted in Fig. 4.2 is added to the possible set of SI (see Section 4.1.2 for more details). Therefore, for the non-boundary

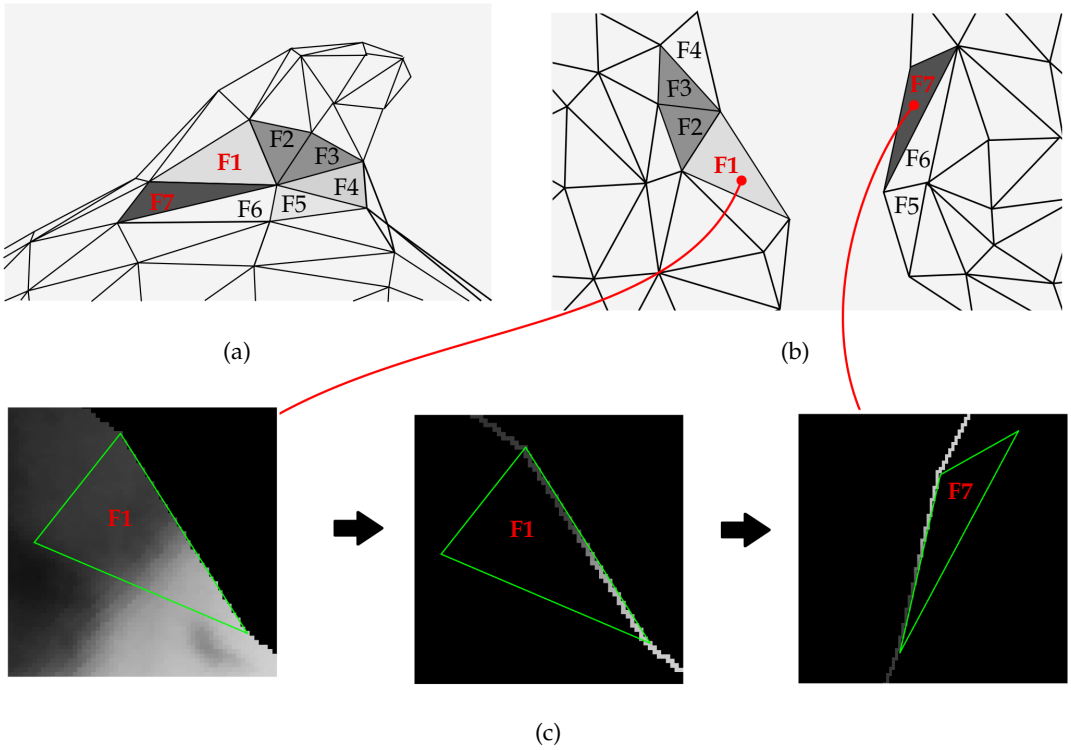
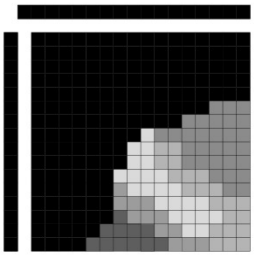



Figure 4.14 – Process of copying border references from remote neighbors in texture map of 3D models. A 3D mesh in (a) is mapped into a 2D atlas in (b). According to this mapping, the block F1 can be used as the remote neighbor of F7, hence, its interior border is copied to the exterior border of F7 (c).

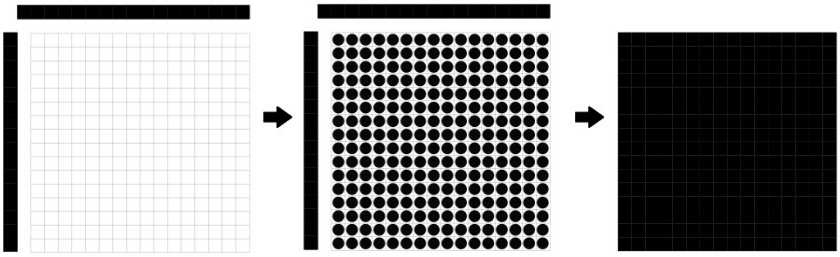
blocks, in which all 4 surrounding sides of the blocks are available, all 12 SI predictions are considered in the set of possible side information. These predictions corresponds to different decoding order that might appear at the decoder side.

After exploiting the inter-block dependencies with the above-mentioned prediction approach, in order to compact the signal energy on a sparse basis and exploit the redundancies within the block, a transformation is then performed on each block and each of its prediction blocks. Blocks of the texture image differ from the classical 2D image block because for a boundary block which lies on a patch edge, only a subset of pixels is informative. To address this problem, we exploit the mesh connectivity to find the mask of informative pixels. Then, boundary blocks and their predictions are transformed using GFT [80] as follows:

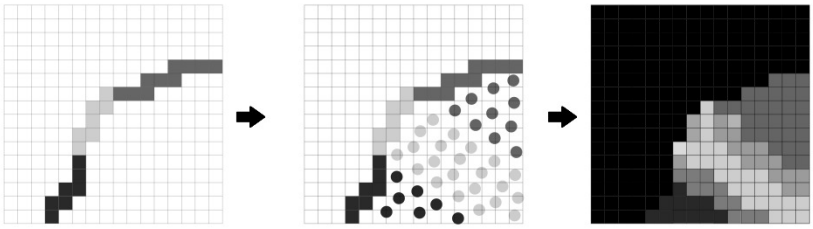
Let us denote the size of a block by $B \times B$, and $\mathbf{M}$ denote mask of informative pixels in the block such that $\mathbf{M}(u, v) = 1$ indicates pixel $(u, v)$ of the block is informative. We form an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E}, \mathbf{A})$ from the mask in which the set of nodes $\mathbb{V}$ in the graph represents informative pixels. $\mathbb{E}$ is the set of edges which is formed by using 4-connected neighboring. There is an edge between a pixel and its neighbor if both are informative. The adjacency matrix $\mathbf{A}$ is constructed from the edges with $\mathbf{A}(i, j) = 1$ if and only if there is an edge between pixels $i$ and $j$ where $i, j = 1, ..., N$ and $N$ represents number of informative pixels in the block. The graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is then calculated in which degree matrix $\mathbf{D}$ is a diagonal matrix whose diagonal element $D(i, i)$ is $D(i, i) = \sum_j \mathbf{A}(i, j)$. By



a) Original block with surrounded pixels.



b) Conventional intra prediction.



c) Proposed intra prediction with irregular references.

Figure 4.15 – Block prediction of a boundary block using the proposed geometry-aware compression scheme. Left: set of reference samples. Middle: propagation of reference samples (reference projection). Right: intra prediction (interpolation).

construction, $\mathbf{L}$ is symmetric positive semi-definite and has a complete set of orthonormal eigenvectors $\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_N]$ with non-negative eigenvalues $\lambda_1, ..., \lambda_N$. The graph Fourier basis is defined as the Laplacian eigenvectors. The GFT of a signal $\mathbf{s} \in \mathbb{R}^N$ is its projection on the eigenvectors given by $\mathbf{U}$

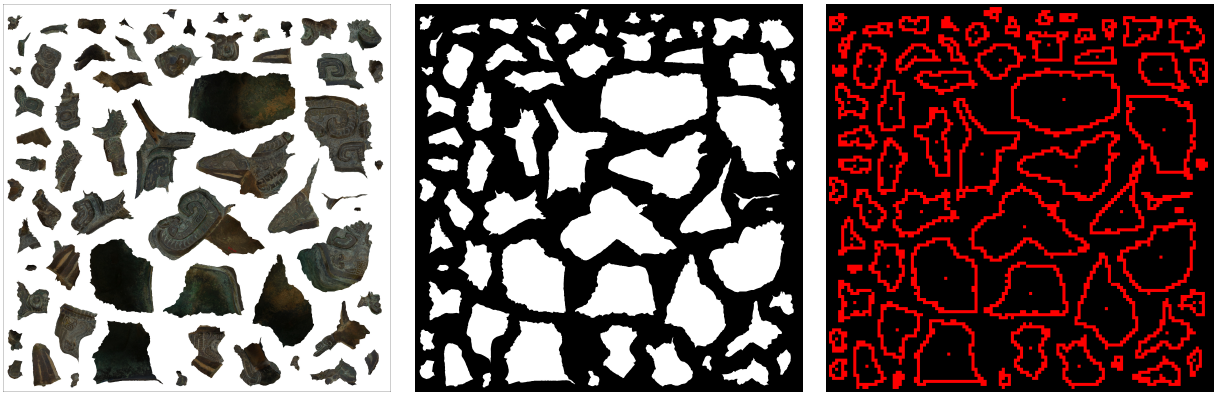
$$\hat{\mathbf{s}} = \mathbf{U}^T \mathbf{s}.$$

For the non-boundary blocks in which all pixels are informative still DCT is used as a GFT with full 4-connected neighboring is equivalent to DCT. The transformed values are then quantized using a uniform quantizer.

### 4.3.2 Placement of access blocks

As explained in Section 3.4.2, the access blocks must be located such that there exists at least one access block within the requested region. We determine all boundary blocks as access blocks. The main

reason for that is because sometimes more than one block in other patches may be needed to decode a boundary block (see Fig. 4.13), and it is highly likely that at least one of these blocks has not yet been decoded. Therefore, to prevent failure in decoding, all boundary blocks are referred to as access blocks. The central block of each patch is also designated as the access block to ensure movement within the patch. An example of access block placement is shown in Fig. 4.16.



(a) (b) (c)

Figure 4.16 – An example of access blocks localization for texture maps of 3D model. (a) Texture map. (b) Mask of informative pixels. (c) the positioning of access blocks (in red) after partitioning the texture map into blocks of size 32x32.

### 4.3.3 Decoding order and navigation

Due to patch discontinuities, the requested blocks also do not fit into one connected component (see Fig. 1.16). The proposed scanning process is as follows: First, we apply a connected component analysis on the requested blocks to provide a list of connected blocks with the 4-connected neighborhood. We then start from the connected component that has an access block in it. Starting from the access block, we scan all blocks in the same connected component using Algorithm 2. Once all blocks in the requested connected component have been processed, the algorithm explores the other connected components. Two possibilities may happen: (i) At least one boundary block in the other connected components can be decoded with already decoded blocks. In this case, we decode this boundary block and then the remaining blocks in the same connected component are processed using Algorithm 2. (ii) If no boundary block in any of the connected components could be decoded, since all boundary blocks are also access block, we select one of these blocks and decode it independently. We repeat the process for the remaining connected components. Note that this process only requires the geometry of the mesh which we assumed is decoded before decoding the texture map. Therefore, the same scanning can be performed at decoder without any need to send the order of block scanning.

For the later requests of the same user, some blocks are already in memory. In that case, instead of starting by the access block, the decoding order begins with any of these blocks in the memory and then proceeds as explained above.

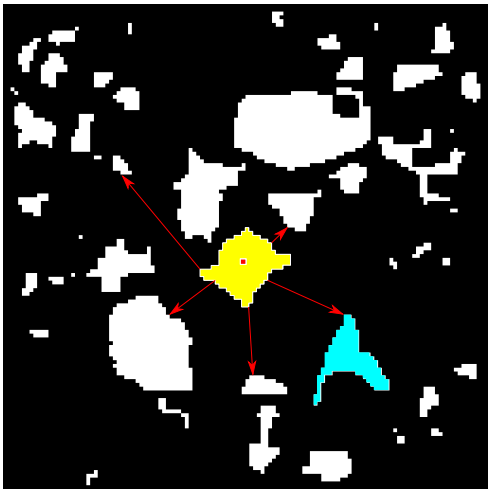


Figure 4.17 – Proposed scanning order for texture map of 3D models. The non-black blocks are the ones requested by the client. First, a connected component to be decoded is selected (here in yellow). This connected component is the first to be decoded because some blocks of it either are already available at the decoder (previous request), or an access block (in red) is available inside it that can be decoded independently of the other blocks. Then, the whole yellow patch is decoded. Similarly, the scanning will explore all other connected components one after the other.

### 4.3.4 Experimental results

In this section, we first show the efficiency of the proposed geometry aware representation by compressing the whole texture image of the 3D models. We then examine our interactive compression using all proposed tools.

#### 4.3.4.1 Compression of the whole image

We compress the whole image in a predictive coding manner. This way we have only on prediction per block. We use the same scanning algorithm proposed in Section 4.3.3. More precisely, we assume all informative blocks are requested by the user. We start from the central block of the biggest connected component and encode this block independently. We then continue the scanning as explained in Section 4.3.3 and encode blocks within the same connected component using regular intra prediction with available sides (see Fig. 4.2). Whenever scanning within a connected component is finished, we search for the border block of the other connected components that can be predicted using the decoded blocks until now with the prediction function proposed in Fig. 4.15. For encoding each block, instead of applying GFT on the signal and its prediction separately, we apply GFT on their residual. Finally, the transformed residual is quantized and encoded with an arithmetic coder.

The input data comes from [84], which consists of three ancient objects with different maps ( normal, diffusion, etc.). We call them Panel, Ewer, and Vase (see Fig. 4.18). Here, only the diffusion map is encoded which represents the “natural" texture of the object. The proposed Geometry-Aware (GA) compression algorithm is compared against the conventional intra prediction as the anchor, in terms of BD-R [8]. In this case, the anchor algorithms are encoded using normal raster scanning that scans

all blocks in the image (informative and non-informative). Two configurations are considered for the anchor algorithms. The first configuration (Anc-1) applies GFT on the residuals of the border blocks. The second anchor (Anc-2) performs similarly to Anc-1, except that DCT is used instead of GFT on the border blocks. To have a fair comparison, PSNR is calculated only on the informative pixels of the atlas. The results are provided as the average BD-R gain over different block sizes, namely 16×16, 32×32 and 64×64.

Table 4.4 summarizes the BD-R gain of the proposed GA algorithm against the two anchor methods. Each negative value indicates the percentage of compression gain achieved by the "GA" configuration against the "Anc" configuration of the corresponding column, when applied on the 3D sample of the corresponding row. Moreover, Fig. 4.18 demonstrates the rate-distortion curves of different configurations on each test sample. As can be seen, a promising compression gain can be achieved by using the proposed geometry-aware intra coding tools.

Table 4.4 – Performance comparison of the the proposed GA algorithm against the two anchors in terms of BD-R (%). For each column the corresponding anchor is used as the reference.

| | Anc-1 | Anc-2 |
|---|---|---|
| Panel | -16.1% | -29.6% |
| Ewer | -11.0% | -23.2% |
| Vase | -17.3% | -19.4% |

#### 4.3.4.2 Interactive compression of mesh texture

In this experimental section, we compare our interactive coder with two baselines. The first one encodes the whole atlas with predictive coding using the GA approach proposed in the previous section. We call it *Whole Atlas (WA)*. This method is very competitive for compression, and thus minimizes the storage $S$, but requires, for every user's request, to transmit the entire atlas. Another baseline is called *All Intra (AI)*, and corresponds to a method that is interactive, but is not able to take into account the correlation between the blocks. In other words, the method AI encodes each block independently as JPEG [101] does. For a fair comparison, the graph-based transform is performed in the method AI (as in ours and WA) by taking into account the mask of informative pixels.

We used two 3D objects in our experiments, one is the Ewer model [83] used also in the previous section (with texture size of 4096x4096), and the other one named Tombs [47] (with texture size of 8192x8192). The objects are depicted in Fig. 4.19. We compress these two models with the three coding schemes leading to three storage costs: $S_{\text{ours}}$, $S_{\text{WA}}$ and $S_{\text{AI}}$. We also asked three users to navigate around each model, each lasts for 30 requests corresponding to successive changes of camera orientations and positions. For our proposed method, we consider the theoretical results and the results that can be obtained with Protograph-Based rate-adaptive LDPC codes [109]. In Fig. 4.20, we plot the accumulated transmission rates at a PSNR of 38dB for the two models during successive requests, i.e.

$$R(T) = \sum_{t=1}^{T} r(t).$$

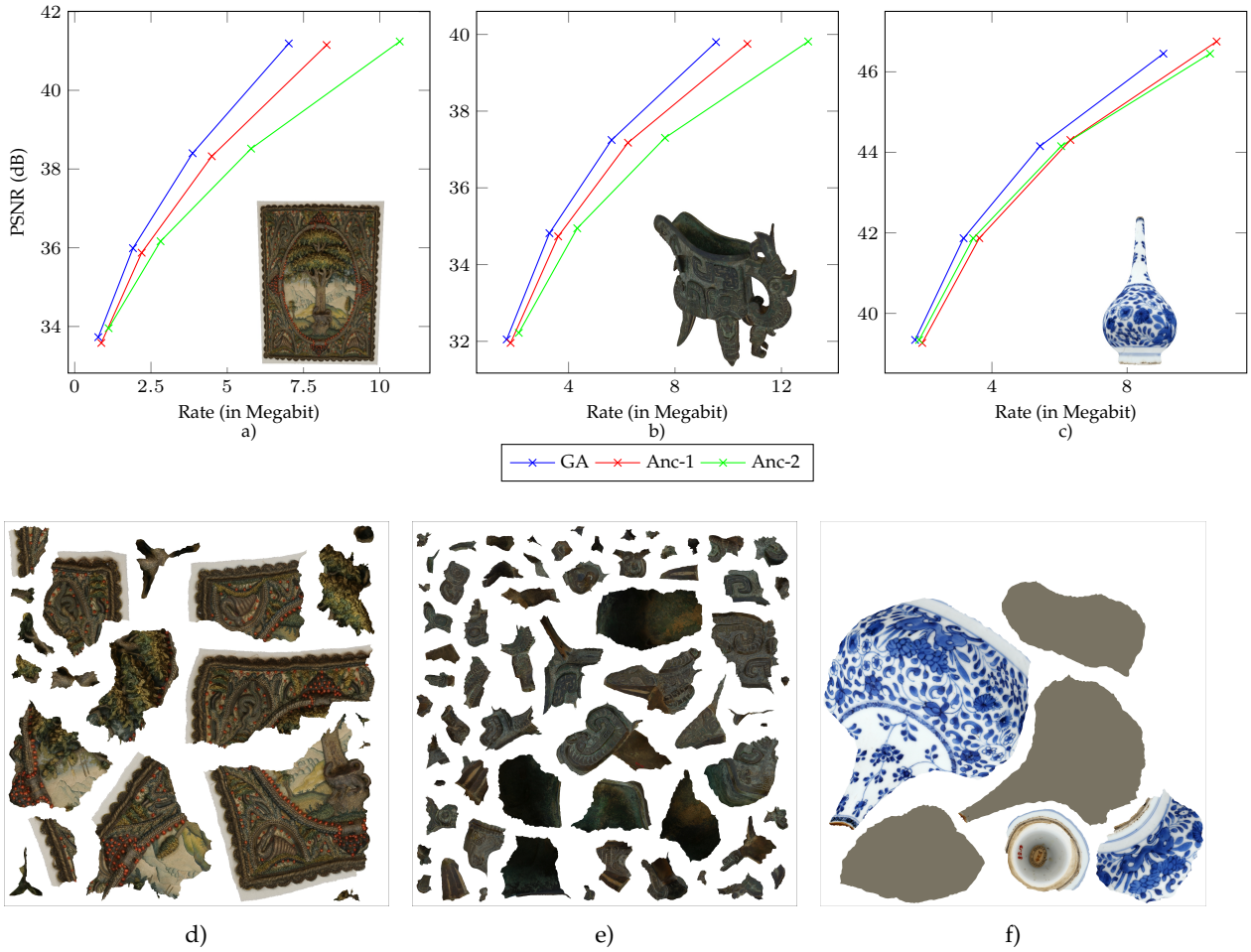



Figure 4.18 – RD curves of the proposed geometry aware compression algorithm against two configurations of the anchor (Anc) on 3 diffusion maps. First row shows the RD performance. Second row shows the corresponding texture map of each model. Left column is model Panel. Middle is for model Ewer. Right column is for model Vase.

Similar behaviors were observed for other PSNR values.

We can see that the theoretical results of our proposed method always performs better than all other schemes for two reasons. First, our scheme always has the smallest transmission cost (for the same quality), which demonstrates the great compactness of the compression. Second, we can see that $R_{\text{ours}}(t)$ evolution is smooth with the request which shows that our scheme transmits what is needed at every instant, and not everything at the beginning of the navigation as the scheme WA does. This is also the case for the scheme AI, but with a higher rate. However, in practice, the sub-optimality of the adopted rate-adaptive coder results in performance deficiency compared to AI which shows room for improvement by designing better rate-adaptive codes. Moreover, we notice that for a very large 3D model such as Tombs, as shown in the bottom row plots of Fig. 4.20, user's do not request to view the whole mesh and as a results there is a large gap in the transmission rate between the proposed method and the WA approach which transmits the whole image in the beginning. This implies that the idea of

gradual data transmission is perfectly reasonable for large 3D models. For small 3D model like Ewer, the top row plots in Fig. 4.20 shows that two users request the whole model after about twenty requests, and in this case, WA method performs better for long-time requests.

(a) (b)

Figure 4.19 – 3D models used for experiments of interactive compression. (a) Model Ewer. (b) Model Tombs.

The rate results demonstrate the benefits of our approach. This comes however with a small storage size overhead. In order to evaluate its impact, we rely on a comparison methodology proposed in Section 2.3.3, where the Bjontegaard metric is computed on the curve $(PSNR, \mathbb{E}_\theta(R_\theta)+\lambda \cdot S)$, for different values of $\lambda$. Results are shown in Table 4.5. We can see that when $\lambda = 2$, meaning that storage matters more than transmission, the best method is the WA. This value corresponds to the size's increase on the server. In other words, both All Intra and the proposed method have a small storage overhead, but for smaller $\lambda$s, *i.e.*, when a trade-off between storage and transmission is desired (transmission cost is becoming important), gradual transmission of compressed data performs better.

Table 4.5 – Weighted BD averaged over all users relative to whole atlas approach

| | $\lambda = 2$ | | | $\lambda = 0.01$ | | | $\lambda = 1e^{-3}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | All Intra | Proposed Theoretical | Proposed Protograph | All Intra | Proposed Theoretical | Proposed Protograph | All Intra | Proposed Theoretical | Proposed Protograph |
| Ewer | 5.13 | 7.18 | 59.69 | -18.18 | **-20.03** | 19.63 | -24.42 | **-27.32** | 8.89 |
| Tombs | 1.17 | 3.27 | 53.81 | **-60.93** | -60.79 | -41.46 | -77.56 | **-77.94** | -66.97 |
| Average | 3.15 | 5.23 | 56.75 | -39.55 | **-40.41** | -10.91 | -50.99 | **-52.63** | -29.04 |

### 4.3.5 Conclusion of the results for the proposed geometry-aware compression of 3D model texture map

In this section, we propose a framework for compressing texture atlases of 3D models. Conventional image compression methods treat atlas of 3D objects simply as 2D signals and do not benefit from their geometry information. Here, a block prediction method with a proper reference selection from the 3D geometry is proposed with a block scanning strategy and an alternative transformation. Experimental results show that the proposed tools increase the coding efficiency of texture maps. We then extended

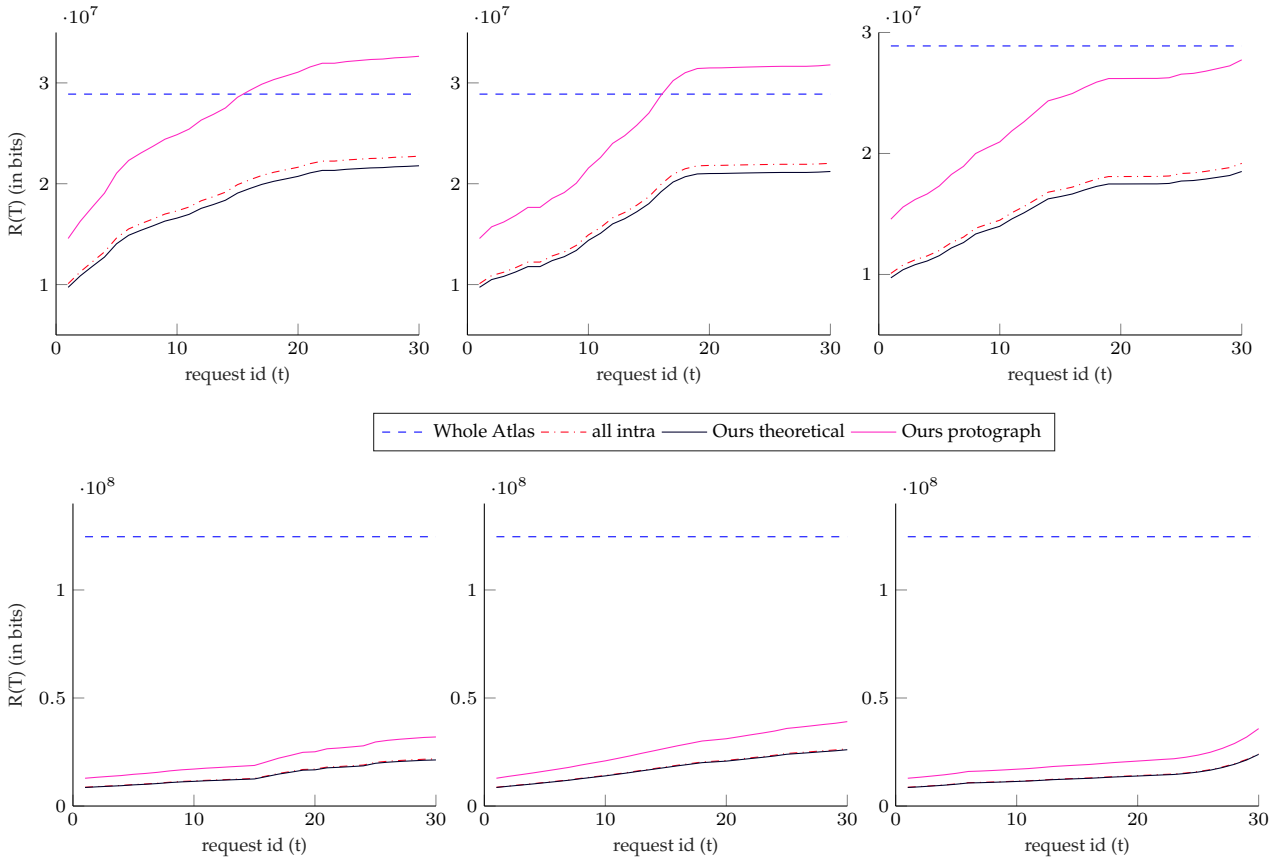



Figure 4.20 – Transmission rate evolution over the user's navigation for the 3D model Ewer at PSNR 38dB. First row represents the results for the Ever model and second row is for model Tombs. (Left column) User 1. (Middle) User 2. (Right column) User 3.

these tools with the help of generic coder proposed in Chapter 3 to provide interactive compression of mesh textures. By replacing classical predictive coding scheme with an incremental coding, and by adapting the intra-prediction and the block scanning order, we have enabled our coder to transmit only what is requested by a user when looking at the 3D model from some points of view. The experiments demonstrate that only the useful information is sent to users and that the compression efficiency obtained by the non-interactive coders is maintained.

## 4.4 Summary

In this chapter, we considered compression of two imaging modalities which are interactive by their nature: 360° images and texture maps of 3D models. We used the generic coding scheme of the previous chapter and designed efficient interactive compression scheme for each of these two modalities. For 360° images, we proposed a greedy algorithm to spread the access blocks all over the equirectangular image while their number is as minimum as possible to avoid increasing the storage size. Indeed,

the number of access blocks increases the storage and not the transmission, as extraction of the stored bistream for these blocks can be performed to adapt to the blocks known at the decoder. Considering the geometrical property of 360° images, we proposed novel prediction modes and efficient scanning of the blocks that treats the two vertical boundaries of the equirectangular images as if they are connected. A scanning order has been proposed to maximize the number of horizontal transitions. This is because horizontal blocks generate better predictions of their neighbors, which results in a lower transmission rate. Experimental results showed that the proposed scheme is perfectly suited for interactive transmission because it balances the trade-off between transmission rate and storage. The transmission rate increases gradually with the duration of the request and the system transmits only the blocks which are requested.

We also presented a better way to compress 360° images by proposing a spherical coder in which the whole encoding procedure is performed directly on the sphere. For that, the sphere is pixelated with a quasi-uniform sampling tool called HEALPix. We then introduced the notion of S-blocks on top of this pixelization, which are blocks defined on the sphere. The proposed scanning order, prediction, and transformation of S-blocks are adapted to the nature of the spherical signal. For that, the scanning starts from the north pole of the sphere and processes blocks ring by ring until it reaches the south pole. Then S-blocks are predicted based on previous scanned S-blocks. Finally, a GFT is proposed to transform the residual signal generated by subtraction of the prediction signal from the S-blocks. Experimental results show the efficiency of this approach compared to non-uniform sampling methods that map the spherical signal to 2D rectangular images.

Finally, we proposed solutions to encode texture maps of 3D models. The texture maps of 3D models consist of separate patches. To compress these textures, we proposed a new representation that uses the 3D geometry to find the neighbors. More precisely, a novel prediction tool is proposed that takes advantage of the topology of the associated 3D models. A block scanning strategy and a GFT are proposed to increase the performance of the coder. Experimental results demonstrate that this geometry-aware representation improves the coding efficiency compared to conventional coders that simply use raster scanning of blocks with conventional intra prediction. We then extend our coder to support interactivity. The placement of access blocks is discussed that enables to decode the blocks whatever the request of the user is. Again, same as before, the scanning algorithm of blocks and the prediction take into account the geometry to find the neighborhood. Experimental results confirm the benefit of using interactive compression for large 3D models.

CHAPTER 5

# MODEL DESCRIPTION COST EVALUATION FOR MODEL SELECTION IN INTERACTIVE CODING

The encoding and decoding operations of the data depends on the source statistics defined by probability distributions. The decoder can not estimate these distributions and it is necessary to transmit it to the decoder to recover the compressed data. In our interactive coder, it is generally costly to store and transmit the source distributions, even though it is crucial to make the decoding work in ideal condition. An alternative is to approximate the distribution by a parametric distribution and to send only its parameters, even if this results in an excess rate. Selecting the best approximate distribution requires to accurately estimate this excess rate. Although this estimation is straightforward for some source coding problems, it is still an open problem when channel codes are used for source coding as it is the case in our interactive coder. This chapter studies the estimation of this excess rate in practice by proposing a code optimization framework to design LDPC codes in the presence of mismatched decoding. We also study the excess rate estimation with rate-adaptive codes through doping technique. We experimentally show that the Kullback–Leibler (KL) divergence is a good measure to estimate the excess rate when LDPC codes with Belief Propagation (BP) decoding are used.

## 5.1 Motivation

Recall that in our interactive coder, each block is considered as a source to be compressed, while a set of predictions (corresponding to different possible navigations) is considered as SI to make the source more compact. The encoder encodes the source considering all these SI sources because it does not know which prediction will be available at the decoder. Let $X$ denote the block that needs to be encoded as a source and $Y_j$ denote its $j$-th SI source. The encoder applies a transformation and then quantization on the source and its corresponding SI sources. This results in pairs of length-n sequences of quantized values $(\boldsymbol{x}^n, \boldsymbol{y}_j^n)$ generated from sources $X$ and $Y_j$. Finally, the system uses the joint probability $P(\boldsymbol{x}^n, \boldsymbol{y}_j^n)$ for encoding/decoding procedure. These empirical probability distributions, often called type of a length-$n$ sequence in the community, play a key role in compression performance because the performance of the compression depends on the data distribution.

In our experiments (related to interactive compression) in the previous chapter, we assumed that

these distributions are known at both encoder and decoder for both the baselines and the interactive coder. However, in practice the decoder can not estimate the distribution (type) and it is necessary to transmit it to the user side for decoding. This transmission can be very costly, especially for images and videos because the dependency between the source and the SI is non-stationary and the corresponding distribution has a large alphabet. There are several ways to transmit the type. The naive approach is to transmit the frequency of the occurrence per any possible symbol in the set of source alphabet. In this case, if we denote the cardinality of the source alphabet by $|\mathcal{K}|$, $(|\mathcal{K}|-1)\log(n)$ bits are required to transmit the type to the decoder [22, Chapter 6] which is not negligible.

An alternative approach is to approximate the type with a parametric model and send only the parameters. However, this simplification imposes an excess rate on the compressed data. Therefore, it is necessary to solve a trade-off between the amount of data transmitted to describe the source distribution (model) and the excess rate imposed on the compressed stream due to using the model for compression. In this case, the model must be chosen such that it minimizes the sum of the description length of both the model and the compressed data [66]. In other words, when we describe a model with an approximate model, the number of bits we *save* in the model description must be greater than the number of bits we *lose* due to data compression using this approximate model. To perform this minimization and select the best model (not necessarily the true one), we need to characterize the coding length of the data according to any model or in other words, we need to know the effect of using a wrong model on the length of the compressed data (excess rate).

The objective of the chapter is *to characterize the excess rate* when a wrong model is used during the encoding and/or decoding procedures. We characterize the excess rate for different source coding schemes in Section 5.3. We show that for classical source coding, either with or without SI source, the excess rate is equal to the KL divergence between the true and the wrong model. However, when the code used for compression is a channel code (model-based coding), as in our interactive coder, the excess rate can be related to the capacity of a channel under mismatched decoding which is an open problem since the 1990s [55]. We conjecture that for some practical implementations, KL divergence can also be a good estimate of the excess rate. Therefore, as a contribution, we propose a whole framework to characterize this excess rate in practice. For that, as commonly adopted in interactive coding schemes, we use LDPC codes with BP decoding. These codes encode and decode the source with linear complexity in the block-length. In Section 5.4 we propose a two-step algorithm to design LDPC codes under BP and mismatched decoding. As a second contribution, in Section 5.5, we propose an algorithm to estimate the excess rate of a class of suboptimal LDPC codes that achieve rate adaptivity using doping. The contribution includes also an analysis of the performance of this type of code.

We focus our attention on binary LDPC codes. First, it is because they are more popular than symbol-based LDPC codes due to their lower complexity. Second, it is shown in [103] that when the correlations between the source and its SI sources are known in advance at the encoder, which is the case in interactive coding as well, binary-based LDPC and symbol-based LDPC perform equally.

It is worth noting that, maximum-likelihood decoding over the Binary Symmetric Channel (BSC) does not depend on the crossover probability since the decoding algorithm is equivalent to minimum Hamming distance. So, if the wrong model is another BSC, no excess rate will be observed even if the relative entropy is not zero. However, this conclusion holds for the maximum-likelihood decoder only

and not for the suboptimal BP decoder, considered here.

## 5.2 Notations and Definitions

Consider a lossless source coding problem for a discrete memoryless source $X$ with finite alphabet $\mathcal{X}$. At the encoder, the length-$n$ source sequences $x^n$ is compressed at rate $R$ bits per source symbol on average. The *fixed-length* encoding is a mapping function from $\mathcal{X}^n$ into an index in the set $\mathcal{M} = \{1, 2, .., 2^{nR}\}$ where each element in the set uses the same number of $R$ bits. On the contrary, a *variable-length* encoding functions assigns to each $x^n \in \mathcal{X}^n$ an encoded sequence whose length depends on the probability of $x^n$. More formally, the encoding function of variable-length coding transforms $x^n \in \mathcal{X}^n$ into set $\mathcal{I}^* = \{0, 1\}^*$ which represents all binary sequences of finite length (including length 0). For the rest of this chapter, whenever it is necessary to distinguish between fixed-length and variable-length coding, we denote the rate per symbol of fixed-length encoding by $R_f$ and the average rate per symbol of variable-length decoding is denoted by $R_v$. The decoding error probability is defined as $P_e^n = P\{\hat{X}^n \neq X^n\}$. A decoding function assigns an estimation $\hat{x}^n(M)$ or an error message $e$ to each element which is in the range of encoding functions, i.e. each elements of $\mathcal{M}$ or $\mathcal{I}^*$.

Throughout this chapter variables $X, Y$ are used for source coding problem and variables $V, U$ are used to present its dual channel coding problem. Assume $\mathcal{V} \subseteq \mathbb{R}$, a binary input ($\mathcal{U} = \{-1, 1\}$) memoryless channel $p_{V|U}(v|u)$ is symmetric if $p_{V|U}(v|u) = p_{V|U}(-v| - u)$ for all $u \in \mathcal{U}$ and $v \in \mathcal{V}$. A symmetric channel satisfies consistency condition if $p_{V|U}(v|u) = e^{-v} p_{V|U}(v| - u)$. $\log$ is of base 2, unless otherwise stated.

## 5.3 Theoretical excess rate under mismatched decoding

### 5.3.1 Source coding with helper

Fig. 5.1 depicts the source coding schemes considered in this thesis. It has been proved that a single source $X$ with distribution $P_X(x)$ can be compressed at rate $H_P(X)$ [78]. In general, the existence of an SI source helps to decrease the rate of the compression. For instance, in conditional (predictive) coding, the source $Y$ is available as a helper during both encoding and decoding operations which reduces the achievable compression rate to the conditional entropy $H_(X|Y)$ (see Section 1.1). By contrast, in all types of model-based coding, there is no explicit reference to the SI realization at the encoder. More precisely, although an SI source is available at the decoder, at the encoder there is either no SI or a set of possible SI sources is available (the encoder doesn't know which one would be available at the decoder). Therefore, instead of relying on SI realization, the encoder must rely on a correlation model between $X$ and $Y$.

Model-based coding covers different variety of source coding problems. The problem of source coding with SI only at the decoder, namely DSC, was first introduced by Slepian and Wolf in their seminal work [82]. They showed that it is possible to have lossless compression and encode correlated sources $(X, Y)$ separately at optimal rate $R = H(X|Y)$, provided that joint decoding of the sources is performed at the decoder. In other words, one can compress correlated sources separately at rates no larger than those needed when they are compressed jointly. In compound conditional coding [27], a

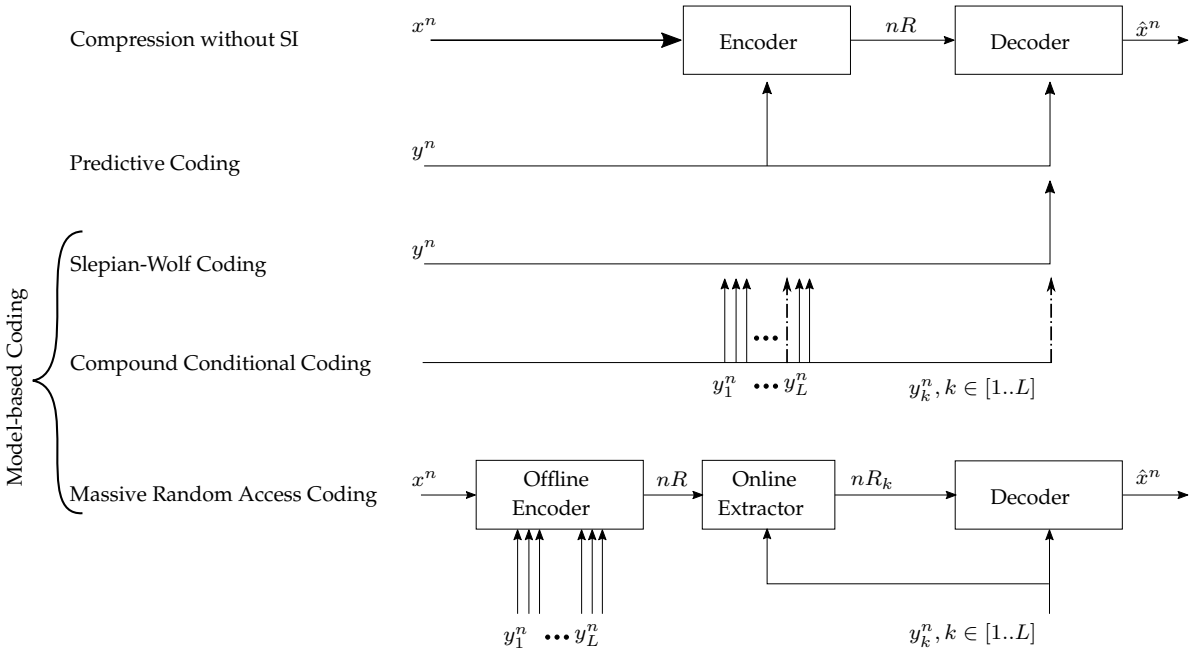



Figure 5.1 – Different source coding schemes based on the availability of SI source at encoder and decoder. The objective is to compress length-$n$ sequence $x^n$ into $nR$ bits, where $R$ is the rate of the compression.

possible set of $L$ SI realizations $y_1^n, y_2^n, ..., y_L^n$ are available to the encoder and only one of them, $y_k^n$, is observed at the decoder. The encoder doesn't know which SI is available at the decoder. In this coding scheme the source is encoded with respect to the worst-case correlation at rate $R = \max_i H(X|Y_i)$ where $i \in [1..L]$. SMRA is the extended version of compound conditional coding which distinguishes the storage and transmission rate [29]. Here as in compound conditional coding, the source is compressed and stored with respect to the worst case assumption, but during the transmission, the system can send the exact amount of data which is needed for decoding considering the SI source available at the decoder.

### 5.3.2 Problem formulation

There is an implicit assumption in all these source coding scenarios: the encoder and the decoder both know all distributions. However, this hypothesis is often ruled out in practice and mismatched decoding happens. This can occur either due to the fact that in practice it is costly to transmit the type of the sequence in universal coding or because of the constraints to have low complexity decoding. Therefore, it is of considerable interest to study the impact of using the wrong distribution at the decoder on the compression rate. To be able to derive theoretical results, for the rest of this chapter we assume vanishing decoding error probability is required and the length of the sequence goes to infinity.

Note that the final rate of the code contains both contribution to the data and the model description. However, the excess rate computed here is related to the data only, as the description length of the model can be directly derived from the number of parameters used. Therefore in the following, we only

consider the computation of the excess rate needed to compress the data.

### 5.3.3 Achievable rate for the excess rate under mismatched decoding

For the sake of completeness, we first consider the classical problem of compressing a single source $X$ which will be a basis for the derivation of new results in the case of source coding with a helper. In variable length coding, if instead of true distribution $P_X$, a wrong distribution $Q_X$ is used to encode and decode the data, [21, Theorem 5.4.3] shows that the average sufficient and necessary rate for lossless compression is $H_P(X) + D_{KL}(P_X||Q_X)$, where $D_{KL}(P_X||Q_X)$ is the KL divergence from $P_X$ to $Q_X$. Therefore, for *variable length codes*, the excess rate is:

$$D_{KL}(P_X||Q_X). \tag{5.1}$$

The next theorem studies the performance of a *fixed length code* under mismatched decoding, when a single source $X$ is to be compressed.

**Theorem 2** (Wrong model excess rate for fixed-length classical source coding without SI)**.** *Let $X$ be a discrete i.i.d. source with unknown distribution to be compressed losslessly and decoded with parametric decoding metric $Q_X$. Consider an encoder that estimates the type of the source $P_{x^n}$ and then builds a codebook for a fixed-length code based on both $P_{x^n}$ and $Q_X$. The codebook depends merely on the parameters of $Q_X$, denoted by $\Theta_Q$, and the rate. The encoder sends to the decoder the $\Theta_Q$, the rate used to encode the sequence and an index of length $nR_f$. The decoder builds the same codebook as the one at the encoder from $\Theta_Q$ and the rate $R_f$ and finally estimates the input sequence from the received index. The excess rate due to using wrong decoding metric $Q_X$ instead of type $P_{x^n}$ is:*

$$\Delta R_f = D_{KL}(P_{x^n}||Q_X).$$

Note that KL divergence as the excess rate holds even if the true distribution $P_X$ of the source $X$ (rather than type $P_{x^n}$) is known at the encoder, but not at the decoder. In this case, the excess rate in Theorem 2 becomes

$$\Delta R_f = D_{KL}(P_X||Q_X). \tag{5.2}$$

Remark: Since the type converges to the true distribution [21, Theorem 11.2.1], we will denote $P$ for both the type and the true distribution, without loss of generality.

*Proof.* *Achievability.* For every $R_f > R_f^*$ , where $R_f^* = H_P(X)$, there exists a sequence of codes with $\lim_{n\to\infty} P_e^n = 0$.
Code construction: Let us first consider the set

$$A^n_{\epsilon_{R_f,Q}}(X) = \{x^n : |-\frac{1}{n}\log Q(x^n) - R_f| < \epsilon\} \tag{5.3}$$

and label each element of this set with a different index $M$. Assign $M = e$ to all $x^n \notin A^n_{\epsilon_{R_f,Q}}(X)$. If

$R_f = H_P(X) + D_{KL}(P||Q)$, we can derive

$$\begin{aligned} 1 &\geq \sum_{\substack{x^n \in A_{\epsilon_{R_f,Q}}, \\ X^n \sim P_X(X^n)}} Q(x^n) \\ &\geq \sum_{\substack{x^n \in A_{\epsilon_{R_f,Q}}, \\ X^n \sim P_X(X^n)}} 2^{-n\left(H_P(X)+D_{KL}(P||Q)+\epsilon\right)} \\ &= 2^{-n\left(H_P(X)+D_{KL}(P||Q)+\epsilon\right)} |A^n_{\epsilon_{R_f,Q}}| \end{aligned}$$

where the second line follows from (5.3) and $|A^n_{\epsilon_{R_f,Q}}|$ represents the cardinality of the set $A^n_{\epsilon_{R_f,Q}}$. Therefore $|A^n_{\epsilon_{R_f,Q}}| \leq 2^{-n\left(H_P(X)+D_{KL}(P||Q)+\epsilon\right)}$ and $n(H_P(X) + D_{KL}(P||Q) + \epsilon)$ bits are sufficient to describe all these sequences without ambiguity. The encoder for mismatched decoding consists in assigning for each sequence $x^n$ its label. The decoder uses the receiving index $M$ and declares $\hat{x}^n = x^n(M)$ for the unique $x^n(M) \in A^n_{\epsilon_{R_f,Q}}(X)$, otherwise, if $x^n \notin A^n_{\epsilon_{R_f,Q}}(X)$, it returns an error.
Error Probability: Since $X_i$ are i.i.d., so are $\log Q(X_i)$:

$$-\frac{1}{n}\log Q(X^n) = -\frac{1}{n}\sum_i^n \log Q(X_i)$$

Hence, by the weak law of large numbers it converges *in probability* to

$$\begin{aligned} -\frac{1}{n}\sum_i \log Q(X_i) &\to -\sum_{x\in\mathcal{X}} P_X(x)\log Q(X) \\ &= -\sum_{x\in\mathcal{X}} P_X(x)\Big(\log P_X(x) - \log\frac{P_X(x)}{Q(x)}\Big) \\ &= H_P(X) + D_{KL}(P||Q) \end{aligned}$$

That is, for any $\delta > 0$,

$$\lim_{n\to\infty} Pr\Big(\big| -\frac{1}{n}\log Q(X^n) - H_P(X) - D_{KL}(P||Q)\big| > \delta\Big) = 0$$

setting $\delta = \epsilon$ one can conclude that as $n$ goes to infinity $P_e^n = Pr\{X^n \notin A_{\epsilon_{R_f,Q}}\}$ tends to zero.
Converse. if there exists a sequence of codes with $P_e^n \to 0$, then $R_f \geq R_f*$.
It has been shown that any sequence of variable length code with vanishing error probability satisfies $R \geq R_f^*$ [21, Theorem 5.4.3]. However, fixed-length codes are a particular case of variable-length codes. Therefore, any sequence of fixed-length code with vanishing error probability also satisfies $R_f \geq R_f^*$, which shows the converse. □

The proof of Theorem 2 admits the following interpretation: if we denote the set of length-$n$ typical sequences with respect to distribution $P_X$ with $A^n_{\epsilon_P}(X)$, by enlarging the set $A^n_{\epsilon_P}(X)$ to the set $A^n_{\epsilon_{R_f,Q}}(X)$ we can insure that all the true typical sequences in $A^n_{\epsilon_P}(X)$ are included in the set $A^n_{\epsilon_{R_f,Q}}(X)$ and the excess rate that is needed to be paid is $D_{KL}(P||Q)$.

For the case of source coding with SI, since in all scenarios the SI is available at the decoder the uncertainty in $Q_{X,Y}$ as the decoding metric can only caused by $Q(X|Y)$, thus $Q_{XY}(x,y) = Q_{X|Y}(x|y)P_Y(y)$.

**Corollary 1** (Wrong model excess rate for predictive coding)**.** *Let $P_{XY}$ be the joint distribution of a pair of i.i.d. sources $(X,Y)$ where $x^n$ is a length-$n$ sequence from source $X$ to be compressed losslessly and $y^n$ is the SI realization available at both encoder and decoder. Assume that the true distribution $P_{XY}$ and the parametric decoding metric $Q_{XY}$ are available at the encoder only. The codebook depends merely on the parameters of $Q_{XY}$ and the rate. The encoder sends both the parameters of $Q_{XY}$ and the rate such that the decoder is able to build the same codebook as the one at the encoder. For fixed-length and variable-length codes, the average excess rate due to mismatch decoding of using wrong distribution $Q_{XY}(x,y) = Q_{X|Y}(x|y)P_Y(y)$ is:*

$$\Delta R_f = \Delta R_v = D_{KL}(P_{XY}||Q_{X|Y}P_Y) \tag{5.4}$$

*Proof.* Let us first consider the problem of compressing the source $X$ for a given $y$ (a realization of source $Y$). This is similar to classical source coding with probability $P_{X|Y}(x|y)$. Thus, from (5.1), (5.2) and conditioning on $y$ we have:

$$\begin{aligned}\Delta R(X|y) &= D_{KL}(P_{X|Y}(x|y)||Q_{X|Y}(x|y)) \\ &= \sum_{x\in\mathcal{X}} P_{X|Y}(x|y)\log\frac{P_{X|Y}(x|y)}{Q_{X|Y}(x|y)}\end{aligned}$$

Averaging over all realizations $y \in \mathcal{Y}$ and adding the term $P_Y(y)$ to the numerator and denominator of $\log$ function we have:

$$\begin{aligned}\Delta R &= \sum_{y\in\mathcal{Y}} P_Y(y)\sum_{x\in\mathcal{X}} P_{X|Y}(x|y)\log\frac{P_{X|Y}(x|y)P_Y(y)}{Q_{X|Y}(x|y)P_Y(y)} \\ &= D_{KL}(P_{XY}||Q_{X|Y}P_Y)\end{aligned}$$

□

**Corollary 2** (Wrong model excess rate for Slepian-Wolf coding)**.** *Let $P_{XY}$ be the joint distribution of a pair of i.i.d. sources $(X,Y)$. Let $x^n$ be a length-$n$ sequence from source $X$ to be compressed at the encoder and $y^n$ the SI realization available only at the decoder. Under the type-level duality (nonlinear codes) the encoder sends the type of $x^n$ which is approximately $P_X(x)$ and $Q_{Y|X}(y|x)$ to the decoder and under linear codebook-level duality the encoder sends $Q_{XY}(x,y)$ to the decoder. Assume that $Q_{XY}(x,y)$ is the decoding metric ($Q_{XY}(x,y) = P_X(x)Q_{Y|X}(y|x)$ for type-level duality). The excess rate for the fixed-length and variable-length codes is related to the mismatch capacity of its dual channel with:*

$$\Delta R = C_M(P_{XY}, P_{XY}) - C_M(P_{XY}, Q_{XY}) \tag{5.5}$$

*where $C_M(P_{XY}, P_{XY})$ and $C_M(P_{XY}, Q_{XY})$ are the matched and mismatched capacities of the dual channel, respectively and their definitions depend on the type of duality (type-level or linear coodbook-level duality) [11, Theroem 1, Theorem 2].*

*Proof.* proofs can be found in [11, Theroem 1, Theorem 2]. □

Corollary 2 shows that the excess rate (5.5) for the Slepian-Wolf (SW) problem under mismatched decoding can not be derived directly as this is tantamount to deriving the mismatched capacity, which is still an open problem [76]. Another interesting issue, is that [11] shows unlike SW coding under matched decoding, (where linear codes suffice to achieve SW limit and its encoding rate is equal to nonlinear fixed-length and variable-length codes), the minimum rate achievable with linear codes under mismatched decoding is strictly higher than that achievable with nonlinear codes.

**Corollary 3** (Wrong model excess rate for SMRA coding)**.** *Let for all $i \in [1,..,L]$ the distribution $P_{XY_i}$ be the pair-wise joint distribution between source $X$ and SI source $Y_i$. Let $x^n$ and $y_i^n$ be the length-$n$ sequences generated from source $X$ (to be compressed) and SI sources $Y_i$, respectively. The encoder observes realizations $x^n$ and $y_i^n$ for all $i \in [1,..,L]$. There are many decoders and each can have a different SI $y_k^n$ where $k \in [1..L]$. Under the type-level duality the encoder sends the type of $x^n$ to the decoder which is approximately $P_X(x)$ and $Q_{Y_k|X}(y_k|x)$. Under linear codebook-level duality the encoder sends $Q_{X|Y_k}(x|y_k)$ to the decoder. Assume that $Q_{XY_k}(x,y_k)$ is the decoding metric $\Big(Q_{XY_k}(x,y_k) = P_X(x)Q_{Y_k|X}(y_k|x)$ for type-level duality and $Q_{XY_k}(x,y_k) = P_Y(y)Q_{X|Y_k}(x|y_k)$ for linear codebook-level duality $\Big)$. Let $(\Delta S, \Delta R_k)$ denote the tuple of excess rates (storage and transmission excess rates, respectively) under mismatched decoding metric $Q_{XY_k}(x,y_k)$ for the SI $Y_k$. For both fixed-length and variable-length codes the excess rates having $L$ SI source realizations $y_1^n, y_2^n, ..., y_L^n$ is equal to*

$$\begin{aligned} \Delta S &= \max_{j\in[1..L]} C_M(P_{XY_j}, P_{XY_j}) - \max_{i\in[1..L]} C_M(P_{XY_i}, Q_{X,Y_i}) \\ \Delta R_k &= C_M(P_{XY_k}, P_{XY_k}) - C_M(P_{XY_k}, Q_{XY_k}) \end{aligned} \tag{5.6}$$

*where $C_M(P_{XY_k}, P_{XY_k})$ and $C_M(P_{XY_k}, Q_{XY_k})$ are the matched and mismatched capacities of the dual channel, respectively and their definition depend on the type of duality (type-level or linear codebook-level duality) [11, Theroem 1, Theorem 2].*

*Proof.* The proof for matched compression of SMRA [29] relies on embedded random binning and the optimal rate under matched decoding for storage is $S^* = \max_{i\in[1..L]} H(X|Y_i)$, and the transmission rate when SI $y_k^n$ is available at the decoder is $R_k^* = H(X|Y_k)$ [29]. Due to the random binning code construction, the results are similar to SW coding and same duality between channel coding and SMRA coding can be derived. Therefore, one can use the excess rate results of SW coding under mismatched decoding (5.5). □

**Corollary 4** (Wrong model excess rate for compound conditional coding)**.** *Let for all $i \in [1,..,L]$ the distribution $P_{XY_i}$ be the pair-wise joint distribution between the source $X$ and the SI $Y_i$. Let $x^n$ and $y_i^n$ be the length-$n$ sequences of source $X$ (to be compressed) and SI sources $Y_i$, respectively. The encoder observes realizations $x^n$ and $y_i^n$ for all $i \in [1,..,L]$. The decoder observes only one of the SI sources $y_k^n$. Under the type-level duality the encoder sends the type of $x^n$ which is approximately $P_X(x)$ and $Q_{Y_k|X}(y_k|x)$ to the decoder. Under linear codebook-level duality the encoder sends $Q_{X|Y_k}(x|y_k)$ to the decoder. Assume that $Q_{XY_k}(x,y_k)$ is the decoding metric $\Big(Q_{XY_k}(x,y_k) = P_X(x)Q_{Y_k|X}(y_k|x)$ for type-level duality and $Q_{XY_k}(x,y_k) = P_Y(y)Q_{X|Y_k}(x|y_k)$ for linear codebook-level duality$\Big)$. For fixed-length and variable-length coding having $L$ SI source realizations*

*$y_1^n, y_2^n, ..., y_L^n$ the excess rate $\Delta R$ is equal to*

$$\Delta R = \max_{j\in[1..L]} C_M(P_{XY_j}, P_{XY_j}) - \max_{i\in[1..L]} C_M(P_{XY_i}, Q_{X,Y_i})$$

*where $C_M(P_{XY_k}, P_{XY_k})$ and $C_M(P_{XY_k}, Q_{XY_k})$ are the matched and mismatched capacities of the dual channel, respectively and their definitions depend on the type of duality (type-level or linear codebook-level duality) [11, Theroem 1, Theorem 2].*

*Proof.* Compound conditional coding is a special case of SMRA coding where the source will be compressed and stored with respect to the least correlated SI and the transmission rate is equal to the storage rate. Therefore, same proof as in Corollary 3 can be used. □

### 5.3.4 Suggestion for the missing formula

The estimation of the excess rate for model-based coding under mismatched decoding, mentioned in Corollaries 2, 3 and 4, is a difficult problem in general and it is related to the capacity of its dual channel. Unfortunately, the mismatched capacity is still an open problem and only some lower bounds are known and even the lower bounds are difficult to compute [36, 76, 55]. A commonly used approach to implement model-based coding in practice is to use linear LDPC codes with syndromes, in which the asymptotically optimal sequence of codes and the asymptotically optimal decoding scheme (e.g., maximum-likelihood or typical-set decoding) are replaced by LDPC codes and BP decoding, respectively. This is because in practice encoding and decoding with linear complexity in the block-length are desired. Since KL divergence fits for other source coding approaches, we propose a conjecture that, for the cases where the uncertainty in $Q_{XY}$ depends only on $Q_{X|Y}$ (same as the case of predictive coding), an approximation of the excess rate in the case of linear codebook might be the same KL divergence $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$ as in classical and predictive coding, Next section gives experimental evidences for the correctness of this formula using SW linear codes and BP decoding.

## 5.4 Achievable rate of linear codes under BP and mismatched decoding

The goal of this section is to provide an estimation of the mismatched capacity $C_M(P_{XY}, Q_{XY})$ mentioned in (5.5), (5.6), or equivalently, if we denote the corresponding dual channel variables by $V, U$, the goal is:

$$C(P_{V|U}, Q_{V|U}) = \max_{\mathcal{C}_n : P_e(\mathcal{C}_n, P_{V|U}, Q_{V|U}) \xrightarrow[n\to\infty]{} 0} R(\mathcal{C}_n) \tag{5.7}$$

where $P_{V|U}$, $Q_{V|U} : \mathcal{U} \to \mathcal{V}$ stands for the dual channel and the dual mismatch decoding metric (due to the duality [11]), $\mathcal{C}_n \subseteq \mathcal{U}^n$ represents the codebook, $v^n \in \mathcal{V}^n$ is the channel output, $R(\mathcal{C}_n) = \frac{1}{n}\log|\mathcal{C}_n|$ is the rate of the block channel code $\mathcal{C}_n$ and $P_e(\mathcal{C}_n, P_{V|U}, Q_{V|U})$ denotes its decoding error probability.

Since the mismatched capacity is an open problem, we estimate (5.7) for practical schemes where LDPC codes with BP decoding is used. To do so, in Section 5.4.1, we first transfer the source coding

problem to its dual channel code and formulate the code optimization problem (5.7) for LDPC codes. later in the same section, we review the existing optimization solutions (without mismatch), and propose a novel optimization solution that combines beneficially advantages of all these methods. The new optimization with mismatch consists in a two-step optimizer in which at each step the objective of the optimizer is to optimize the code parameter with respect to (5.7). In the first optimizer, we propose a way to have a first rough, but not far from optimum, estimation of the code parameters (Section 5.4.2), followed by a verification and refinement stage to ensure zero decoding error probability for this rough estimation (Section 5.4.3). Finally, in the second optimizer we optimize again this rough estimation with a more accurate method (Section 5.4.4).

It is worth noting that there exist researches which have studied the performance of LDPC codes with BP decoding of Binary Input Additive White Gaussian Noise (BIAWGN) in the presence of noise power estimation error [73, 72], However, these works have studied the problem from the channel threshold point of view, i.e. for a fix LDPC code they have analyzed the effect of using an incorrect estimation of noise power on the threshold. To the best of our knowledge, our effort is the first to study the problem from the excess rate point of view when BP decoding is used to design and decode linear codes for source coding problem under mismatched decoding.

### 5.4.1 Code optimization under mismatched decoding

It is well known that there is an intimate connection between channel coding and SW coding problems [105, 12, 102, 49]. The duality between channel coding and SW coding problem with general distribution structure has been established in [12] for linear coding framework. Using this linear codebook-level duality one can transform any general SW coding problem to its corresponding dual channel problem and expect the same decoding error probability from both problems under maximum likelihood or even BP decoding.

Suppose that we want to encode the length-$n$ sequence $X^n$ with decoder SI $Y^n$ coming from the joint probability distribution $P_{XY}$ under the decoding metric $Q_{XY}$. Optimal decoding rule achieves in matched decoding when $P_{XY}(x,y) = Q_{XY}(x,y)$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. Assuming input $U$ in the alphabet $\mathcal{X}$ and output $V = [V_1, V_2]$ in the alphabet $\mathcal{X} \times \mathcal{Y}$, the joint distributions are first converted to cyclic-symmetric channels $P_{V|U}$ (channel) and $Q_{V|U}$ (decoding metric) [11, 12], respectively, with

$$V_1 = U \oplus X, \quad V_2 = Y \tag{5.8}$$

where $U$ is independent of $(X, Y)$ and $\oplus$ denotes the addition in the finite field $\mathcal{X}$.

This linear codebook-level duality also holds when BP decoding algorithm is used for decoding [11]. Assuming an all-zero codeword, for a source distribution $P_{XY}$ under decoding metric $Q_{XY}$, let $m_0^Q$ denote the initial Log-Likelihood Ration (LLR) messages of the equivalent channel $Q_{V|U}$. For any binary input messages ($\mathcal{X} = \{0, 1\}$), this initial message and its associated probability distribution are

computed as:

$$\begin{aligned} m_0^Q(v=(v_1,v_2)) &= \ln \frac{Q_{V|U}(v=(v_1,v_2)|u=0)}{Q_{V|U}(v=(v_1,v_2)|u=1)} \\ &= \ln \frac{Q_{X|Y}(x=v_1|y=v_2)}{Q_{X|Y}(x=v_1\oplus 1|y=v_2)} \end{aligned} \tag{5.9}$$

$$P_0 = \sum_{v=(v_1,v_2)\in\mathcal{X}\times\mathcal{Y}}^{|\mathcal{X}|\cdot|\mathcal{Y}|} P_{X,Y}(x=v_1,y=v_2)\cdot\delta_{m_0^Q(v)} \tag{5.10}$$

where the superscript $Q$ in $m_0^Q$ indicates that the initial values are computed using the decoding metric $Q_{XY}$, $\delta_t$ is a Dirac delta function at point $t$ and $\oplus$ is the modulo-2 addition in the finite field $\mathcal{X}$.

Having the initial messages and their associated probabilities, which can be computed through (5.9) , (5.10), we are able to apply Density Evolution (DE) [65] to analyze the performance of the decoding for a specific LDPC code ensemble. Consequently, given any source distribution $P_{XY}$ and the corresponding decoding metric $Q_{XY}$, one can design capacity-achieving LDPC codes to encode $X$ with SI $Y$ based on the dual channel model using the existing algorithms [64, 74, 92]. Let us denote the degree distributions of the LDPC code ensemble for Variable nodes (VNs) and Check nodes (CNs) with $\lambda(x)=\sum_{i=2}^{d_v}\lambda_i x^{i-1}$ and $\rho(x)=\sum_{j=2}^{d_c}\rho_j x^{j-1}$, respectively, where the coefficient of $x^i$ represents the percentage of edges connected to nodes of degree $i+1$. The rate of the channel is equal to:

$$R\big(\mathcal{C}_n(\rho,\lambda)\big) = 1-\frac{\sum_{i=2}^{d_c}\rho_i/i}{\sum_{i=2}^{d_v}\lambda_i/i}. \tag{5.11}$$

Assuming infinite block length (cycle-free graph), two approaches exist to optimize the degree distributions: Traditional way to solve the code optimization problem is to fix the rate of the code and find the channel with the maximum threshold that can be decoded for this code rate [64]. This leads to

$$\max_{(\rho,\lambda):R(\mathcal{C}_n(\rho,\lambda))=R_0}\sigma^* = \max_{(\rho,\lambda):R(\mathcal{C}_n(\rho,\lambda))=R_0}\left[\max_{P_{V|U}:P_e(\mathcal{C}_n,P_{V|U})\xrightarrow[n\to\infty]{}0}\sigma(P_{V|U},\mathcal{C}_n)\right].$$

The second approach optimizes the rate instead of the threshold. There exist ways to solve directly the rate optimization

$$\max_{(\rho,\lambda):P_e(\mathcal{C}_n(\rho,\lambda),P_{V|U})\xrightarrow[n\to\infty]{}0} R\big(\mathcal{C}_n(\rho,\lambda)\big) \tag{5.12}$$

however, the difficulty is due to the fact that the decoding process (BP) is iterative, and that the vanishing error probability constraint is evaluated not only when the blocklength goes to infinity but also when the number of iterations of BP also goes to infinity. The infinite number of iterations makes this constraint be an infinite order polynomial of the code parameter $(\rho,\lambda)$. Solutions to this problem consist in either simplifying the analysis with extrinsic information transfer (EXIT) charts which yields to a linear constraint [92], or to expand the constraint into multiple linear constraints [74]. Both methods have drawbacks: the first one is not only too optimistic but also not always consistent, and the second one requires an accurate initialization. Therefore, we propose a way to solve the consistency problem

of the EXIT chart analysis and combine the EXIT chart based optimization (Section 5.4.2) with a novel channel hardening approach (Section 5.4.3) to provide an efficient initialization to start the optimization problem proposed in [74]. In the following, we detail our approach for the case of mismatch decoding, but note that this can also efficiently apply for the case without mismatch.

### 5.4.2 Rough solution with consistency or channel decomposition

An EXIT chart [91], is a technique which tracks the mutual information between the transmitted bit $U$ and the soft LLR messages $L$ corresponding to this bit. This mutual information can be computed as [91][71, chapter 9.6]:

$$I(U;L) = 1 - \sum_{u=\pm 1} \frac{1}{2} \int_{-\infty}^{\infty} p_{L|U}(l|u) \cdot \log_2 \Big( \frac{p_{L|U}(l|u=-1) + p_{L|U}(l|u=1)}{p_{L|U}(l|u)} \Big) dl \tag{5.13}$$

The technique relies on the Gaussian approximation of messages exchanged between VN and CN processors, in which the output extrinsic information of one processor is the input *a priori* information for the other one and vice versa. Here the same notation of [91] is used i.e. the mutual information between the extrinsic (a priori) information coming out of (into) a processor and the code bit associated with that processor is denoted by $I_E$ ($I_A$). In a BIAWGN channel, the input LLR messages have also Gaussian distribution and are consistent. Assuming consistency condition is also valid for other Gaussian messages exchanged between VNs and CNs, (5.13) for VNs will be simplified to:

$$I(U,L) = J(\sigma) = 1 - \int_{-\infty}^{+\infty} \frac{e^{-\frac{-(l-\frac{\sigma^2}{2})^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \log_2(1+e^{-l}) dl$$

Let $\sigma_A$ and $\sigma_E$ denote the standard deviations of the consistent-Gaussian distribution of the messages coming into and out of a VN of degree $d_v$, respectively. The extrinsic and a priori mutual information of the VN is equal to:

$$I_{AC} = J(\sigma_A) \text{ and } I_{EV} = J\Big(\sqrt{\sigma_{in}^2 + \sigma_0^2}\Big) \tag{5.14}$$

where $\sigma_0^2$ represents the standard deviation of the initial LLR message distribution and $\sigma_{in}^2 = (d_v - 1)\sigma_A^2$ is the variance of the Gaussian input messages coming from the neighboring CNs to VNs of degree $d_v$.

For non-Gaussian SW coding problems which have discrete LLR distribution as in (5.10), the output distribution of VN update with consistent-Gaussian input messages is indeed a mixture of Gaussian as:

$$p_{L|U}(l|u=+1) = P_0 \circledast \mathcal{N}(\frac{\sigma_{in}^2}{2}, \sigma_{in}^2) = \sum_{v=(v_1,v_2)\in\mathcal{X}\times\mathcal{Y}}^{|\mathcal{X}|\cdot|\mathcal{Y}|} P_{X,Y}(x=v_1, y=v_2) \cdot \mathcal{N}(\frac{\sigma_{in}^2}{2} - m_0^Q(v), \sigma_{in}^2)$$

where $\circledast$ represents the convolution operation. Therefore, when we have discrete output, the messages are no more consistent (they are still symmetric) and thus (5.14) is no longer valid for $I_{EV}$. We propose two solutions to compute $I_{EV}$ (extrinsic mutual information of VN), first by assuming that the distribution is still consistent, and second by decomposing the output distribution to BSC [46].

*1- Assuming consistency assumption:* Inspired by [69, eq. 35] one can assume the distribution is still

consistent, let us define the two variable function $J(\sigma, t)$ as follows:

$$J(\sigma,t) = 1 - \int_{-\infty}^{\infty} \frac{e^{-\frac{(l-\frac{\sigma^2}{2}-t)^2}{2\sigma^2}}}{\sqrt{2\pi}\sigma} \log_2(1+e^{-l})dl$$

the EXIT function for SW coding problem under decoding metric $Q_{XY}(x,y)$ of a source with joint distribution $P_{XY}(x,y)$ is equal to:

$$I_{EV} = \sum_{d=2}^{d_v} \lambda_d \cdot \sum_{\substack{v=(v_1,v_2) \\ \in \mathcal{X}\times\mathcal{Y}}}^{|\mathcal{X}|\cdot|\mathcal{Y}|} P_{XY}(v) J\Big(\sqrt{(d-1)[J^{-1}(I_{AV})]^2}, m_0^Q(v)\Big) \tag{5.15}$$

we then approximate this mutual information with Gauss-Hermite quadrature.

2- *Decomposition into BSCs:* A binary-input symmetric memoryless channel can be separated into sub-channels which are BSCs [46]. Since $p_{L|U}(l|u)$ is symmetric, using a quantizer we can decompose it to $(W+1)$ BSCs with intervals $0 < \zeta_0 < \zeta_1 < ... < \zeta_W < +\infty$. We have

$$\begin{aligned}
P_w &= \int_{\zeta_{w-1}}^{\zeta_w} p_{L|U}(l|u=+1)dl + \int_{-\zeta_w}^{-\zeta_{w-1}} p_{L|U}(l|u=+1)dl \\
\epsilon_w &= \frac{1}{P_w} \int_{-\zeta_w}^{-\zeta_{w-1}} p_{L|U}(l|u=+1)dl \\
I_w &= 1 - h_b(\epsilon_w)
\end{aligned}$$

where $P_w$ is the probability of sub-channel $w$, $\epsilon_w$ is its corresponding cross-over probability of the BSC sub-channel and $I_w$ is the corresponding mutual information of the sub-channel. $h_b(\epsilon)$ denotes the binary entropy function. Without loss of generality, the sub-channel $w = 0$ can be interpreted as a BSC with crossover probability 0.5 [46]. The mutual information of $I_{EV}$ can be obtained by taking the expectation of the mutual information of the subchannels:

$$I_{EV} = \sum_{d=2}^{d_v} \lambda_i \, \mathbb{E}_w\{I_w^d\} = \sum_{d=2}^{d_v} \lambda_i \sum_{w=0}^{W} P_w^{(d)} I_w^{(d)} \tag{5.16}$$

As it can be seen in (5.15) or (5.16), the EXIT function for the VNs depends on the initial messages and $I_{AV}$. To obtain the CN EXIT function, the approximate duality property is exploited [4, 92]. This states that a degree-$d$ single parity-check code and that of a degree-$d$ repetition code are related as

$$I_{E,SPC}(d, I_A) = 1 - I_{E,REP}(d, 1 - I_A)$$

the EXIT function for CN only depends on $I_{AC}$.

EXIT curves can be used to design LDPC codes [92]. We will consider only check-regular LDPC codes. In order to converge to a vanishingly probability of error for decoding, the EXIT chart of the VN has to lie above the inverse of the EXIT chart for the CN. The target in the code optimization is to maximize the rate (5.11) while considering a fixed CN degree distribution and a fixed $P_0$. Therefore, we

obtain the optimization method as

$$\begin{cases} \text{maximize} & \sum_{i=2}^{d_v} \lambda_i / i \\ \text{subject to} & I_{EC}^{-1}(I_{AC}) < I_{EV}(I_{AV}, P_0) \\ \text{and to} & \sum_{i=2}^{d_v} \lambda_i = 1, \lambda_i \geq 0, i = 2, 3, .., d_v \end{cases} \tag{5.17}$$

where it is solved by discretization of $I_{AV} \in (0, 1)$ and applying linear programing. Using this optimizer we can have a first rough estimate of the code parameters which satisfies (5.7) approximately.

### 5.4.3 Refined solution for a fake harder channel

In general, the EXIT chart optimization is optimistic in the sense that the optimized degree distribution might not have vanishing probability of error [69]. To make sure that the optimized degree distribution is valid, after the EXIT chart optimization we evaluate the decoding error probability of its output degree distribution with discretized DE [74]. If the optimized degree distribution coming out of the EXIT chart optimization do not have a zero decoding error probability, we optimize the degree distribution for a more difficult channel.

Keeping in mind that for the initial messages, the position of the LLRs in the space and their associated probabilities will be determined by the decoding metric $Q_{XY}$ and the source distribution $P_{XY}$, respectively, the term difficult channel means that with the same decoding metric we assume that the probability of receiving the wrong symbol is higher than the true one. This affects the probabilities of the initial messages and can be achieved by decreasing the probability of positive LLRs and increasing the negative LLRs probabilities in $P_0$ by a gap $\epsilon$:

$$P_{XY}(x = v_1, y = v_2) = \begin{cases} P_{XY}(x = v_1, y = v_2) - \epsilon \cdot P_Y(y = v_2) & \text{if } m_0^Q(v = (v_1, v_2)) > 0 \\ P_{XY}(x = v_1, y = v_2) + \epsilon \cdot P_Y(y = v_2) & \text{if } m_0^Q(v = (v_1, v_2)) < 0 \end{cases} \tag{5.18}$$

This way we are sure that the decoding metric is fixed and only its probability distribution is changing. For a binary-input binary-output source, this corresponds to increasing the crossover probabilities ($P_{X|Y}(x = 1|y = 0)$ and $P_{X|Y}(x = 0|y = 1)$) by $\epsilon$.

We increase the difficulty of the channel until the optimized degree can converge to zero decoding error by DE test. In the EXIT curve computations, we apply both VN EXIT curves of (5.15) and (5.16) and pick the one which provides the higher rate.

### 5.4.4 Final solution

With the EXIT chart optimization we are able to have a rough estimation of the LDPC degree distributions, but this estimation is not optimal. Indeed, the assumption of having Gaussian density for the messages used in EXIT chart, was to simplify and stabilize the numerical computation of the evolution of the message densities. In order to relax this Gaussian assumption, we tune this rough estimation by another optimization, proposed by Chung et al. [74], which uses discretized DE in its “inner" loop and takes as input the rough estimation of degree distribution and tries to optimize it iteratively. The details

of the algorithm can be found in [74]. For the case of SW coding, again the initial message density of (5.10) is used during the calculation of message densities.

To design LDPC codes using discretized DE, again the channel rate is maximized using linear programming which results in optimizing the $\lambda(x)$ for a fixed $\rho(x)$. For that, unlike EXIT chart optimization, the algorithm must be initiated with a $\lambda(x)$ that results the channel code rate lower than the desired rate. Then the optimization is run to update the degree distributions to $\lambda'(x)$ and increase the channel code rate maintaining the following constrains:

$$\begin{cases} \text{maximize} & \sum_{i=2}^{d_v} \lambda_i / i \\ \text{subject to} & \sum_{i=2}^{d_v} \lambda_i = 1, \lambda_i \geq 0, i = 2, 3, .., d_v \\ \text{and to} & \lambda'(x) \text{ is not significantly different from } \lambda(x) \\ \text{and to} & \lambda'(x) \text{ produces smaller probability of error} \end{cases} \tag{5.19}$$

We recursively tune the output of (5.19) until the output $\lambda(x)$ does not change much. The details of the algorithm is given in Algorithm 3. Here $EXIT\_opt(m_0^Q, P, d_c)$ is the exit chart optimization function discussed in Section 5.4.2 which takes as input the initial messages and their associated probabilities for a fixed check-regular LDPC codes with $\rho(x) = x^{d_c-1}$. The output of this function is the optimized variable degree distribution $\lambda$. The function $DE(m_0^Q, P, (\lambda, \rho))$ analyzes the performance of LDPC code ensemble for a pair of degree distribution $(\lambda, \rho)$ using DE algorithm and the output of this function is the probability of error $p_e$. The function $QDE\_opt(m_0^Q, P, \lambda_{init}, dc)$ optimize the degree distribution starting from $\lambda_{init}$ for fixed $m_0^Q, P, d_c$ using quantized/discretized DE discussed here. The overall block diagram of the algorithm is also shown in Fig. 5.2.

## 5.5 Rate estimation of rate-adaptive linear codes under mismatched decoding

Since the code optimization of an LDPC code is a heavy process, the mismatched decoding may not be handled that way. Instead, another technique exists which consists in adapting the rate of an already optimized code using a rate-adaptive technique. When the possible set of SI sources is available at the encoder, the encoder has the ability to run the decoding algorithm at the encoder and check if zero decoding error can be achieved. Using this ability and by relaxing the fixed-length requirement, we are able to use closed loop iterative doping technique for source coding [10], which consists of feeding the source symbol with least reliability to the BP decoder.

We propose a way to mimic this behavior in DE analysis which has infinite block length and all-zero codeword assumption, in which an area $\Delta P$ is reduced from the middle (around zero value) of the LLR probability distribution of variable nodes and added to the largest positive LLR value probability. Every $D$ iterations this area will be increased by $\Delta P$ with respect to the previous $D$ iterations. In other words, within an iteration interval $[i \cdot D, (i+1) \cdot D)$ the probability $i \cdot \Delta P$ is reduced from the LLR probabilities around zero and will be added to the probability of largest positive LLR. The excess rate is equal to $i \cdot \Delta P$ in which the decoding error probability (probability of negative LLR values) converges to zero.

**Algorithm 3** Algorithm for LDPC code design

**Input:**
$P_{XY}$ as the true joint distribution,
$Q_{XY} = Q_{X|Y}.P_Y$ as the decoding metric,
Degree $d_c$ for the check-regular nodes, i.e. $\rho(x) = x^{d_c-1}$
Maximum degree of the variable nodes $d_v^{max}$
The maximum value of permitted gap $\epsilon$ to $\epsilon_{\max}$ and increase step of the gap to $\Delta_\epsilon$
**Output:** $\lambda^{opt}$
*Initialization*: $\mathbb{A} = \emptyset$
Initialize $\epsilon \leftarrow 0$
**while** $\epsilon \leq \epsilon_{\max}$ **do**
  $m_0^Q \leftarrow$ using $Q_{XY}$ in (5.9)
  Assign probabilities to $m_0^Q$ using (5.18)
  $\lambda^{opt} \leftarrow EXIT\_opt(m_0^Q, P, d_c)$
  $p_e \leftarrow DE(m_0^Q, P, (\lambda^{opt}, \rho))$
  **if** $p_e \approx 0$ **then**
    break
  **end if**
  $\epsilon \leftarrow \epsilon + \Delta_\epsilon$
**end while**
**repeat**
  $\lambda^{opt} \leftarrow QDE\_opt(m_0^Q, P, \lambda^{opt}, dc)$
**until** $\lambda^{opt}$ or the rate resulting from $(\lambda^{opt}, \rho)$ converge

## 5.6 Experimental results

The goal of this section is to compute the excess rate practically using LDPC codes with BP decoding and compare it with KL divergence suggested in Section 5.3.4. We proposed two approaches to compute the excess rate practically in Sections 5.4 and 5.5. Thanks to the framework introduced in Section 5.4 and illustrated in Fig. 5.2, we are able to design LDPC codes under any decoding metric. To verify and compare the performance of the proposed code optimization framework, we picked the same source distribution of [87, section IV-A] and set the parameters of our second optimizer same as what is mentioned in there. In [87], after changing manually the initial $\lambda$ distribution of the variable nodes several times, the best SW rate reported is equal to 0.6 while in our method, without any need to do the manual initialization (initial $\lambda$ comes from the proposed EXIT chart rough optimizer), compression rate of 0.589 is achieved (the lower the rate, the better the compression) with the same maximum VN degree and check regular node degree. The degree distributions of the code are given below:

$$\lambda(x) = 0.238796x + 0.210703x^2 + 0.117978x^5 + 0.125822x^6 + 0.306701x^{19}$$
$$\rho(x) = x^6$$

For the rest of code design experiments, the maximum VN degree is set to 100 in all experiments. For all the designed codes, the SW code rate is $r_{sw} = \log|\mathcal{X}| - r_{ch}$, where $r_{ch}$ is the designed channel code rate. At high channel code rates (low SW code rates), with a fixed maximum VN degree, LDPC codes with higher check-regular degree perform better. Therefore, for designing code with respect to

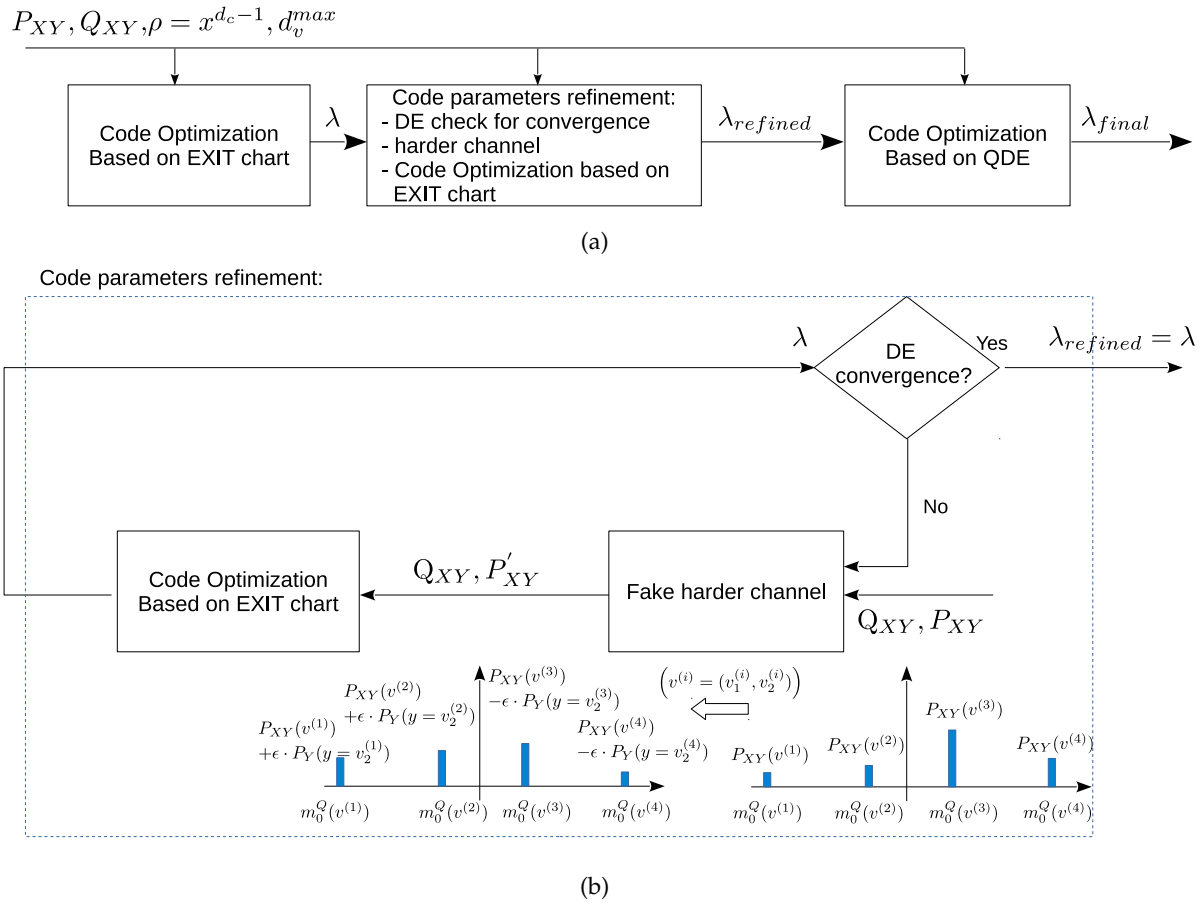



Figure 5.2 – The proposed code optimization block diagram. (a) Overall block diagram. (b) refinement diagram

the true distribution $P_{XY}$ we increase the degree of the check nodes until the optimized code produced by the code optimizer framework has less than 0.01 difference in rate with respect to SW limit. We denote this CN degree with $D_c^*$ and the corresponding SW rate with $r_P$. For each mismatched decoding metric, we optimize the code for all CN degrees which are less than or equal to $D_c^*$ and pick the one which has lower SW rate and denote it by $r_Q$. Finally we compare the excess rate $\Delta R = r_Q - r_P$ with supposition formulated in Section 5.3.4, i.e. $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$. The distribution is said to be over-estimated (under-estimated) if the conditional entropy w.r.t the wrong model $Q$, i.e. $H_Q(X|Y)$, is greater (less) than the conditional entropy with respect to the true model $P$, i.e. $H_P(X|Y)$.

### 5.6.1 Excess rate estimation for designed codes

Assuming that the input and output alphabets are binary, i.e. $|\mathcal{X}| = |\mathcal{Y}| = 2$, two cases are considered for the true joint distributions. First, we assume the simple case where the channel from $Y$ to $X$ is symmetric, i.e. the cross-over probability is $P_{X|Y}(0|1) = P_{X|Y}(1|0)$. We then release this assumption and assume the more general case $P_{X|Y}(0|1) \neq P_{X|Y}(1|0)$. The reason we choose channel $Y$ to $X$ instead

of $X$ to $Y$ is only to be sure that same $P_Y(y)$ is used for both source information $P_{XY}$ and decoding metric $Q_{XY}$. In all experiments a symmetric channel $Q_{X|Y}(0|1) = Q_{X|Y}(1|0) = q$ is assumed for the mismatched decoding metric $Q_{XY} = Q_{X|Y}P_Y$ where $q$ is changing from 0.009 to 0.26 with step 0.0087 to have 30 decoding metrics for each experiment.

The check nodes have degree $d_c = 7$, thus $\rho(x) = x^6$ and 11 bit quantization level is used for discretized DE. First we design the LDPC codes according to Algorithm 3 using the true distribution $P_{XY}$ and compute the initial messages $m_0^P(v)$ with their associated probabilities $P_{XY}$. We compute the rate of optimized degree $r_P$. Then for each mismatched experiment $Q_{XY}$, we compute the initial messages $m_0^Q(v)$ with the same associated probabilities $P_{XY}$ and again optimize the variable nodes degree distribution using Algorithm 3 and compute the output rate $r_Q$. Finally, we compare the excess rate $\Delta R = r_Q - r_P$ with $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$. The code will be made available online.

A cross over probability of 0.1 is used for the experiments regarding the symmetric case. Two possible situations are considered in this case, either $Y$ (and consequently $X$) is uniformly distributed, i.e. $P_Y(0) = P_Y(1) = 0.5$, or its distribution is not uniform and $P_Y(0) = 0.1$. Results of the excess rates are shown in Fig. 5.3 as a function of KL divergence. For the experiments regarding the non-symmetric channel from $Y$ to $X$, the cross-over probabilities of $P_{X|Y}(0|1) = 0.05, P_{X|Y}(1|0) = 0.1$ are assumed. In this case, as show in Fig. 5.4, three different possibilities for the distribution of $Y$ are considered: $P_Y(0) = 0.1, P_Y(0) = 0.9$ and $P_Y(0) = 0.5$.

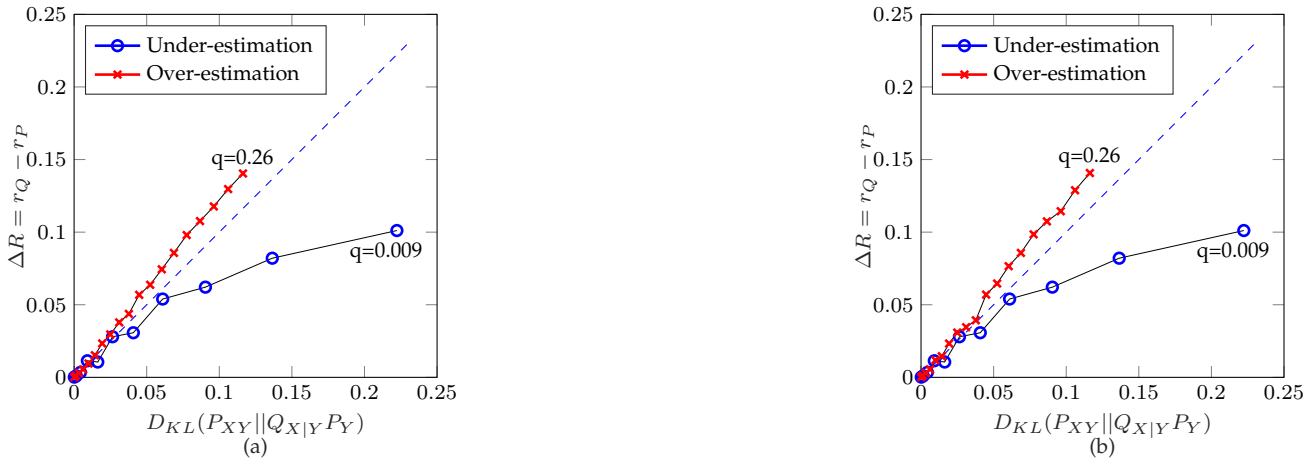



Figure 5.3 – Excess rate under mismatched decoding metric $Q_{XY}(x,y) = Q_{X|Y}(x|y)P_Y(y)$ as a function of $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$ for the symmetric channel from $Y$ to $X$. Each point shows the excess rate for a specific $Q_{X|Y}(0|1) = Q_{X|Y}(1|0) = q$. The dashed line is the identity line $D_{KL}(P_{XY_i}||Q_{X|Y_i}P_{Y_i})$. (a) $P_Y(0) = P_Y(1) = 0.5$. (b) $P_Y(0) = 0.1$.

Figs. 5.3 and 5.4 show that, for small KL divergences, this metric seems to be a good measure to estimate the excess rate. As the KL divergence increases, depending on over-estimating or under-estimating the source distribution, two behaviors are seen. For the same KL divergence, under-estimation of the source distribution produces less excess rate than its over-estimation. In other words, while for the over-estimated cases the excess rate is well correlated with KL divergence, for under-estimated cases this metric acts as an upper bound. This is consistent with the results obtained in [73, 72], where it

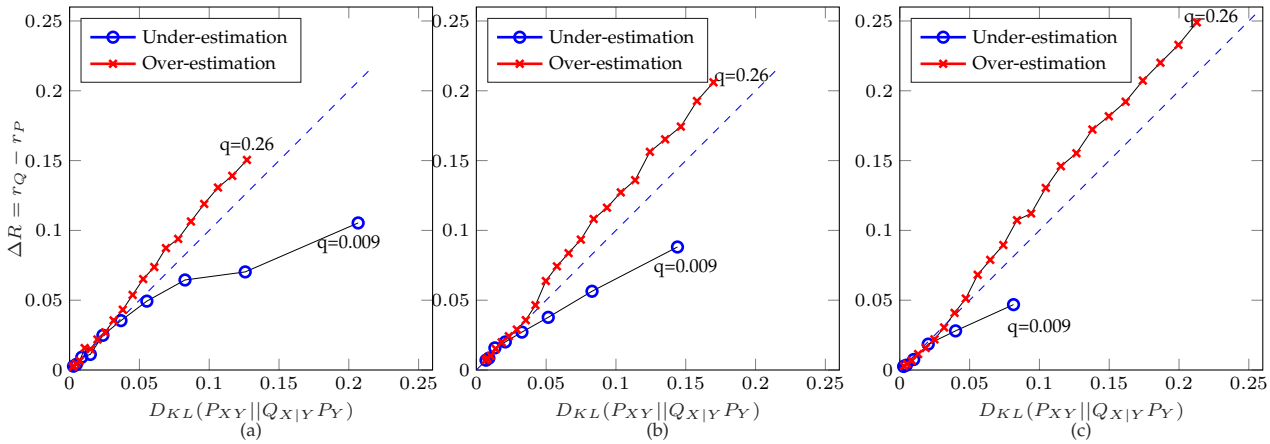



Figure 5.4 – Excess rate under mismatched decoding metric $Q_{XY}(x,y) = Q_{X|Y}(x|y)P_Y(y)$ as a function of $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$ for the non-symmetric channel from $Y$ to $X$. Each point shows the excess rate for a specific $Q_{X|Y}(0|1) = Q_{X|Y}(1|0) = q$. The dashed line is the identity line $D_{KL}(P_{XY_i}||Q_{X|Y_i}P_{Y_i})$. (a) $P_Y(0) = 0.1$. (b) $P_Y(0) = P_Y(1) = 0.5$. (c) $P_Y(0) = 0.9$.

is stated that for BP decoding (assuming Gaussian channel) over-estimation of the true model is more detrimental than its under-estimation (note that their definition of over/under-estimation is the opposite of our definition because they define over/under-estimation on the signal-to-noise ratio mismatch rather than rate).

Considering the fact that the performance of LDPC codes with BP decoding decreases as the channel rate decreases (SW code increases), as shown in Fig. 5.5, the percentage of error in estimating the excess rate with respect to the SW rate of the designed code, defined in (5.20), is less than 7.5% for all over-estimated experiments and around 40% of them have less than 1% of error.

$$\frac{|\Delta R_i - D_{KL}(P_{XY_i}||Q_{X|Y_i}P_{Y_i})|}{r_Q} \tag{5.20}$$

### 5.6.2 Excess rate estimation for rate-adaptive codes

We use the same setup of non-symmetric distribution shown in Fig. 5.4 for estimation of excess rate in rate-adaptive codes, i.e. $P_{X|Y}(0|1) = 0.05, P_{X|Y}(1|0) = 0.1$ and $P_Y(0)$ is equal to 0.1, 0.5 or 0.9. We use the codes which are optimized for these three cases as a base code for doping. Again symmetric distribution $Q_{X|Y}(0|1) = Q_{X|Y}(0|0) = q$ is assumed for the decoding metric. Since the doping technique uses the code which is optimized for the true distribution, the decoding metrics shouldn't be very far from the true model. Moreover, another limitation is that the doping technique is not suitable for the cases when the true model is under-estimated such that the estimated model is far from the true model. This is because in this case we are too optimistic about the model and since the doping technique tries to feed the source symbols which are less reliable, it is not able to change the wrong bits to the true ones. Therefore, we pick some of the decoding metrics from the previous setup mentioned in Section 5.6.1 which are over-estimated, but not too far from the true model (KL divergence less than 0.1) or under-

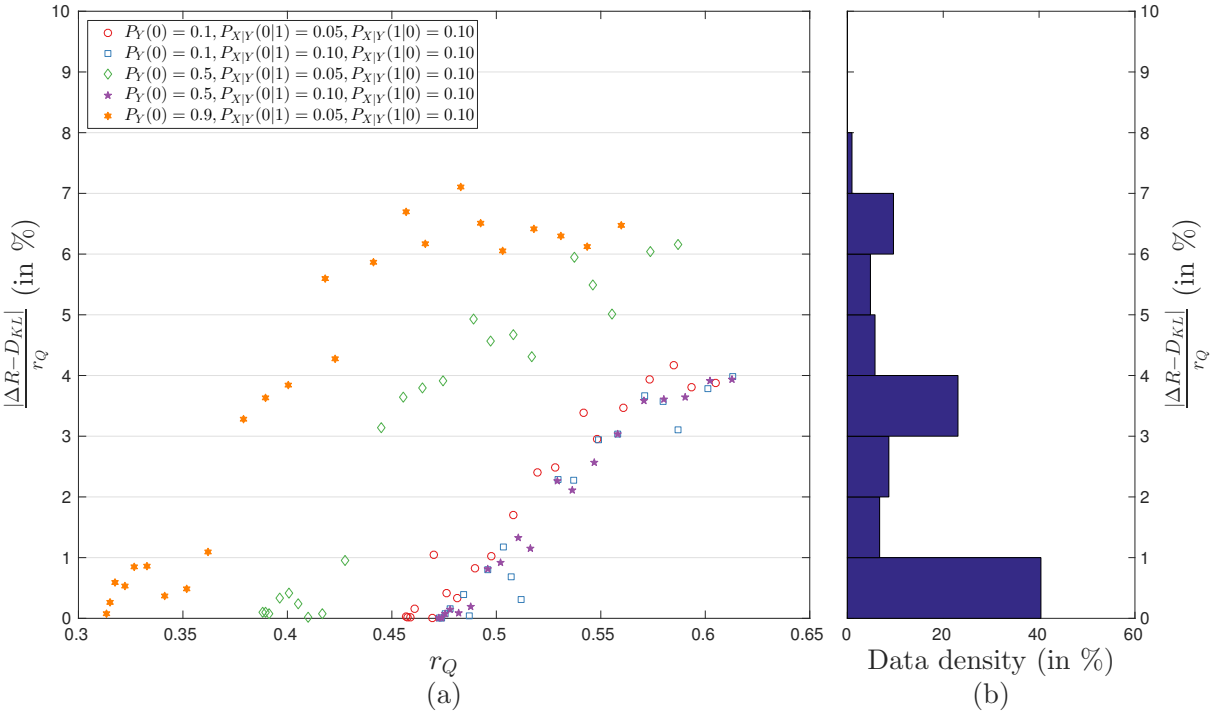



Figure 5.5 – (a) Percentage of error between KL divergence and estimated excess rate with respect to the SW rate of the designed code ($r_Q$), when the source distribution is over-estimated. (b) Density of the data for each percent of error.

estimated, but very close to the true model (KL divergence less than 0.005). Results are shown in Fig. 5.6. Results show that again KL divergence is a good measure to estimate the excess rate, even for the doped codes, provided that the wrong model remains close to the true model.

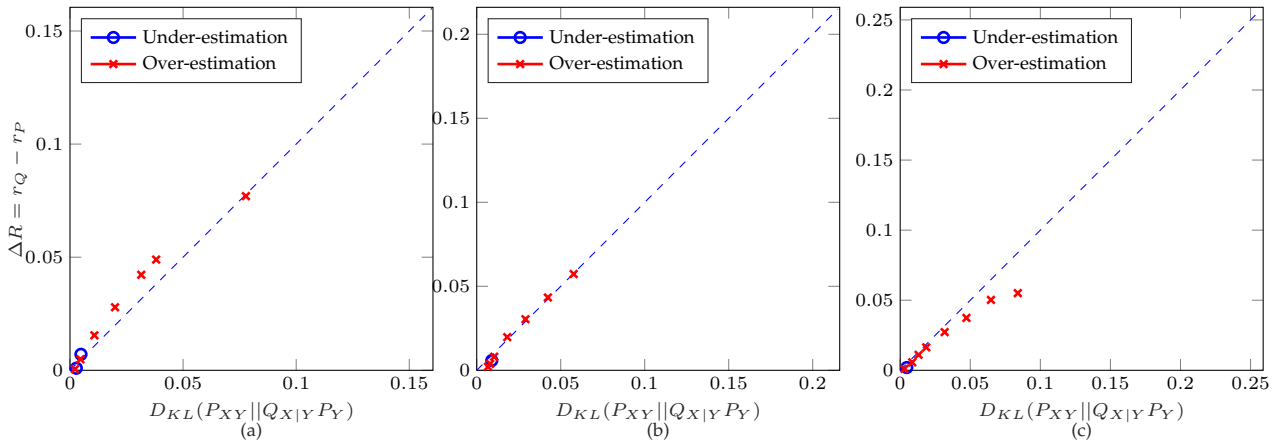



Figure 5.6 – Excess rate estimation using the doping technique under decoding metric $Q_{XY}(x, y) = Q_{X|Y}(x|y)P_Y(y)$ as a function of $D_{KL}(P_{XY}||Q_{X|Y}P_Y)$ for the non-symmetric channel from $Y$ to $X$. Each point shows the excess rate for a specific $Q_{X|Y}(0|1) = Q_{X|Y}(1|0) = q$. The dashed line is the identity line $D_{KL}(P_{XY_i}||Q_{X|Y_i}P_{Y_i})$. (a) $P_Y(0) = 0.1$. (b) $P_Y(0) = P_Y(1) = 0.5$. (c) $P_Y(0) = 0.9$.

## 5.7 Summary

In interactive coding, selecting a model requires to evaluate the impact of a wrong model on the compression rate. We characterized this excess rate in terms of the mismatched capacity of the dual channel, for model based source coding, which is an open problem. For these coding schemes, we showed experimentally that the KL divergence between the true and the wrong model is a proper estimate of the excess rate experienced by mismatched BP decoding of LDPC codes. This was evidenced by a new algorithm that allows to design LDPC codes decoded with BP according to a wrong metric. More precisely, it was shown that the KL matches well the case where the wrong model is pessimistic (overestimation of the rate) while it is an upper bound for the optimistic case (underestimation case). Moreover, it has been shown that the KL divergence is also an accurate estimate to evaluate the excess rate for a class of variable rate LDPC codes (doped codes), provided that the wrong model remains close to the true model.

# CONCLUSIONS AND FUTURE WORK

## Conclusions

With the tremendous advances made in interactive media technologies, these data have become increasingly popular. This type of media has unique features that distinguish it from other media. First, the data for this type of media is very large, with a great deal of redundancy between the data items. Second, users request only part of the database each time. This makes data compression of such a database very challenging. On the one hand, to maximize data compression, all dependencies must be exploited together. On the other hand, upon user's request, only the required data must be sent to have optimal transmission.

In this dissertation, the goal was to provide a generic interactive coding scheme. For that, we considered a representation that can encompass different interactive media: which is a 2D projection. We presented new tools related to the compression of 2D images which provide interactivity. If we consider the process pipeline of interactive communication depicted in Fig. 5.7, we had contributions in each step of this procedure. In this regard, in the first stage, we proposed an **evaluation** framework in Chapter 2 by which we can compare different methods that provide interactive compression for 2D visual data. In the proposed framework, along with the distortion sensed at the user's side, we differentiated the transmission rate from storage and considered all these three criteria together in the evaluation. We proposed three metrics for the evaluation: First, the pair of (BD-R, BD-S) was proposed to compute the average bit rate saving/loss for the same range of distortion between different schemes. Second, iso-values were proposed for the case when the evaluation must be considered under fixed (and not average) metric. Finally, weighted-BD was introduced to combine storage rate and transmission rate based on their weighted importance for the same range of distortion between different schemes. We illustrated the efficiency of the framework with the classical compression schemes that were proposed for 360° images.

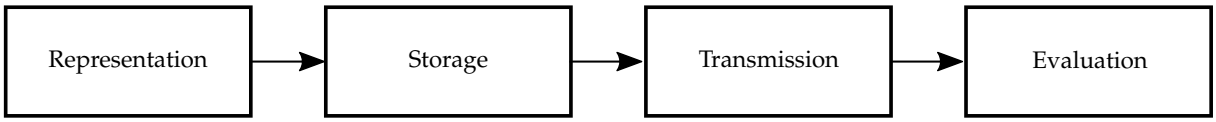



Figure 5.7 – Processing pipeline for interactive communication.

In Chapter 3, we proposed the generic interactive compression scheme for 2D images to minimize the **transmission** cost in the communication system with a reasonable **storage** cost. In this scheme, blocks are encoded considering that a set of predictions is available for each block knowing that one of these predictions will be available at the decoder. These different predictions come from different neighbors of the block that enable different user navigations. This is in contrast to classical compression methods in which there is only one prediction per block and this prediction is known at both encoder and de-

coder. The neighborhood is defined by a navigation graph determined by the application. The scheme encodes the block considering all possible predictions using incremental codes that can be implemented by rate-adaptive channel codes. A single bitstream is built per each block that compensates for the worst prediction. For better predictions in the set, a substream can be extracted from the bitstream. Moreover, some blocks are introduced as access blocks for which an empty prediction is added to the prediction set to begin decoding operation. When a user observes part of the image, it corresponds to the request of a set of blocks. In this scheme, the system transfers only the requested blocks and then blocks are decoded one after the other with a specific scanning order.

In Chapter 4, we focused on compression of two imaging modalities that are inherently interactive media: 360° images and textures of 3D models. With the help of the generic coder proposed in Chapter 3, we designed an efficient interactive codec that can transmit only what a user requests. This can be achieved with a reasonable storage overhead at the server. In this regard, we provided a set of predictions for each block, while the geometry of each imaging modality is taken into account in the production of the prediction set. We developed an optimal scanning method that processes blocks one after the other such that the overall correlation between blocks is better exploited. Experimental results showed that the proposed scheme is perfectly suited for interactive transmission because it can balance the trade-off between the transmission rate and the storage. Another figure of merit of the proposed scheme is that the transmission rate increases gradually over time.

We also proposed better **representations** for each of these modalities in the same chapter. For 360° images, we proposed an efficient spherical coder in which all coding procedures of the image are performed directly on the sphere. More precisely, we defined spherical blocks on the sphere. This allowed us to propose novel predictions, transformation, and scanning of these blocks which are all defined on the sphere. For the transformation, we proposed a GFT approach to transform the residual signal that adapts better to the geometry of the sphere. This representation can be integrated in the interactive compression scheme proposed for 360° images. We also proposed a geometry-aware representation for texture map of 3D models that uses 3D geometry to find the neighbors within the texture. All scanning, prediction, and transformation of the blocks are adapted to the topology of the 3D model. In this scheme, only informative blocks participate in compression. A GFT approach was proposed to take into account informative pixels for the transformation. This geometry aware representation was also exploited in the interactive compression scheme proposed for texture map of 3D models. Experimental results of both representations demonstrated the effectiveness of the proposed representations.

The basis of the compression for incremental codes is based on the correlation model between the data. It is generally costly to store and transmit these models. A solution to this problem is to approximate these models with simpler parametric models. However, this approximation introduces an excess rate on the compression. To have a better description of the model, which affects both **storage** and **transmission** rate, we studied the effect of choosing the model on the compression rate in Chapter 5. We characterized this excess rate in terms of the mismatched capacity of the dual channel. For interactive coding schemes, we showed that KL divergence between the true and the wrong model is a proper estimate of the excess rate experienced by mismatched BP decoding of LDPC codes.

## Future work

Several approaches can be considered to extend the work presented in this thesis.

The first possible future work is to expand the work into the temporal domain. Exploiting temporal redundancy between neighboring frames is a powerful tool in video compression that provides higher compression rates compared to the intra compression of images as a result of a better prediction produced with inter-prediction. This is mainly because inter-prediction uses the block matching technique to find the most similar reference region (in another frame) to the block to be encoded, while the intra prediction uses a few pixels on the sides of the block as references to produce the prediction. The generic coder proposed in Chapter 3 can be extended to support also the temporal domain. For each block, the corresponding inter-prediction must be added to the set of predictions. However, simple block matching can not be used directly for spherical images represented by 2D projections because this mapping introduces distortion which modifies the area of the projected objects and a constant linear displacement on the motion plane turns into a non-constant and non-linear displacement on the spherical imaging surface [100, 25]. Therefore, proper motion compensation algorithms must be proposed to take into account the geometry of the data in an interactive scenario. Moreover, a scanning strategy with proper signaling must be presented to benefit from both types of predictions (intra and inter predictions) while minimizing the transmission rate as much as possible.

Another powerful property that can be added to the generic coder to improve its coding efficiency is the splitting of the blocks into sub-blocks. This property has already been used in classical video compression scheme to improve the prediction signal. However, care must be taken for interactive compression, as a naive splitting of blocks results in short length codes, and the performance of channel codes drops significantly with such codes. Therefore, the proposed method should preferably be such that it does not reduce the size of the sequence we want to encode while allowing the blocks to be subdivided to produce better predictions.

In Chapter 5, we discussed the model selection problem in interactive communication. To derive the results which are close to theory, we used a key feature of information theory, i.e. the assumption of infinite block-length codes. However, in practice, the length of the code is not infinite, and one needs to investigate the excess rate in case of finite block-length.

Moreover, investigating strategies which avoid transmitting the model is of great interest. For that, classical compression schemes estimate the source model by sequential encoders like arithmetic coding encoder, and then *implicitly* embed the model into the compressed bitstream using techniques such as context-adaptive coding [52]. This way, the decoder can reconstruct the model before decoding the bitstream. Similar techniques have been proposed for Slepian-Wolf model-based coding using the connection they have with channel codings. For example, [33, 95] propose methods to estimate the constant crossover probability of stationary-correlated binary sources, and [23, 32] propose solutions to compute the varying cross-over probability of nonstationary-correlated binary sources. Since the proposed interactive compression is based on model-based coding, research on similar methods for incremental codes is valuable.

# AUTHOR'S PUBLICATION

## International peer-reviewed conference papers

N. Mahmoudian Bidgoli, T. Maugey and A. Roumy, "Correlation model selection for interactive video communication," *2017 IEEE International Conference on Image Processing (ICIP),* Beijing, 2017, pp. 2184-2188.

F. Nasiri, N. Mahmoudian Bidgoli, F. Payan and T. Maugey, "A Geometry-aware Framework for Compressing 3D Mesh Textures," *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* Brighton, United Kingdom, 2019, pp. 4015-4019.

N. Mahmoudian Bidgoli, T. Maugey and A. Roumy, "Evaluation Framework for 360-Degree Visual Content Compression with User View-Dependent Transmission," *2019 IEEE International Conference on Image Processing (ICIP),* Taipei, Taiwan, 2019, pp. 3756-3760.

N. Mahmoudian Bidgoli, T. Maugey, A. Roumy, F. Nasiri and F. Payan. "A geometry-aware compression of 3D mesh texture with random access," *2019 Picture Coding Symposium (PCS),* Ningbo, China, 2019.

N. Mahmoudian Bidgoli, T. Maugey and A. Roumy. "Intra-coding of 360-degree images on the sphere," *2019 Picture Coding Symposium (PCS),* Ningbo, China, 2019.

## Journal Papers in preparation

N. Mahmoudian Bidgoli, T. Maugey and A. Roumy, "Fine granularity access in interactive compression of 360-degree images based on rate adaptive channel codes," submitted to *IEEE Transactions on Multimedia*.

N. Mahmoudian Bidgoli, T. Maugey and A. Roumy, "Model selection for interactive coding," (to be submitted).

## National peer-reviewed conference papers

F. Nasiri, N. Mahmoudian Bidgoli, F. Payan and T. Maugey, "Codage d'atlas de textures avec prédiction guidée par la topologie des maillages," *Colloque Gretsi (Symposium of the French Research group on signal and image processing),* 2019.
N. Mahmoudian bidgoli, T. Maugey and A. Roumy. "Compression de contenus 360° et transmission adaptée à la navigation de l'utilisateur," *Colloque Gretsi (Symposium of the French Research group on signal and image processing),* 2019.

# LIST OF FIGURES

# List of Tables

# LIST OF ALGORITHMS

# BIBLIOGRAPHY


[1] R. G. d. A. Azevedo, N. Birkbeck, F. De Simone, I. Janatra, B. Adsumilli, and P. Frossard, « Visual Distortions in 360-degree Videos », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* (2019), pp. 1–1.

[2] A. AbuBaker, R. Qahwaji, S. Ipson, and M. Saleh, « One Scan Connected Component Labeling Technique », *in*: *2007 IEEE International Conference on Signal Processing and Communications*, Nov. 2007, pp. 1283–1286.

[3] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, « Google Street View: Capturing the World at Street Level », *in*: *Computer* 43.*6* (June 2010), pp. 32–38.

[4] A. Ashikhmin, G. Kramer, and S. ten Brink, « Extrinsic information transfer functions: model and erasure channel properties », *in*: *IEEE Transactions on Information Theory* 50.*11* (Nov. 2004), pp. 2657–2673.

[5] G. V. d. Auwera, M. Coban, F. Hendry, and M. Karczewicz, *AHG8: Truncated Square Pyramid Projection (TSP) For 360 Video*, 4th meeting report JVET-D0071, Chengdu: Joint Video Exploration Team (JVET) of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11, Oct. 2016.

[6] L. Balmelli, G. Taubin, and F. Bernardini, « Space-Optimized Texture Maps », *in*: *Computer Graphics Forum* 21.*3* (2002), pp. 411–420.

[7] A. C. Beers, M. Agrawala, and N. Chaddha, « Rendering from Compressed Textures », *in*: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, New York, NY, USA: ACM, 1996, pp. 373–378.

[8] G. Bjontegaard, « Calculation of average PSNR differences between RD-curves », *in*: *VCEG-M33* (2001).

[9] Chip Brown, *Bringing pixels front and center in VR video*, Mar. 2017, URL: `https://blog.google/products/google-ar-vr/bringing-pixels-front-and-center-vr-video`.

[10] G. Caire, S. S. Shitz, and S. Verdu, « Universal data compression with LDPC codes », *in*: *TURBOCODING 2003, 3rd International Symposium on Turbo Codes and Related Topics, 1-5 September 2003, Brest, France*, Brest, FRANCE, Sept. 2003.

[11] J. Chen, D. He, and A. Jagmohan, « On the Duality Between Slepian–Wolf Coding and Channel Coding Under Mismatched Decoding », *in*: *IEEE Transactions on Information Theory* 55.*9* (Sept. 2009), pp. 4006–4018.

[12] J. Chen, D. He, and E. Yang, « On the Codebook-Level Duality Between Slepian-Woif Coding and Channel Coding », *in*: *2007 Information Theory and Applications Workshop*, Jan. 2007, pp. 84–93.

[13] Z. Chen, Y. Li, and Y. Zhang, « Recent advances in omnidirectional video coding for virtual reality: Projection and evaluation », *in*: *Signal Processing* 146 (2018), pp. 66–78, ISSN: 0165-1684.

[14] G. Cheung, A. Ortega, and N. Cheung, « Interactive Streaming of Stored Multiview Video Using Redundant Frame Structures », *in*: *IEEE Transactions on Image Processing* 20.3 (Mar. 2011), pp. 744–761.

[15] N. Cheung, A. Ortega, and G. Cheung, « Distributed source coding techniques for interactive multiview video streaming », *in*: *2009 Picture Coding Symposium*, May 2009, pp. 1–4.

[16] NM. Cheung and A. Ortega, « Compression algorithms for flexible video decoding », *in*: *Visual Communications and Image Processing (VCIP'08)*, vol. SPIE 6822, Jan. 2008.

[17] *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*, White Paper Document ID: 1551296909190103, Cisco, Feb. 2019, URL: `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html`.

[18] X. Corbillon, « Enable the next generation of interactive video streaming », PhD thesis, Ecole nationale supérieure Mines-Télécom Atlantique, 2018.

[19] X. Corbillon, F. De Simone, and G. Simon, « 360-Degree Video Head Movement Dataset », *in*: *Proceedings of the 8th ACM on Multimedia Systems Conference*, MMSys'17, Taipei, Taiwan: ACM, 2017, pp. 199–204.

[20] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, « Viewport-adaptive navigable 360-degree video delivery », *in*: *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–7.

[21] T.M. Cover and J.A. Thomas, *Elements of information theory, second Edition*, Wiley, 2006.

[22] I. Csiszár and P.C. Shields, « Information Theory and Statistics: A Tutorial », *in*: *Foundations and Trends® in Communications and Information Theory* 1.4 (2004), pp. 417–528, ISSN: 1567-2190.

[23] L. Cui, S. Wang, and S. Cheng, « Adaptive Slepian-Wolf Decoding Based on Expectation Propagation », *in*: *IEEE Communications Letters* 16.2 (Feb. 2012), pp. 252–255.

[24] W. Dai, G. Cheung, N. Cheung, A. Ortega, and O. C. Au, « Rate-distortion optimized merge frame using piecewise constant functions », *in*: *2013 IEEE International Conference on Image Processing*, Sept. 2013, pp. 1787–1791.

[25] F. De Simone, P. Frossard, N. Birkbeck, and B. Adsumilli, « Deformable block-based motion estimation in omnidirectional image sequences », *in*: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2017, pp. 1–6.

[26] R. Deaux, *Introduction to the Geometry of Complex Numbers*, Dover Books on Mathematics, Dover Publications, 2013, ISBN: 9780486158044.

[27] S. C. Draper and E. Martinian, « Compound conditional source coding, Slepian-Wolf list decoding, and applications to media coding », *in*: *2007 IEEE International Symposium on Information Theory*, June 2007, pp. 1511–1515.

[28] E. Dupraz, T. Maugey, A. Roumy, and M. Kieffer, « Rate-Distortion Performance of Sequential Massive Random Access to Gaussian Sources with Memory », *in*: *2018 Data Compression Conference*, Mar. 2018, pp. 406–406.

[29] E. Dupraz, T. Maugey, A. Roumy, and M. Kieffer, « Transmission and Storage Rates for Sequential Massive Random Access », working paper or preprint, May 2018, URL: `https://hal.inria.fr/hal-01799016`.

[30] E. Dupraz, A. Roumy, T. Maugey, and M. Kieffer, « Rate-storage regions for Extractable Source Coding with side information », *in*: *Physical Communication* (2019), p. 100845, ISSN: 1874-4907.

[31] A. El Gamal and A. Orlitsky, « Interactive Data Compression », *in*: *25th Annual Symposium on Foundations of Computer Science, 1984.* Oct. 1984, pp. 100–108.

[32] Y. Fang, « LDPC-Based Lossless Compression of Nonstationary Binary Sources Using Sliding-Window Belief Propagation », *in*: *IEEE Transactions on Communications* 60.*11* (Nov. 2012), pp. 3161–3166.

[33] Y. Fang and J. Jeong, « Correlation parameter estimation for LDPC-based slepian-wolf coding », *in*: *IEEE Communications Letters* 13.*1* (Jan. 2009), pp. 37–39.

[34] M. S. Floater and K. Hormann, « Surface Parameterization: a Tutorial and Survey », *in*: *Advances in Multiresolution for Geometric Modelling*, ed. by N. A. Dodgson, M. S. Floater, and M. A. Sabin, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 157–186.

[35] C. Fu, L. Wan, T. Wong, and C. Leung, « The Rhombic Dodecahedron Map: An Efficient Scheme for Encoding Panoramic Video », *in*: *IEEE Transactions on Multimedia* 11.*4* (June 2009), pp. 634–644.

[36] A. Ganti, A. Lapidoth, and I. E. Telatar, « Mismatched decoding revisited: general alphabets, channels with memory, and the wide-band limit », *in*: *IEEE Transactions on Information Theory* 46.*7* (Nov. 2000), pp. 2315–2328.

[37] K. Garrett, *9 Virtual Tour Statistics You Need to Know in 2019*, Feb. 2019, URL: `https://www.lcp360.com/blog/virtual-tours-statistics`.

[38] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, « HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere », *in*: *The Astrophysical Journal* 622.*2* (Apr. 2005), pp. 759–771.

[39] M. Hosseini and V. Swaminathan, « Adaptive 360 VR Video Streaming: Divide and Conquer », *in*: *2016 IEEE International Symposium on Multimedia (ISM)*, Dec. 2016, pp. 107–110.

[40] *Incense Burner from Iran 3D Model*, 2016, URL: `https://skfb.ly/6MunN`.

[41] Jacek Jankowski and Martin Hachet, « A Survey of Interaction Techniques for Interactive 3D Environments », *in*: *Eurographics 2013 - State of the Art Reports*, ed. by M. Sbert and L. Szirmay-Kalos, The Eurographics Association, 2013.

[42] M. Karczewicz and R. Kurceren, « The SP- and SI-frames design for H.264/AVC », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 13.*7* (July 2003), pp. 637–644.

[43] P. Krajcevski, S. Pratapa, and D. Manocha, « GST: GPU-decodable Supercompressed Textures », *in*: *ACM Trans. Graph.* 35.6 (Nov. 2016), 230:1–230:10.

[44] E. Kuzyakov and D. Pio, *Next-generation video encoding techniques for 360 video and VR*, 2016, URL: `https://code.fb.com/virtual-reality/next-generation-video-encoding-techniques-for-360-video-and-vr/`.

[45] J. Lainema, F. Bossen, W. Han, J. Min, and K. Ugur, « Intra Coding of the HEVC Standard », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pp. 1792–1801.

[46] I. Land and J. Huber, « Information Combining », *in*: *Found. Trends Commun. Inf. Theory* 3.3 (Nov. 2006), pp. 227–330.

[47] A. Laurent, *Fethiye tombs, Turquie 3D Model*, 2016, URL: `https://sketchfab.com/search?q=fethiye-tombs-turquie/`.

[48] K. Lesniak, C. S. Tucker, S. Bilen, J. Terpenny, and C. Anumba, « Networked, Real Time Translation of 3D Mesh Data to Immersive Virtual Reality Environments », *in*: *ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 1B, 36th Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2016, V01BT02A053.

[49] A. D. Liveris, Zixiang Xiong, and C. N. Georghiades, « Compression of binary sources with side information at the decoder using LDPC codes », *in*: *IEEE Communications Letters* 6.10 (Oct. 2002), pp. 440–442.

[50] A. Maglo, I. Grimstead, and C. Hudelot, « POMAR: Compression of progressive oriented meshes accessible randomly », *in*: *Computers & Graphics* 37.6 (2013), Shape Modeling International (SMI) Conference 2013, pp. 743–752.

[51] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, « 3D Mesh Compression: Survey, Comparisons, and Emerging Trends », *in*: *ACM Comput. Surv.* 47.3 (Feb. 2015), 44:1–44:41.

[52] D. Marpe, H. Schwarz, and T. Wiegand, « Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 13.7 (July 2003), pp. 620–636.

[53] T. Maugey, I. Daribo, G. Cheung, and P. Frossard, « Navigation Domain Representation For Interactive Multiview Imaging », *in*: *IEEE Transactions on Image Processing* 22.9 (Sept. 2013), pp. 3459–3472.

[54] C. M. Mendes, K. Apaza-Agüero, L. Silva, and O. R. P. Bellon, « Data-driven progressive compression of colored 3D mesh », *in*: *2015 IEEE International Conference on Image Processing (ICIP)*, Sept. 2015, pp. 2490–2494.

[55] N. Merhav, G. Kaplan, A. Lapidoth, and S. Shamai Shitz, « On information rates for mismatched decoders », *in*: *IEEE Transactions on Information Theory* 40.6 (Nov. 1994), pp. 1953–1967.

[56] B. Micusik, « Two-view geometry of omnidirectional cameras », PhD thesis, Czech Technical University in Prague, 2004.

[57] B. Motz, G. Cheung, and A. Ortega, « Redundant frame structure using M-frame for interactive light field streaming », *in*: *2016 IEEE International Conference on Image Processing (ICIP)*, Sept. 2016, pp. 1369–1373.

[58] King-To Ng, Shing-Chow Chan, and Heung-Yeung Shum, « Data compression and transmission aspects of panoramic videos », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 15.*1* (Jan. 2005), pp. 82–95.

[59] *Open heritage: Google art and culture*, URL: `https://artsandculture.google.com/project/openheritage`.

[60] N. Perraudin, M. Defferrard, T. Kacprzak, and R. Sgier, « DeepSphere: Efficient spherical convolutional neural network with HEALPix sampling for cosmological applications », *in*: *Astronomy and Computing* 27 (2019), pp. 130–146.

[61] Andrew Pressley, « Gauss's Theorema Egregium », *in*: *Elementary Differential Geometry*, London: Springer London, 2001, pp. 229–246.

[62] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, « Optimizing 360 Video Delivery over Cellular Networks », *in*: *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, ATC '16, New York City, New York: ACM, 2016, pp. 1–6.

[63] Report ITU-R BT.2420-0, *Collection of usage scenarios and current statuses of advanced immersive audio-visual systems*, tech. rep., International Telecommunication Union (ITU), 2018.

[64] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, « Design of capacity-approaching irregular low-density parity-check codes », *in*: *IEEE Transactions on Information Theory* 47.*2* (Feb. 2001), pp. 619–637.

[65] T. J. Richardson and R. L. Urbanke, « The capacity of low-density parity-check codes under message-passing decoding », *in*: *IEEE Transactions on Information Theory* 47.*2* (Feb. 2001), pp. 599–618.

[66] J. Rissanen, « A Universal Prior for Integers and Estimation by Minimum Description Length », *in*: *The Annals of Statistics* 11.*2* (June 1983), pp. 416–431.

[67] M. Rizkallah, F. De Simone, T. Maugey, C. Guillemot, and P. Frossard, « Rate Distortion Optimized Graph Partitioning for Omnidirectional Image Coding », *in*: *2018 26th European Signal Processing Conference (EUSIPCO)*, Sept. 2018, pp. 897–901.

[68] S. Rossi and L. Toni, « Navigation-aware adaptive streaming strategies for omnidirectional video », *in*: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2017, pp. 1–6.

[69] A. Roumy, S. Guemghar, G. Caire, and S. Verdu, « Design methods for irregular repeat-accumulate codes », *in*: *IEEE Transactions on Information Theory* 50.*8* (Aug. 2004), pp. 1711–1727.

[70] A. Roumy and T. Maugey, « Universal lossless coding with random user access: The cost of interactivity », *in*: *2015 IEEE International Conference on Image Processing (ICIP)*, Sept. 2015, pp. 1870–1874.

[71] W. Ryan and S. Lin, *Channel Codes: Classical and Modern*, Cambridge University Press, 2009, ISBN: 9780521848688.

[72] H. Saeedi and A. H. Banihashemi, « Design of irregular LDPC codes for BIAWGN channels with SNR mismatch », *in*: *IEEE Transactions on Communications* 57.1 (Jan. 2009), pp. 6–11.

[73] H. Saeedi and A. H. Banihashemi, « Performance of Belief Propagation for Decoding LDPC Codes in the Presence of Channel Estimation Error », *in*: *IEEE Transactions on Communications* 55.1 (Jan. 2007), pp. 83–89.

[74] Sae-Young Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, « On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit », *in*: *IEEE Communications Letters* 5.2 (Feb. 2001), pp. 58–60.

[75] K. Sayood, *Introduction to Data Compression, Third Edition (Morgan Kaufmann Series in Multimedia Information and Systems)*, 5th, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2018, ISBN: 012620862X.

[76] J. Scarlett, A. Martinez, and A. G. i. Fabregas, « Mismatched Decoding: Error Exponents, Second-Order Rates and Saddlepoint Approximations », *in*: *IEEE Transactions on Information Theory* 60.5 (May 2014), pp. 2647–2666.

[77] A. Sgarro, « Source coding with side information at several decoders », *in*: *IEEE Transactions on Information Theory* 23.2 (Mar. 1977), pp. 179–182.

[78] C. E. Shannon, « A mathematical theory of communication », *in*: *The Bell System Technical Journal* 27.3 (July 1948), pp. 379–423.

[79] L. G. Shapiro and G. C. Stockman, *Computer Vision*, 1st, Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001, ISBN: 0130307963.

[80] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, « The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains », *in*: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98.

[81] Gwendal Simon, « Texture Images are not Regular Images:How to Leverage Geometry Information? », *in*: *IEEE MMTC Review-Letter* 10.2 (Apr. 2019), p. 3.

[82] D. Slepian and J. Wolf, « Noiseless coding of correlated information sources », *in*: *IEEE Transactions on Information Theory* 19.4 (July 1973), pp. 471–480.

[83] Smithsonian Institution, *Ewer with Birds, Snakes, and Humans*, URL: `https://3d.si.edu/explorer/ewer-with-birds-snakes-and-humans#downloads/`.

[84] Smithsonian Institution, *The 3D Program*, URL: `https://3d.si.edu`.

[85] J.P. Snyder, *Flattening the Earth: Two Thousand Years of Map Projections*, University of Chicago Press, 1993, ISBN: 9780226767475.

[86] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, « Overview of the High Efficiency Video Coding (HEVC) Standard », *in*: *IEEE Transactions on Circuits and Systems for Video Technology* 22.12 (Dec. 2012), pp. 1649–1668.

[87] Z. Sun, C. Tian, J. Chen, and K. M. Wong, « LDPC code design for asynchronous Slepian-Wolf coding », *in*: *IEEE Transactions on Communications* 58.*2* (Feb. 2010), pp. 511–520.

[88] R. Szeliski, « Image Alignment and Stitching: A Tutorial », *in*: *Found. Trends. Comput. Graph. Vis.* 2.*1* (Jan. 2006), pp. 1–104.

[89] A. TaghaviNasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, « Adaptive 360-degree video streaming using layered video coding », *in*: *2017 IEEE Virtual Reality (VR)*, Mar. 2017, pp. 347–348, DOI: 10.1109/VR.2017.7892319.

[90] M. Tanimoto, « FTV: Free-viewpoint Television », *in*: *Signal Processing: Image Communication* 27.*6* (2012), pp. 555–570, ISSN: 0923-5965.

[91] S. ten Brink, « Convergence behavior of iteratively decoded parallel concatenated codes », *in*: *IEEE Transactions on Communications* 49.*10* (Oct. 2001), pp. 1727–1737.

[92] S. ten Brink, G. Kramer, and A. Ashikhmin, « Design of low-density parity-check codes for modulation and detection », *in*: *IEEE Transactions on Communications* 52.*4* (Apr. 2004), pp. 670–678.

[93] *The power of media-rich map listings*, July 2015, URL: https://www.google.com/streetview/business/.

[94] I. Tosic and P. Frossard, « Low bit-rate compression of omnidirectional images », *in*: *2009 Picture Coding Symposium*, May 2009, pp. 1–4.

[95] V. Toto-Zarasoa, A. Roumy, and C. Guillemot, « Maximum Likelihood BSC Parameter Estimation for the Slepian-Wolf Problem », *in*: *IEEE Communications Letters* 15.*2* (Feb. 2011), pp. 232–234.

[96] E. Upenik, M. Rerabek, and T. Ebrahimi, « On the performance of objective metrics for omnidirectional visual content », *in*: *2017 Ninth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2017, pp. 1–6.

[97] D. Varodayan, A. Aaron, and B. Girod, « Rate-adaptive codes for distributed source coding », *in*: *Signal Processing* 86.*11* (2006), Special Section: Distributed Source Coding, pp. 3123–3130.

[98] L. Velho and J. Sossai Jr., « Projective texture atlas construction for 3D photography », *in*: *The Visual Computer* 23.*9* (Sept. 2007), pp. 621–629.

[99] B. Vishwanath, T. Nanjundaswamy, and K. Rose, « Motion Compensated Prediction for Translational Camera Motion in Spherical Video Coding », *in*: *2018 IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, Aug. 2018, pp. 1–4.

[100] B. Vishwanath, T. Nanjundaswamy, and K. Rose, « Rotational motion model for temporal prediction in 360 video coding », *in*: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2017, pp. 1–6.

[101] G. K. Wallace, « The JPEG still picture compression standard », *in*: *IEEE Transactions on Consumer Electronics* 38.*1* (Feb. 1992), pp. xviii–xxxiv.

[102] L. Wang and Y. Kim, « Linear code duality between channel coding and Slepian-Wolf coding », *in*: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept. 2015, pp. 147–152.

[103] R. P. Westerlaken, S. Borchert, R. K. Gunnewiek, and R. L. Lagendijk, « Analyzing Symbol and Bit Plane-Based LDPC in Distributed Video Coding », *in*: *2007 IEEE International Conference on Image Processing*, vol. 2, Sept. 2007,

[104] M. Wien, *High Efficiency Video Coding: Coding Tools and Specification*, Signals and Communication Technology, Springer Berlin Heidelberg, 2014, ISBN: 9783662442753.

[105] A. Wyner, « Recent results in the Shannon theory », *in*: *IEEE Transactions on Information Theory* 20.*1* (Jan. 1974), pp. 2–10.

[106] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, « Recognizing scene viewpoint using panoramic place representation », *in*: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 2695–2702.

[107] T. Yamasaki and K. Aizawa, « Patch-based compression for Time-Varying Meshes », *in*: *2010 IEEE International Conference on Image Processing*, Sept. 2010, pp. 3433–3436.

[108] Y. Yang and Y. Zhang, « A High-Realistic Texture Mapping Algorithm Based on Image Sequences », *in*: *2018 26th International Conference on Geoinformatics*, June 2018, pp. 1–8.

[109] F. Ye, E. Dupraz, Z. Mheich, and K. Amis, « Optimized Rate-Adaptive Protograph-Based LDPC Codes for Source Coding With Side Information », *in*: *IEEE Transactions on Communications* 67.*6* (June 2019), pp. 3879–3889.

[110] M. Yu, H. Lakshman, and B. Girod, « A Framework to Evaluate Omnidirectional Video Coding Schemes », *in*: *2015 IEEE International Symposium on Mixed and Augmented Reality*, Sept. 2015, pp. 31–36.

[111] Matt Yu, Haricharan Lakshman, and Bernd Girod, « Content Adaptive Representations of Omnidirectional Videos for Cinematic Virtual Reality », *in*: *Proceedings of the 3rd International Workshop on Immersive Media Experiences*, ImmersiveME '15, Brisbane, Australia: ACM, 2015, pp. 1–6.

[112] A. Zare, A. Aminlou, and M. M. Hannuksela, « Virtual reality content streaming: Viewport-dependent projection and tile-based techniques », *in*: *2017 IEEE International Conference on Image Processing (ICIP)*, Sept. 2017, pp. 1432–1436.

[113] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, « HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications », *in*: *Proceedings of the 24th ACM International Conference on Multimedia*, MM '16, Amsterdam, The Netherlands: ACM, 2016, pp. 601–605.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

# Titre: Compression pour la communication interactive de contenus visuels

**Mot clés :** compression interactive, codes incrémentaux, accès aléatoire, transmission dépendante de l'utilisateur, image à 360 degrés, texture modèle 3D

**Resumé :** Les images et vidéos interactives ont récemment vu croître leur popularité. En effet, avec ce type de contenu, les utilisateurs peuvent naviguer dans la scène et changer librement de point de vue. Les caractéristiques de ces supports posent de nouveaux défis pour la compression. D'une part, les données sont capturées en très haute résolution pour obtenir un réel sentiment d'immersion. D'autre part, seule une petite partie du contenu est visualisée par l'utilisateur lors de sa navigation. Cela induit deux caractéristiques : une compression efficace des données en exploitant les redondances au sein du contenu (pour réduire les coûts de stockage) et une compression avec accès aléatoire pour extraire la partie du flux compressé demandée par l'utilisateur (pour réduire le débit de transmission). Les schémas classiques de compression ne peuvent gérer de manière optimale l'accès aléatoire, car ils utilisent un ordre de traitement des données fixe et prédéfini qui ne peut s'adapter à la navigation de l'utilisateur.

Le but de cette thèse est de fournir de nouveaux outils pour les schémas interactifs de compression d'images. Pour cela, comme première contribution, nous proposons un cadre d'évaluation permettant de comparer différents schémas interactifs de compression d'image / vidéo. En outre, des études théoriques antérieures ont montré que l'accès aléatoire peut être obtenu à l'aide de codes incrémentaux présentant le même coût de transmission que les schémas non interactifs au prix d'une faible augmentation du coût de stockage. Notre deuxième contribution consiste à créer un schéma de codage générique pouvant s'appliquer à divers supports interactifs. À l'aide de ce codeur générique, nous proposons ensuite des outils de compression pour deux modalités d'images interactives : les images omnidirectionnelles (360 degrés) et les cartes de texture de modèle 3D. Nous proposons également de nouvelles représentations de ces modalités. Enfin, nous étudions l'effet de la sélection du modèle sur les taux de compression de ces codeurs interactifs.

# Title: Compression for interactive communication of visual contents

**Keywords :** interactive compression, incremental codes, random access, user-dependent transmission, 360° image, 3D model texture map

**Abstract:** Interactive images and videos have received increasing attention due to the interesting features they provide. With these contents, users can navigate within the content and explore the scene from the viewpoint they desire. The characteristics of these media make their compression very challenging. On the one hand, the data is captured in high resolution (very large) to experience a real sense of immersion. On the other hand, the user requests a small portion of the content during navigation. This requires two characteristics: efficient compression of data by exploiting redundancies within the content (to lower the storage cost), and random access ability to extract part of the compressed stream requested by the user (to lower the transmission rate). Classical compression schemes can not handle random accessibility because they use a fixed pre-defined order of sources to capture redundancies.

The purpose of this thesis is to provide new tools for interactive compression schemes of images. For that, as the first contribution, we propose an evaluation framework by which we can compare different image/video interactive compression schemes. Moreover, former theoretical studies show that random accessibility can be achieved using incremental codes with the same transmission cost as non-interactive schemes and with reasonable storage overhead. Our second contribution is to build a generic coding scheme that can deal with various interactive media. Using this generic coder, we then propose compression tools for 360° images and 3D model texture maps with random access ability to extract the requested part. We also propose new representations for these modalities. Finally, we study the effect of model selection on the compression rates of these interactive coders.