**Impact des ROI dans le streaming vidéo**

Encadrant : **Olivier FOURMAUX**

Etudiants : **Sonia LOUNIS, Fabien MANSON et Alexandre MAZARS**

Table des matières

[1 Cahier des charges 2](#_Toc40890949)

[1.1 Présentation du projet 2](#_Toc40890950)

[1.1 Maintenant et évolution du projet 2](#_Toc40890951)

[2 Plan de développement 3](#_Toc40890952)

[3 Bibliographie 3](#_Toc40890953)

[4 Analyse 3](#_Toc40890954)

[5 Conception 3](#_Toc40890955)

[6 Compte rendu 3](#_Toc40890956)

# Cahier des charges

# 1.1 Présentation du projet

Ce projet est réalisé dans le cadre de l’UE PRES de l’année 2019, Il est proposé par Mr. Olivier FOURMAUX et porte sur l’étude de l’impact du streaming vidéo adaptatif et la mise en place d’une démonstration de cette technologie.

**1.2 Objectif du projet**

Dans un premier temps nous allons étudier le fonctionnement de l’encodage différencié des régions d’intérêts ainsi que la transmission vidéo via http. Dans un second temps nous mettrons en place un système de codage et décodage adaptatif spatial. Enfin nous devrons présenter une démonstration de l’encodage différencié au travers d’un casque de réalité virtuelle ou via une émulation sur un écran et de son impact sur le réseau.

**1.3 Contexte du projet**

Ce projet s’inscrit dans la dynamique actuelle de la diffusion massive de contenu vidéo haute résolution qui ne cesse d’augmenter. Le développement de la réalité virtuelle et du contenu vidéo en VR représente une partie du futur de la diffusion de contenu vidéo. On peut trouver quelques études qui ont déjà travaillé sur ce sujet en proposant des méthodes d’optimisation de la transmission de vidéo UHD et du découpage de celle-ci en régions d’intérêt.

**1.4 Contraintes liées au projet**

Contrainte de temps le projet est à rendre en mai 2020 en prévision d’une soutenance de ce dernier un peu plus tard le mois. Les dates ne sont pas encore précisées. Les dates ont été changées à cause de l’épidémie du Covid-19, le rapport est à rendre le 10 juin 2020 et la soutenance aura lieu le 15 ou le 16 juin 2020 probablement en visio-conférence.

**1.5 Documentation du projet**

La documentation de la partie technique du projet sera minimaliste. Elle pourra se trouver sous l’une des deux formes suivantes : directement dans le code du projet sous la forme de commentaire de code très détaillé ou dans un fichier texte annexe (on peut même envisager un fichier PDF si le projet ne prend pas trop de retard).

## Maintenant et évolution du projet

Le projet n’aura plus aucun suivi passé la date de soutenance. Il ne sera plus mis à jours ni même modifié. Cependant ou pourra retrouver l’intégralité du code source sur une page GitHub dédiée.

**- cahier des charges inchangé, c'est le même que celui du rapport 1**

**- plan du développement adapté (mis à jour) + gantt–2–Left.png + gantt–2-Right.png**

**- la bibliographie actualisée [1000-1500 mots] (dans notre cas c'est inutile car les articles que l'on a trouvés sont tous importants)**

**- l'analyse actualisée [1000-1500 mots]**

**- la conception réalisée [1000-1500 mots]**

**- un compte rendu de projet [1500-2500 mots] (détail du déroulement, de la réalisation, la validation et la livraison de ce qui est demandé)**

**- annexes (inutile dans notre cas je pense)**

# Plan de développement

# Bibliographie

La bibliographie reste quasiment la même. Nous avons simplement rajouté quelques sites web tels que le site de GPAC ainsi que la page Wikipédia officielle de GPAC.

1] Tarek El-Ganainy and Mohamed Hefeeda. 2016. Streaming Virtual Reality Content. *arXiv:1612.08350 [cs]* (December 2016). Retrieved December 5, 2019

[2] Dan Grois and Ofer Hadar. 2011. Recent Advances in Region-of-interest Video Coding. In *Recent Advances on Video Coding*. DOI:https://doi.org/10.5772/17789

[3] Mohammad Hosseini and Viswanathan Swaminathan. 2016. Adaptive 360 VR Video Streaming: Divide and Conquer! (September 2016). Retrieved December 5, 2019 from <https://arxiv-org.accesdistant.sorbonne-universite.fr/abs/1609.08729v5>

[4] JongBeom Jeong, Dongmin Jang, Jangwoo Son, and Eun-Seok Ryu. 2018. 3DoF+ 360 Video Location-Based Asymmetric Down-Sampling for View Synthesis to Immersive VR Video Streaming. *Sensors* 18, 9 (September 2018), 3148. DOI:https://doi.org/10.3390/s18093148

[5] Christopher Müller and Christian Timmerer. 2011. A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP. In *Proceedings of the 19th ACM International Conference on Multimedia* (MM ’11), 723–726. DOI:https://doi.org/10.1145/2072298.2072429

[6] D. V. Nguyen, Huyen T. T. Tran, and Truong Cong Thang. 2019. A client-based adaptation framework for 360-degree video streaming. *Journal of Visual Communication and Image Representation* 59, (February 2019), 231–243. DOI:https://doi.org/10.1016/j.jvcir.2019.01.012

[7] Duc V. Nguyen, Huyen T. T. Tran, and Truong Cong Thang. 2019. Adaptive Tiling Selection for Viewport Adaptive Streaming of 360-degree Video. *IEICE Transactions on Information and Systems* E102.D, 1 (2019), 48–51. DOI:https://doi.org/10.1587/transinf.2018MUL0001

[8] Cagri Ozcinar, Ana De Abreu, and Aljosa Smolic. 2017. Viewport-aware adaptive 360° video streaming using tiles for virtual reality. *arXiv:1711.02386 [cs]* (November 2017). Retrieved December 5, 2019 from <http://arxiv.org/abs/1711.02386>

[9] Georgios Papaioannou and Iordanis Koutsopoulos. 2019. Tile-based Caching Optimization for 360° Videos. *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing* 171--180 (July 2019). Retrieved December 5, 2019 from <http://graphics.cs.aueb.gr/graphics/docs/papers/MOBIHOC-2019.pdf>

[10] Y. Sánchez de la Fuente, R. Skupin, and T. Schierl. 2017. Video processing for panoramic streaming using HEVC and its scalable extensions. *Multimed Tools Appl* 76, 4 (February 2017), 5631–5659. DOI:https://doi.org/10.1007/s11042-016-4097-4

[11] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (MMSys ’11), 133–144. DOI:https://doi.org/10.1145/1943552.1943572

[12] Alireza Zare, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2016. HEVC-compliant Tile-based Streaming of Panoramic Video for Virtual Reality Applications. In *Proceedings of the 24th ACM International Conference on Multimedia* (MM ’16), 601–605. DOI:https://doi.org/10.1145/2964284.2967292

[13] Alireza Zare, Maryam Homayouni, Alireza Aminlou, Miska M. Hannuksela, and Moncef Gabbouj. 2019. 6K and 8K Effective Resolution with 4K HEVC Decoding Capability for 360 Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 15, 2s (July 2019), 68:1–68:22. DOI:https://doi.org/10.1145/3335053

[14] GPAC Multimedia Open Source Project, Telecom Paris, [*https://gpac.wp.imt.fr*](https://gpac.wp.imt.fr)*,*  (dernière mise à jour en Octobre 2019), site consulté entre Mars et Juin 2020

[15] GPAC Project on Advanced Content, Wikipedia The Free Encyclopedia, [*https://en.wikipedia.org/wiki/GPAC\_Project\_on\_Advanced\_Content*](https://en.wikipedia.org/wiki/GPAC_Project_on_Advanced_Content), page créée en 2008, dernière mise à jour de la page en mars 2020, site consulté entre Mars et Juin 2020

[16] GPAC Code source, Github.com, [*https://github.com/gpac/gpac*](https://github.com/gpac/gpac), Dernière mise à jour du code source le 3 Juin 2020, site consulté régulièrement entre Mars et Juin 2020

# Analyse [1000 – 1500 mots] current 994

Le concept de région d’intérêt a été mis en place dans le but de réduire la charge réseau tout en conservant un maximum de qualité vidéo. Dans ce projet on s’intéresse particulièrement à son impact sur les vidéos en 360 degré car c’est là où elle a le plus d’impact. En effet, lorsqu’on visionne une vidéo en 360 degré, notre champ de vision est réduit à une petite partie de la vidéo car cela évite de voir l’effet de déformation lié à la projection cylindrique équidistante (type de projection le plus courant pour passer d’une sphère a un plan). Sans utiliser les régions d’intérêt, toute la partie de la vidéo qui se trouve en dehors du champ de vision est quand même téléchargée depuis le serveur vers le client ce qui pose en problème de bande passante. La région d’intérêt permet de définir une région particulière (généralement la zone que regarde l’utilisateur) qui sera prioritaire par rapport aux autres zones. Un débit élevé et donc une meilleur qualité d’image sera attribuée aux régions ayant une forte priorité. Le reste des régions moins prioritaires auront un débit attribué plus faible permettant de réduire grandement la charge de la bande passante. Le choix de la région d’intérêt est libre et arbitraire, il peut s’agir de la région ou regarde l’utilisateur (capteurs d’un casque VR, pointeur de souris), d’une zone fixe ou encore d’une zone importante dans la vidéo par exemple une zone avec de nombreux détails nécessitant une meilleure qualité d’image. Ce principe de région d’intérêt est très important car il ne faut pas oublier qu’une vidéo en 360° est au minimum en 4k 30ips (ultra haute définition). La plupart du temps les vidéos ont une résolution bien supérieur à de la simple 4K, on trouve notamment de la 6K et même dans certains cas de la 8K (images de respectivement 6000 et 8000 pixels de large). Voici un exemple pour mieux comprendre l’intérêt du streaming dynamique et des régions d’intérêt. Nous disposons d’une vidéo en 4k60ips (image de 4096x2160 pixels) avec un débit binaire de 35Mbits/s. Nous créons un flux mpd avec différents bitrates, par exemple 15Mbits/s, 5 Mbits/s et 1Mbits/s. Dans le cas où le client dispose d’une bande passante inférieure à 35 Mbits/s et qu’il souhaite visionner la vidéo sans utiliser le streaming dynamique, son expérience de visionnage ne sera pas bonne. La vidéo sera saccadée car le débit de lecture est supérieur au débit de chargement de la vidéo. Une solution consiste à laisser charger la vidéo intégralement en mémoire pour la lire ensuite mais ce n’est pas envisageable pour des films qui, en très haute résolution, pèsent plusieurs dizaines de giga octets. La seconde solution intéressante est d’avoir recours au streaming dynamique. Certes le client ne dispose pas d’une bande passante suffisante pour visionner la vidéo dans sa résolution native, mais grâce aux différents bitrates encodés et disponibles via le flux décrit par le fichier .mpd, il peut visionner la vidéo de manière fluide. C’est le lecteur qui se charge de sélectionner le bitrate correspondant à la bande passante disponible. Les conséquences de l’utilisation d’un tel système sont une qualité d’image dégradée et une charge d’encodage en amont élevée. Pas de secret, pour obtenir plusieurs versions d’une vidéo avec des bitrates différents il faut encoder la vidéo de multiples fois ce qui requiert une puissance et un temps de calcul élevée.

La méthodologie pour mettre en place un système de stream de vidéo utilisant le principe de région d’intérêt est toujours la même. Dans un premier temps, il faut trouver un média source. De préférence libre de droit est ayant une résolution d’image au moins égale à de la 4K avec une fréquence d’image de 30ips (images par secondes) au minimum. On peut Notamment utiliser le site Mettle.com qui propose quelques vidéos d’une trentaine de secondes en VR filmées depuis un drone ou une caméra statique. Une fois que l’on a téléchargé le média source, il faut appliquer un découpage en tuiles selon une grille de taille choisie, généralement une grille carrée de 5x5 tuiles. Ce découpage en tuiles peut se faire avec des encodeurs comme ffmpeg et kvazaar. Le choix du découpage est totalement arbitraire, On peut choisir de découper des tuiles de même taille dans la vidéo on de découper de zones de taille et de formes variées. On peut par exemple ne découper qu’une zone dans la partie de la vidéo correspondant au bas/haut de la vidéo VR qui sont statistiquement les zones les moins regardées dans la sphère complète. Il faut aussi produire des clones du média source avec une qualité d’image volontairement dégradée. On prend par exemple la vidéo source en pleine résolution, et on crée d’autres clones, ayant chacun des résolutions de plus en plus faibles. Par exemple avec un média source ayant 4096p (pixels) de large, on peut créer des clones ayant les résolutions suivantes 2560p, 1920p, 720p, 360p. Une fois le découpage effectué, on obtient pour chaque tuile plusieurs segments de différentes résolutions. Il faut ensuite créer un fichier descripteur de flux mpeg-dash (.mpd) qui permettra de localiser les différentes tuiles et les différentes résolutions disponibles pour chaque tuile. Ensuite il ne reste plus qu’à trouver un lecteur qui permette d’ouvrir un flux mpeg-dash ce qui n’est pas chose aisée car la plupart des lecteurs sont des technologies d’entreprises privées comme YouTube ou Facebook. Il existe cependant un projet open source de lecteur dash en JavaScript et GPAC qui est un lecteur multimédia capable de lire ce type de flux. Le rôle du lecteur n’est pas si simple que cela car il doit non seulement afficher la vidéo de manière fluide (ce qui n’est pas trivial avec des très hautes résolutions) mais aussi télécharger les bonnes tuiles correspondant à la bande passante disponible. Le lecteur GPAC est donc une très bonne option car il est bien documenté et très complet c’est pourquoi nous l’utilisons pour la partie décodage et affichage du flux vidéo.

# Conception [1000 – 1500 mots] current 1084

La méthodologie que nous avons suivie est quasiment identique à celle disponible sur la page GitHub[[1]](#footnote-1) de GPAC, elle y est décrite assez succinctement mais avec suffisamment d’exemples pour que cela soit compréhensible mais surtout reproductible avec notre environnement de test.

Nous avons donc dans un premier temps récupéré un média source sur le site Mettle.com[[2]](#footnote-2). Nous avons ensuite modifié le média source en le copiant en miroir pour doubler sa durée, car 30 secondes ce n’est pas assez pour observer un changement de débit majeur (car la taille minimale recommandée du buffer d’images est de 1 seconde et que le lecteur met un certain temps à passer d’un flux à un autre). Le média résultant est une vidéo au format MP4 d’environ une minute de durée. Pour le découpage en tuiles, L’équipe GPAC recommande d’utiliser l’encodeur Kvazaar, Nous avons Donc installé et utilisé Kvazaar pour découper le média en tuiles. Nous avons découpé le média avec à chaque fois le même type de découpage en tuiles de même taille. Nous avons juste fait varier le nombre de tuiles en prenant des nombres de tuiles impair (3x3, 5x5, 7x7, etc…) pour toujours avoir une tuile centrale car un des algorithmes de démonstration de la zone d’intérêt implémenté dans le lecteur choisi le centre de l’image comme zone prioritaire. Il est ainsi plus facile de voir un changement de débit si une seule tuile se trouve au centre au lieu de deux. Le découpage en tuile est une opération extrêmement lourde au niveau encodage. A titre d’exemple sur un CPU dual Core 2.6Ghz découper 1 minute de vidéo en 4K 30ips prend plus de 1h30 et il ne faut pas oublier qu’il faut refaire l’encodage à chaque fois que l’on change la résolution ou le nombre de tuiles. Nous avons remarqué au cours de nos différents essais de découpage en tuile que certaines configurations ne sont pas stables. Au-delà de 49 tuiles par image (7x7 tuiles de même taille), le lecteur ne parvient pas décoder le flux correctement et plante après quelques secondes de vidéo. Pour éviter de devoir attendre pendant des heures et des heures la fin de l’encodage nous avons créé un script bash qui prend une vidéo source en entrée et génère le fichier descripteur de flux en découpant au préalable la vidéo en tuiles et en générant plusieurs clones avec des résolutions plus petites. Dans un premier temps, il faut convertir le média source (qui est au format mp4) au format YUV car Kvazaar ne prend pas en charge d’autre format que le YUV. Nous avons réalisé cette conversion en utilisant ffmpeg, un encodeur vidéo très puissant. La commande que nous utilisons pour convertir le média est la suivante :

ffmpeg –threads 4 –i media\_source.mp4 media\_source.yuv

Il est important de noter que nous n’utilisons aucune forme de compression de fichier. Cela permet de réduire le temps d’encodage et de préserver au maximum la qualité original du média source. L’option « -threads » permet d’utiliser le multithreading dans l’encodage d’un fichier. Elle est définie à 4 car nous ne disposons que de processeur à 2 ou 4 cœurs maximum.

La seconde étape consiste au découpage en tuiles du média source toujours au format YUV. Les encodeurs kvazaar et ffmpeg sont tous les deux capables de réaliser un découpage en tuile cependant kvazaar est utilisé dans les démonstrations de l’équipe GPAC. C’est pourquoi nous avons décidé de l’utiliser. Nous utilisons donc kvazaar avec les options suivantes :

* la source : -i <media\_src.yuv>
* la résolution d’entrée : --input-res largeur\*hauteur
* la sortie : -o <media\_out.hvc>
* le découpage en tuiles : --tiles nb\_X\*nb\_Y
* le bitrate : -bitrate <valeur\_en\_bit>

Il est important de noter que les valeurs nb\_X et nb\_y qui correspondent au respectivement aux nombres de tuiles en largeur et en hauteur sont indépendants, on peut par exemple faire un découpage en 5x7 pour avoir 5 tuiles en largeur et 7 en hauteur. Il est important de noter que kvazaar encode chaque tuile dans un fichier hvc séparé. Cela veut dire que plus le nombre total de tuile est grand, plus le nombre de fichier créés est élevé ce qui, dans certains cas, peut poser problème comme nous le verrons dans la partie compte rendu de projet.

Après avoir découpé le media source et les différents clones en tuiles, il faut utiliser MP4Box pour rassembler les tuiles en différents segments de résolutions. Toutes les tuiles d’une même zone qui ont des résolutions différentes sont assemblées en un seul segment. Les segments seront eux même inclus dans le fichier descripteur de flux (mpd).

Une fois que le descripteur de flux est créé, il faut trouver un lecteur média capable de décoder un tel fichier. Cela tombe bien car l’équipe GPAC a aussi développé un lecteur multimédia (MP4Client) capable de lire un descripteur de flux VR tout en utilisant un algorithme de sélection de tuiles. MP4Client est, à l’origine, développé sous MacOs mais y a plusieurs versions portées sous Linux et Windows. Aucun des membres du groupe ne possède de système Apple nous avons donc utilisé les deux autres versions. Dans un premier temps nous avons testé de lire directement le flux en local pour vérifier le bon fonctionnement du lecteur. Le lecteur MP4Client est capable d’ouvrir, de lire et d’adapter le débit vidéo à la bande passante. Un détail important de ce lecteur est qu’il est aussi disponible sur les plateformes mobiles et qu’il est capable d’afficher la vidéo au format casque VR (que l’on soit sur une version mobile ou PC) avec une séparation pour chaque œil. Le lecteur propose aussi une option très intéressante permettant de limiter la bande passante du lecteur pour simuler différentes configurations réseau.

Une fois que tout le traitement vidéo et la création du fichier descripteur de flux est terminée, nous avons mis en place un réseau de test. Nous avons un ordinateur portable sous Debian 9 branché à un routeur qui sert de serveur DASH, il contient tous les segments de la vidéo ainsi que le descripteur de flux. De l’autre côté, connecté au routeur, une autre machine portable tournant sous Windows et Linux avec le lecteur GPAC installé sur chaque OS. Toutes les machines sont connectées via des câbles Ethernet (cat 5E vitesse maximum 1Gbit/s). La machine client possède un double OS pour tester les deux versions de GPAC. La bande passante locale mesurée (via transfert de fichier local) sur le réseau de test est d’environ 100Mbit/s dans la meilleure configuration.

Ajouter une partie montrant les résultat

# Compte rendu de projet [1500 – 2000 mots] current 220

INTRODUCTION

Le but de ce projet était de mettre en place une chaine de streaming dynamique complète en utilisant l’encodage différencié avec les régions d’intérêt.

DEVELOPPEMENT

Ce que l’on a réussi à faire

Les problèmes rencontrés et les solutions au pb

Pb de stabilité du lecteur GPAC, que ça soit sous Windows ou linux, il crash quand on cherche à changer les réglages par défaut. Pour remédier à ce problème nous avons tenté d’utiliser un autre lecteur JavaScript dynamique capable de lire un flux mpd le lecteur dash.js mais sans succès car seulement Edge (navigateur web de Microsoft) prend en charge un flux dynamique. Nous n’avons malheureusement pas réussi à implémenter le lecteur dans une page web sans erreur du lecteur. Cette option nous prenant beaucoup trop de temps et n’étant pas satisfaisante nous avons décidé de l’abandonner et de revenir au lecteur GPAC et d’essayer de le paramétrer pour éviter les crashs. Ce que nous avons quasiment réussi à faire !

CONCLUSION

Conclusion générale du projet

Beaucoup d’inconnues perturbent les résultats : bande passante sur nos machines, box, internet, la puissance du pc, gpac pas toujours stable et très limité niveau puissance…

Idéal : refaire les mêmes tests dans une configuration de laboratoire.

OUVERTURE

Il reste de nombreux points à améliorer….

L’idéal serait d’avoir une adaptation pixel par pixel permet de ne pas voir le découpage des tuiles.

Il serait intéressant de tester la même configuration réseau amis sous MacOs pour vérifier que la stabilité de l’application n’est pas responsable d’erreur de mesures.

Nombre max de fichier fait crash l’explorateur de GPAC client

Ouverture : MacOs mieux que Windows car installation et stabilité de GPAC trop faible

1. [*https://github.com/gpac/gpac/wiki/HEVC-Tile-based-adaptation-guide*](https://github.com/gpac/gpac/wiki/HEVC-Tile-based-adaptation-guide) [↑](#footnote-ref-1)
2. <https://vimeo.com/214402865> [↑](#footnote-ref-2)