



**Rapport de stage 2A : Stage d'application**

**Digital twin and streaming 3D**

Structure d'accueil du stagiaire :

UMR 5218 - IMS - Laboratoire de l'Intégration du Matériau au Système

Maître de stage : Mr. Yannick Berthoumieu

Rapporteur : Mr. Charles Consel

Durée du stage : Du 23 mai au 12 septembre 2025

Debache Raphaël

ENSEIRB-MATMECA

Télécommunication 2ème année

**Résumé :**

Le développement des technologies de réalité virtuelle, des caméras de profondeur et des réseaux haut débit (5G/6G) ouvre de nouvelles perspectives pour la téléopération robotique immersive. Dans ce contexte, la transmission en continu de flux volumétriques (nuages de points dynamiques) représente un défi majeur en raison des contraintes de **débit réseau** et de **latence**.

Ce stage avait pour objectif d'étudier et de comparer différentes approches de compression et de transmission de flux vidéo volumétrique, afin de proposer une solution adaptée aux contraintes d'une application de téléopération immersive. Une analyse des normes MPEG récentes (V-PCC, MIV) a montré que leur maturité et la latence induite ne permettaient pas de répondre aux besoins du projet. Une solution alternative a donc été développée, basée sur une **transformation du format de profondeur (Z16) en YUV**, suivie d'une **compression vidéo via GStreamer** et d'une transmission en continu sur une architecture client-serveur.

Les travaux réalisés ont permis de mettre en place une chaîne de compression et de transport fonctionnelle, intégrée à une application Unity pour la visualisation en casque de réalité virtuelle. Des tests ont validé la faisabilité de la solution en local, avec des mesures de débit et de latence conformes aux contraintes du cahier des charges.

Ce stage m'a également permis d'acquérir des compétences en **compression vidéo, programmation en C++, développement avec GStreamer et intégration dans un environnement VR/Unity**, tout en ouvrant la voie à des perspectives de déploiement sur réseaux 5G/6G et à une intégration sur robot en conditions réelles.

## Sommaire

- 1. Présentation de l'entreprise et du contexte**
  - Historique et missions du laboratoire UMR 5218 – IMS
  - Équipe MOTIVE et axes de recherche
  - Contexte du projet de téléopération robotique immersive
- 2. Cahier des charges du projet**
  - Problématique et enjeux
  - Objectifs du projet
  - Périmètre du projet
  - Besoins fonctionnels et techniques
- 3. Moyens et ressources**
  - Matériel : caméra Intel RealSense D455, casque Meta Quest 3, ordinateurs, routeur
  - Logiciel : Unity, GStreamer, bibliothèques C++ et plugins Unity
- 4. Modalités de suivi et validation**
  - Organisation des réunions et jalon
  - Critères de validation : débit, latence, qualité visuelle
- 5. Contraintes**
  - Matérielles
  - Utilisateur / ergonomie
  - Performance du système
- 6. Livrables attendus**
- 7. Analyse de l'existant**
  - Solutions et normes existantes : V-PCC, MIV, uvgrTP
  - Limites des solutions standards
- 8. Solution développée**
  - Transformation Z16 → YUV
  - Compression et transport avec GStreamer
  - Architecture générale
  - Plugin GStreamer Unity
  - Bibliothèque et plugin Realsense
- 9. Validation et résultats**
  - Expérience utilisateur
  - Mesures des performances : débit, latence, PSNR

- Résultats vidéo (scènes test)

#### **10. Retour sur expérience**

- Apprentissage technique : C++, GStreamer, VR, nuages de points
- Gestion de projet et collaboration

#### **11. Conclusion**

#### **12. Bibliographie**

#### **13. Glossaire**

- AR, VR, XR, MR
- Formats d'image et vidéo : RGB, YUV, RGBD, Z16
- Protocoles et normes : SLAM, UDP, RTP, RTCP, H264, GOP, V-PCC, MIV, UVG, MPEG
- Indicateurs de qualité : PSNR
- Autres termes : MTU, I-frame, P-frame, B-frame

#### **14. Annexe : Gestion de projet**

- Planification et organisation
- Communication et coordination
- Gestion des ressources et des risques
- Suivi, validation et livrables
- Retour personnel sur la gestion de projet

## Présentation de l'entreprise et du contexte

Le laboratoire de l'Intégration du Matériaux au Système a été créé le 1er janvier 2007, par la fusion de trois unités de recherche bordelaises, avec une stratégie scientifique commune de développement principalement centrée dans le domaine des Sciences et de l'Ingénierie des Systèmes, à la convergence des Sciences et Technologies de l'Information et de la Communication , et des Sciences pour l'Ingénieur .

Le laboratoire est rattaché à trois tutelles, le CNRS, l'Université de Bordeaux et Bordeaux Aquitaine INP.

Au CNRS, l'UMR5218 est rattachée en principal à CNRS Ingénierie et en secondaire à CNRS Sciences Informatiques.

Axée sur la conception de nouveaux algorithmes, l'équipe MOTIVE (MOdels, Texture, Images, VolumEs) consacre son activité à l'inférence de grandeurs physiques, à la restauration ou à la reconstruction d'images afin de caractériser des phénomènes observables ou partiellement observables. En ce qui concerne son thème de recherche, MOTIVE s'intéresse particulièrement aux textures, qu'elles soient planaires (2D) ou solides (3D). Il entend proposer de nouveaux algorithmes non seulement pour l'analyse, la classification ou la synthèse de textures mais aussi pour la détection ou la reconstruction d'objets dans des images ou des volumes texturés.

Le projet de téléopération de robot avait déjà été proposé à d'autres étudiants auparavant par mon maître de stage. Une partie de ces derniers a étudié les différents types d'application que l'on pouvait développer avec le casque et ont décidé d'intégrer une certaine plateforme de développement utilisée pour les jeux vidéos (Unity) tandis que l'autre partie des étudiants a développé un script python permettant de guider le robot en détectant les mouvements de l'opérateur à partir des données d'une caméra volumétrique.

Le stage a eu lieu au sein des bâtiments de l'ENSEIRB-MATMECA où j'ai travaillé avec un autre étudiant de l'école en stage lui aussi. Le but du projet était de développer un système de téléopération qui permette à un utilisateur de contrôler un robot en continu et à distance et en même temps de pouvoir être immergé dans l'environnement du robot à l'aide d'un flux vidéo volumétrique en continu.

## Cahier des charges du projet

### Projets et Enjeux



Source : [realite-virtuelle.com](http://realite-virtuelle.com)

Lors de la téléopération d'un robot, la qualité de perception de l'environnement est cruciale : elle conditionne la réactivité de l'opérateur face aux imprévus. Une vision limitée à deux dimensions complique l'estimation des profondeurs et accroît la charge cognitive, tandis qu'une latence trop élevée augmente significativement le risque d'erreur, notamment dans les contextes critiques (sauvetage, nucléaire, médical).

Ce stage explore des pistes pour améliorer l'acuité visuelle de l'opérateur grâce à la réalité virtuelle et aux caméras à nuages de points. L'intégration d'un casque à réalité virtuelle ouvre aussi la voie à la réalité augmentée, en permettant d'afficher en plus des données capteurs sans masquer l'environnement réel. Les caméras volumétriques offrent quant à elles une représentation précise et mesurable de la scène grâce à la connaissance des caractéristiques physiques de la caméra et les techniques de time of flight intégrées dans la caméra. L'un des principaux défis techniques réside dans la réduction de la latence, qui doit rester inférieure aux seuils critiques (environ 100 ms pour une expérience immersive fluide). L'utilisation d'algorithmes de compression et protocoles de transport adaptés, constitue une réponse pragmatique et efficace.

## Objectifs du projet

L'objectif est de permettre à l'opérateur de mieux apprécier la profondeur et la disposition spatiale des éléments de l'environnement du robot sans latence et en continu. L'idée envisagée est d'afficher dans un casque à réalité virtuelle, un nuage de points dynamique représentant la scène dans laquelle se trouve le robot. Ce flux en continu serait capté par une caméra volumétrique qui serait disposée au niveau de la tête du robot humanoïde. Cela permettrait à l'utilisateur de pouvoir se balader dans le nuage de points à l'aide des fonctionnalités du casque et donc dans la scène dans laquelle le robot doit être opéré. Mon travail a été de développer une solution de transfert en continu du flux vidéo volumétrique et la visualisation du flux. Lorsque l'autre stagiaire s'est occupé d'intégrer la solution de contrôle du robot et celle concernant mon travail autour d'une application Unity.

## *Périmètre du projet.*

L'environnement du robot étant amené à évoluer rapidement, seule la zone directement visible par la caméra est retranscrite. Le robot peut modifier son champ de vision en orientant la tête, mais l'acquisition reste limitée à ce périmètre.

Le travail mené dans ce stage s'est concentré sur le développement et la mise en œuvre de techniques de compression vidéo et de téléopération en continu, dans un cadre expérimental. Les tests ont été réalisés en local, sur une architecture simple de type client-serveur, afin de valider la faisabilité et les performances du système.

En revanche, certaines dimensions sortent du périmètre : l'intégration complète sur robot en conditions réelles, la conception d'une interface utilisateur avancée ou encore la réalisation de tests à grande échelle. Ces aspects, bien que pertinents pour des développements futurs, n'ont pas pu être abordés compte tenu de la durée et de la portée du stage.

### ***Besoin fonctionnels et techniques du projet***

L'utilisateur doit pouvoir :

- Utiliser le système sur le casque choisi.
- Allumer et éteindre le flux vidéo volumétrique depuis le casque.
- Voir les paramètres de connexion avec le robot depuis le casque.
- Se déplacer dans l'environnement 3D du robot.
- Contrôler le robot en même temps qu'il visualise l'environnement du robot.
- Utiliser le système de manière intuitive :
  - Le flux retour du robot doit être avec une latence faible.
  - Le flux retour du robot doit offrir une qualité visuelle suffisante.
  - Le flux retour du robot doit être stable (gestion de la gigue réseau (jitter), pertes de paquets, reconnexion automatique)

### ***Moyens et ressources***

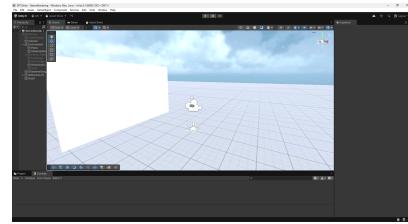
- Caméra Intel-Realsense D455 : Pourvue de 2 capteurs (stéréoscopiques) et un projecteur à infra-rouge, d'un capteur image couleur RGB, d'une unité de mesure inertie et d'une unité de traitement, cette caméra permet d'obtenir une vidéo volumétrique, en 3D, de l'environnement du robot. Les flux fournis par la caméra qui seront utilisés durant ce stage sont : un flux couleur au format RGB et un flux profondeur au format Z16 tous deux ont une résolution 1280x720 pixels à 30 images par secondes obtenus après un traitement interne à la caméra à partir des flux des capteurs embarqués. Ces flux correspondent à une projection planaire d'un nuage de points. En effet le format Z16 transcrit pour chaque pixel la profondeur réelle associée en un entier codé sur 16 bits. Celle-ci est obtenue par traitement du flux du capteur stéréo et du projecteur infrarouge. On peut ainsi



retrouver la distance réelle entre la caméra et un objet filmé à l'aide des caractéristiques intrinsèques aux capteurs et extrinsèques, exprimant la disposition relative entre les capteurs sur la caméra. La caméra vient de paire avec un kit de développement logiciel (Intel Realsense SDK) dans lequel de nombreuses fonctions utiles pour l'utilisation de la caméra sont déjà implémentées telles que l'alignement des images en couleur et en profondeur, la conversion des images en couleur et profondeur en nuage de points, des filtres temporels, spatiaux et de remplissage ainsi que d'autres fonctions. Sont utilisés lors du stage le wrapper python et la bibliothèque C++. (source photo : Intel)

- Unity : Moteur de jeu multiplateforme qui permet le développement d'applications VR/XR sur PC Windows.

Composée de plusieurs scènes 3D, une application unity peut comporter de nombreuses fonctionnalités notamment le rendu sur casque VR.



- Deux ordinateurs : Deux ordinateurs avec le système d'exploitation Windows et disposant de cartes graphiques.

● Casque Meta Quest 3 : Il s'agit d'un casque à réalité virtuelle permettant le développement d'applications de réalité augmentée ou de réalité virtuelle en utilisant un rendu stéréoscopique pour donner une impression de 3D à l'utilisateur. Le casque est muni de système de localisation spatiale de l'utilisateur tel que le SLAM et de reconnaissance de mouvement pour le contrôle des applications avec ou sans manette. Une application Meta Quest est disponible, Airlink, et permet de se connecter à un ordinateur muni d'une carte graphique.

De plus grâce à un certain plugin Unity l'on peut connecter l'application Unity au casque. (source photo : Meta)

**For Meta Quest 3**



- Routeur : Utilisé comme un borne Wifi et switch haut débit, le routeur permet de connecter deux ordinateurs au même réseau local.

### ***Modalités de suivi et validation***

Le suivi assuré par le maître de stage se faisait lors de réunions hebdomadaires, menées conjointement avec l'autre stagiaire. Ces rencontres correspondaient à la clôture d'un jalon ou d'un demi-jalon. Pour chaque stagiaire, elles permettaient de discuter des difficultés rencontrées lors de la réalisation, d'éventuelles modalités d'intégration avec le travail de l'autre stagiaire, ainsi que du jalon suivant à développer.

La validation des solutions développées au cours du projet a été réalisée au moyen de tests de débit, de mesure de qualité du flux, du temps de latence, de taux de compression et plus généralement à l'aide d'une expérience de validation qui permettra de vérifier que toutes les fonctionnalités utilisateur demandées sont présentes dans la solution.

Données sur le débit. Test de qualité après compression.

L'expérience de validation correspond à la situation suivante. Deux ordinateurs sont connectés au même réseau local, tous les deux sont connectés au même routeur. Un est branché à la caméra à vidéo volumétrique lorsque l'autre ordinateur est connecté au casque. Un seul programme doit être lancé au préalable sur l'ordinateur connecté à la caméra. Tandis que sur l'autre ordinateur l'utilisateur doit pouvoir connecter le casque au PC et lancer l'application. Une fois le casque porté, l'utilisateur doit pouvoir utiliser toutes les fonctionnalités décrites dans les besoins fonctionnels.

### ***Contraintes***

Contraintes matérielles :

- Caméra Intel RealSense : portée limitée et champ de vision restreint, ce qui conditionne la zone de capture de l'environnement.
- Casque Meta Quest 3 : nécessite un **débit réseau élevé et stable** pour garantir une **expérience fluide**.

Contraintes utilisateur :

- Ergonomie : l'ensemble des **fonctionnalités** doit être **centralisé** pour une utilisation intuitive.
- **Confort** en réalité virtuelle : la solution doit limiter les phénomènes de **cybersickness** (nausées, désorientation) liés à la **VR**.

Contraintes liées aux performances du système :

- Latence : Pour une interaction naturelle, la latence idéale visée est inférieure à **100 ms**, avec une tolérance maximale de **450 ms** en cohérence avec les recommandations de l'UIT-T [1].

### ***Livrables attendus***

Sont fournis à la fin de ce projet, les codes commentés et les programmes client et serveur associés à l'expérience de validation, un schémas d'architecture (client/serveur, pipeline de traitement, intégration Unity), des jeux de tests / scripts de validation, un guide d'utilisation de ces programmes et d'installation des programmes tiers utilisés. Un guide de développement pour que d'autres étudiants puissent poursuivre et améliorer ce projet sera également fourni.

### **Analyse de l'existant**

À l'heure actuelle, il n'existe pas de solution gratuite, complète et open-source permettant d'envoyer et de recevoir en continu un flux vidéo volumétrique. L'enjeu réside dans le **débit élevé** nécessaire à la transmission d'un nuage de points généré par la caméra (environ 800 000 points, nécessitant un débit d'environ 1.1Gbit/s), largement supérieur aux capacités des équipements disponibles.

Il est donc indispensable de mettre en place des **méthodes de compression** afin de réduire la quantité de données à transmettre tout en conservant une qualité suffisante pour la visualisation. L'objectif est de définir et d'implémenter une **chaîne de compression et de transport** spécifique à la vidéo volumétrique, permettant l'affichage en réalité virtuelle d'un nuage de points dynamique représentant fidèlement l'environnement du robot, tout en respectant les contraintes fonctionnelles et techniques identifiées.

**Formats pour rendu 3D :**

*Média immersifs* : Les médias immersifs sont des contenus qui permettent à l'utilisateur de percevoir et interagir avec un environnement virtuel de manière réaliste. Leur immersion se caractérise par le **Degree of Freedom (DoF)**, c'est-à-dire le nombre de dimensions dans lesquelles l'utilisateur peut se déplacer ou orienter sa perception :

- **3DoF (Three Degrees of Freedom)** : rotation de la tête seulement (pitch, yaw, roll) ; l'utilisateur peut regarder autour de lui mais pas se déplacer dans l'espace.
- **6DoF (Six Degrees of Freedom)** : rotation + translation (avant/arrière, gauche/droite, haut/bas) ; l'utilisateur peut se déplacer librement et interagir dans l'environnement virtuel.

Ainsi, plus le DoF est élevé, plus l'expérience immersive est **réaliste et interactive**, car elle intègre à la fois la **position et l'orientation** de l'utilisateur dans l'espace.

*Multi-vue* : Le format multi-vue correspond à plusieurs points de vue d'une même scène par plusieurs capteurs caméra.

*Stéréo* : Ce format est un cas particulier du format Multi-vue, il est en général utilisé pour reproduire l'effet parallaxe de la vision humaine et ainsi pouvoir obtenir une estimation de la profondeur.

*LiDAR* : Il s'agit de l'acronyme de “Light Detection and Ranging” pour “détection et télémétrie par la lumière” et correspond soit à l'appareil soit à la technique de mesure permettant d'obtenir les informations de télémétrie. Le principe de fonctionnement est le suivant, presque toujours à l'aide d'un laser on réalise des mesures sur une surface en émettant par impulsions un faisceau de lumière généré artificiellement et renvoyé par la cible vers son émetteur. La distance est donnée en analysant le temps de trajet entre l'émission d'une impulsion et la détection d'une onde réfléchie (appelée en anglais Tof, Time of flight) et en tenant compte des propriétés de la lumière.

*RGBD* : On appelle vidéo RGB-D, un flux composé en 2 sous-flux. Le premier sous-flux décrit la couleur de la texture observée projetée sur un plan et enregistrée au format RGB et peut être

obtenue à l'aide d'une caméra classique, tandis que le second sous-flux correspond à la distance qui sépare la caméra de l'objet filmé pour chaque pixel. Cette distance peut être obtenue soit à l'aide d'un LiDAR soit à l'aide d'un système stéréo et d'une unité de traitement.

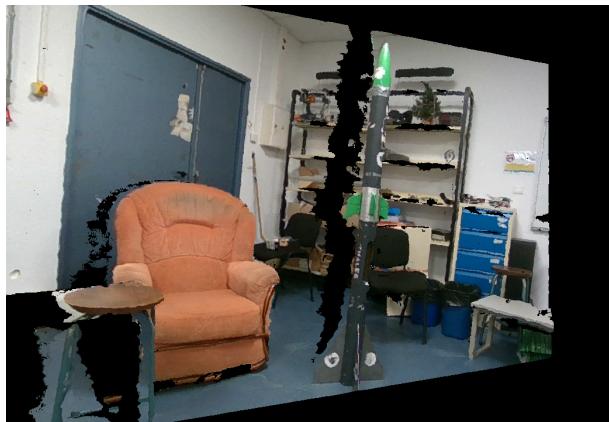


Flux Z16



Flux RGB

*Nuages de points* : Le format nuage correspond à un ensemble de points (ordonné ou non ordonné), chaque point est décrit par 2 triplets ; la position géométrique (X, Y, Z) du point et la couleur du point (R, G, B).



Vue du flux au format nuage de points issu de la caméra sous deux points de vue.

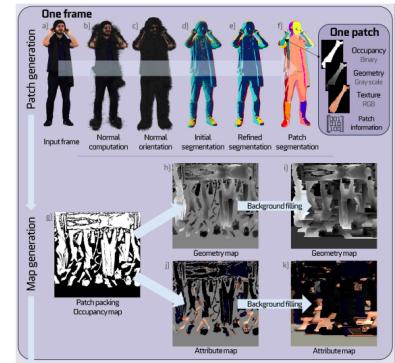
*Vidéo 360* : Appelée également vidéo immersive, il s'agit d'un enregistrement vidéo d'une scène du monde réel où l'image est enregistrée dans toutes les directions en même temps réalisé à l'aide d'un système caméra spécial. Ce format met l'utilisateur au milieu d'une sphère dont il est le centre et permet 3 degrés de liberté ainsi ce format à lui seul ne prend pas en compte l'information en profondeur.

### Normes V-PCC et MIV et utilisation de la bibliothèque uvgRTP.

Les standards *Video-based Point Cloud Compression (V-PCC)* [2] ou *Immersive Video (MIV)* [3] dérivent de la norme *Volumetric Video Compression (V3C)* développée par le groupe MPEG et proposent une méthode state-of-the-art en termes de qualité de compression de média immersifs. Les deux normes de compression visent à projeter l'information 3D en atlas, des images de taille définie au début de la compression pour tirer partie des algorithmes de compression vidéo 2D existant.

Le standard *Video-based Point Cloud Compression (V-PCC)* [2] est développé pour le format de nuages de points dynamiques. Cette compression repose sur le principe suivant : pour un instant donné, le nuage de points est découpé en patchs (groupe de points dont les vecteurs de normales sont colinéaires et orientés dans le même sens).

Sur chaque patch, la couleur et la profondeur sont découpées puis concaténées à des atlas séparés et spécifique aux informations de couleur ou profondeur tandis qu'est créée un autre atlas d'occupation permettant de discriminer sur les atlas, les informations liées au nuage de point. Des informations sur la position des patchs est par ailleurs conservée.



Schema explicatif issu de [2]

Le second standard, *Immersive Video (MIV)* [3], est adapté aux formats image multi-plans ou image multi-vues et profondeurs. Pour chaque images de chaque vidéo, un processus en 3 étapes est opéré : les textures (image et profondeur) redondantes sont détectées et segmentées pour former des patchs mais est gardé une trace des positions des patchs dans les vues avant compression (étape de Pruning), les patchs sont ensuite concaténés dans des atlas (étape de patching), pour finalement être encodés.

Pour cette même norme, l'on peut trouver les implémentations des encodeurs et décodeurs de références (utilisés pour faire les tests de développement par le groupe MPEG) ainsi que les



**Fig. 3.** Input views (left), views after pruning (middle), and after packing into atlas (right); sequence Kitchen.

implémentations réalisées par le groupe Ultra Vidéo Groupe (UVG) qui portent vers le développement de solution pour le public. Le groupe UVG a également développé une bibliothèque de transport basée sur le protocole RTP en intégrant les paquets *Volumetric Video Compression (V3C)* [4].

### Représentation de l'étape du pruning etude [3]

#### **Solution non normée transformation Z16 et gstreamer pour la compression et transport.**

Cette solution s'appuie sur l'étude [7], qui propose un algorithme permettant de convertir le format Z16 (16 bits par pixel) en format YUV (3x8 bits par pixel), mieux adapté aux méthodes classiques de compression vidéo. Selon cette étude, la transformation présente une certaine robustesse face aux algorithmes de compression, qui ont tendance à réduire, voire à supprimer, certaines données. Ainsi, après conversion du format Z16, il devient possible de compresser et de décompresser le flux de la caméra à l'aide du codec H.264.

Dans cette solution la bibliothèque C/C++, Gstreamer, permettrait de réaliser la diffusion vidéo. La bibliothèque Gstreamer existe depuis 1999 et est développée par freedesktop.org qui est un organisme de collaboration entre différents projets de logiciels libres comme GNOME, KDE, Xfce, Enlightenment, Xgl/AIGLX ou encore X.Org. Celle-ci propose de nombreuses fonctionnalités notamment pour les flux vidéos en continu, l'accélération matérielle pour l'encodage et décodage vidéo. Il s'agit d'une ressource stable, souvent mise à jour et avec une importante documentation.

GStreamer fournit une architecture basée sur des pipelines multi-threads, où chaque élément (source, encodeur, sink) peut travailler en parallèle sans bloquer les autres ce qui est adapté aux applications en temps réel. Les queues créent automatiquement des threads séparés pour découpler les traitements. La communication interne est centralisée par le bus, qui regroupe tous les messages (erreurs, changements d'état, EOS) et permet à l'application de les gérer

proprement. Enfin, des signaux asynchrones comme ceux d'appsrv (need-data, enough-data) régulent l'envoi de données. Cela permet de séparer clairement le chemin des données vidéo du chemin des messages de contrôle.

### **Solutions envisageables**

Parmi les solutions décrites ci-dessus, l'utilisation de logiciels référence MPEG a été mise de côté. Bien que la qualité après décompression soit optimale plusieurs tests que nous avons pu opérer et qui ont été vérifiés par cette étude [2], nous indiquent que avec nos équipements la latence induite par la compression du nuage de points ou de la vue texture et profondeur en continu ne correspondait pas aux contraintes données dans le cahier des charges. En effet, la latence cumulée par la compression puis décompression à la fois avec l'encodeur/décodeur MIV ou avec l'encodeur/décodeur TMC dépasse une seconde. Une autre solution aurait été d'utiliser l'encodeur V-PCC développé par UVG qui est beaucoup plus rapide (encodage en dessous de 200 ms), cependant il n'y a pas d'implémentation par le groupe UVG du décodeur et l'utilisation du décodeur de référence donne une latence de plus de 450 ms ce qui ne valide pas les contraintes du cahier des charges.

### Détails de la réalisation de la solution choisie.

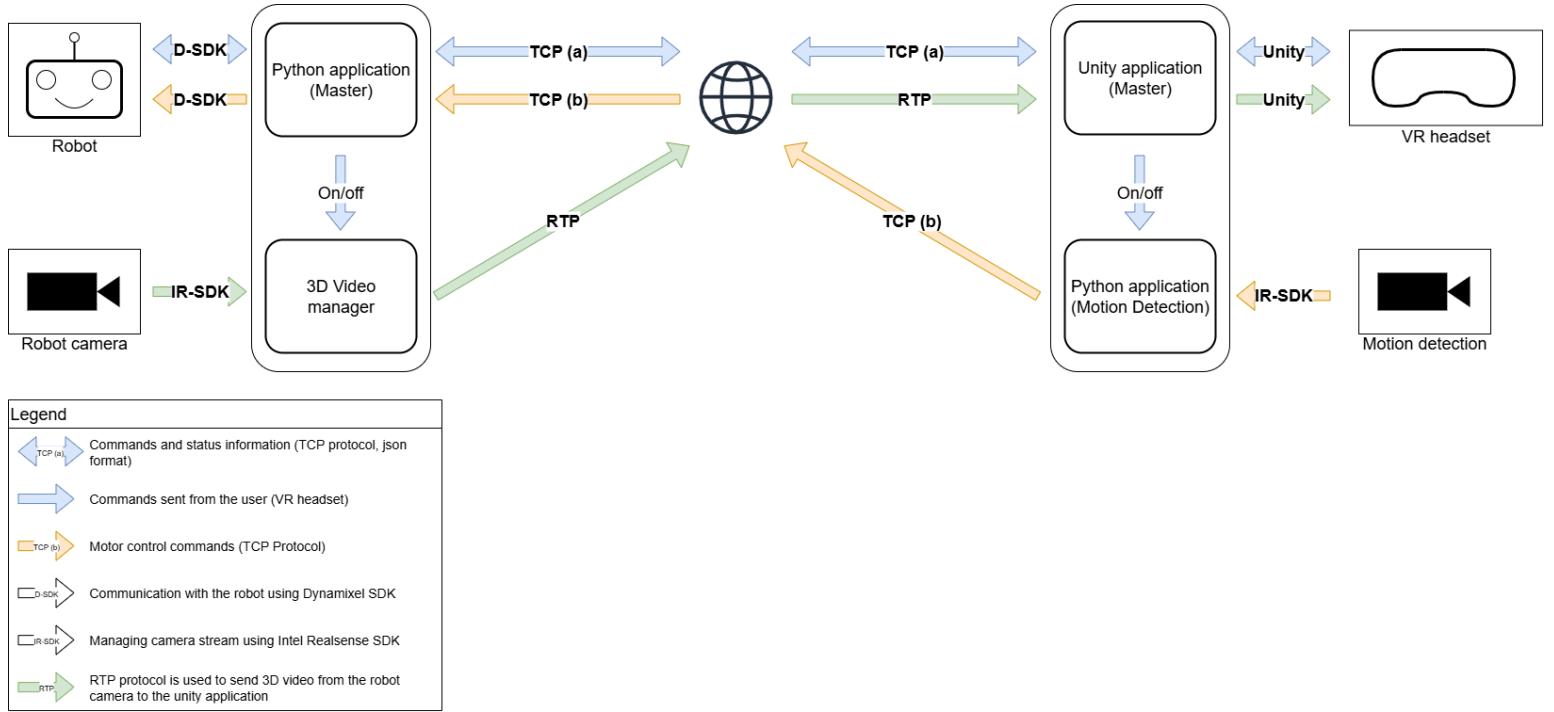
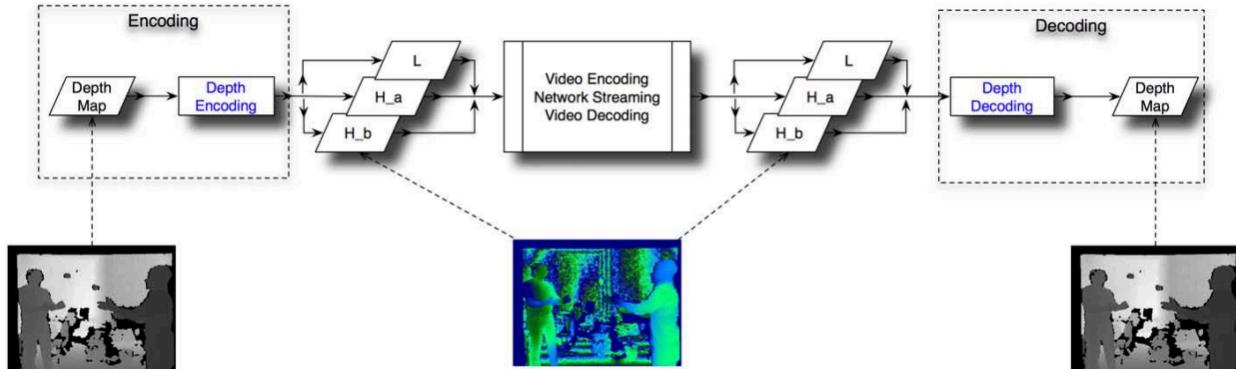


Schéma de l'architecture générale de la réalisation

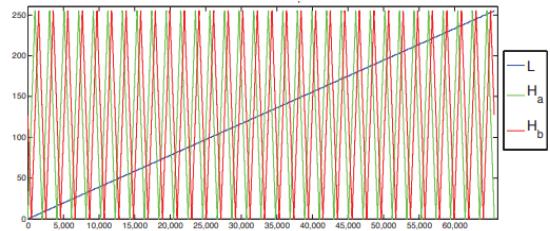
La solution adoptée comprend l'utilisation et l'implémentation en C++ de l'algorithme développé dans [8] combinée avec l'utilisation de la bibliothèque Gstreamer. Au-delà des performances compatibles avec le cahier des charges en raison de la simplicité de la solution, le choix de cette solution repose également sur les possibilités d'intégration. Notamment l'existence d'un plugin GStreamer pour Unity, permet une intégration dans l'application plus simple au vu du temps imparti pour réaliser le stage.

### Détail sur l'algorithme et implémentation de la transformation du format Z16 au format YUV



L'intérêt de cette transformation intra-image est qu'elle permet de retranscrire chaque profondeur ( $d$ ) du canal 16 bits en 3 canaux ( $L(d)$ ,  $H_a(d)$ ,  $H_b(d)$ ). Par la suite, ces canaux seront utilisés respectivement comme les canaux YUV dans la pipeline de compression vidéo (h264, h265).

- $L(d)$  : encode une approximation de la profondeur par une normalisation linéaire sur 8 bits.
- $H_a(d)$  et  $H_b(d)$  : affinent cette approximation en restituant les détails fins. Ils sont obtenus par des **fonctions triangulaires** de période  $np$ , déphasées de  $\pi/4$  (quadrature) puis normalisées. Leur continuité et leur oscillation rapide permettent de conserver une précision élevée après compression.

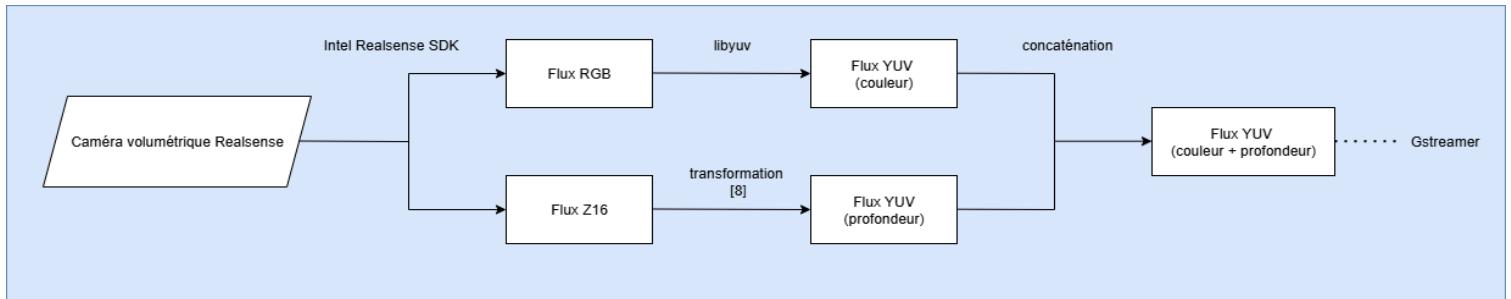


Le paramètre  **$np$**  joue un rôle central :

- Un  **$np$  élevé** (périodes plus nombreuses, donc période réduite plus courte) permet une meilleure précision de reconstruction, ce qui se traduit par une augmentation du PSNR.
- Un  **$np$  trop élevé** en revanche, rend les signaux  $H_a$  et  $H_b$  très oscillants, ce qui complique la tâche des encodeurs vidéo standards basés sur la prédiction et la compensation de mouvement. Cela peut dégrader l'efficacité de compression ou augmenter le débit nécessaire.

Une étude sur les performances de la compression après transformation a été développée lors de l'étude [8] et propose une valeur optimale de np.

Parallèlement, une amélioration a été apportée pour rendre l'application de l'algorithme plus rapide. Conseillée dans l'étude, il s'agit de tabuler les valeurs de L, Ha et Hb de l'algorithme. Cette solution a contribué à l'augmentation du débit.



### **Compression et Transport gstreamer.**

La compression et le transport du flux volumétrique ont été réalisés à l'aide de pipeline GStreamer, en exploitant l'accélération matérielle de la carte graphique pour l'encodage et décodage vidéo.

Après transformation du flux profondeur au format YUV et transformation du flux couleur au format YUV également, chaque image couleur est concaténée avec l'image profondeur puis injectée dans la pipeline via un élément source par l'application, appsink. L'encodage est effectué avec l'encodeur matériel H.264 (nvh264enc, encodeur avec accélération matérielle), configuré en mode faible latence (GOP [groupe d'images encodées] réduit, Bi-directional frame [codée en utilisant la précédente et la suivante comme référence] désactivés, débit constant). Les trames sont ensuite encapsulées dans le format RTP puis envoyées sur le réseau par UDP. Le fait de concaténer la profondeur

Cette architecture permet d'obtenir un flux en temps réel avec une latence minimale, tout en réduisant fortement le débit nécessaire par rapport au signal brut en Z16. Les paramètres de la pipeline ont été définis de manière empirique à la suite de plusieurs campagnes de tests

comparant différentes valeurs de bitrate, de GOP et de MTU, afin d'évaluer leur impact sur la qualité visuelle et le délai de transmission.

Il en ressort que **la réduction du GOP** (intervalle entre deux images clés) diminue l'efficacité de la compression mais rend le flux plus robuste face aux pertes de paquets, puisque les erreurs de transmission sont corrigées plus rapidement par l'insertion d'une nouvelle image de référence (I-frames). De la même manière, **la réduction du MTU** (taille maximale des paquets réseau) fragmente davantage le flux, mais permet de limiter les pertes liées à la corruption d'un paquet unique, ce qui améliore la résilience du système au prix d'une surcharge réseau légèrement plus élevée.

### Plugin Gstreamer Unity.

Pour la **réception** et l'intégration dans l'application de réalité virtuelle, un **plugin GStreamer pour Unity** a été utilisé (disponible en open-source sur GitHub [9]). Ce plugin permet de définir directement une pipeline de réception compatible avec Unity et de récupérer le flux décodé sous forme de textures exploitables par le moteur graphique. La pipeline de réception implémente une réception sous udp, un dépaquetage rtp, et un décodage h264. La DLL fournie initialement a dû être modifiée pour synchroniser, premièrement la réception des images en continu pour appliquer la transformation à chaque instant, puis deuxièmement à l'envoi des images RGB et Z16 au plugin Realsense développé ci-dessous. La gestion de la mémoire entre l'application Unity en C# et le DLL en C++ a nécessité une certaine attention.

Le partage d'images entre les deux parties du programme fonctionne de la manière suivante : d'un côté, un module écrit en C++ reçoit les images et les met en mémoire. Dès qu'une nouvelle image est prête, il envoie un signal pour prévenir la partie Unity. Unity peut alors copier ces images dans son propre espace mémoire afin de les afficher.

Pour éviter que les deux parties du programme n'accèdent à la même image exactement au même moment (ce qui pourrait créer des erreurs), un mécanisme de verrouillage est utilisé. Enfin, pour que C++ et Unity puissent bien communiquer entre eux, certaines règles de compatibilité sont respectées dans la façon dont les fonctions sont déclarées.

## Bibliothèque et plugin Realsense

Du côté robot, la bibliothèque C++ est utilisée pour se connecter à la caméra, récupérer le flux volumétrique et à chaque instant appliquer une filtration par seuil adaptée aux caractéristiques de la caméra afin d'enlever les valeurs aberrantes liées à une mauvaise réflexion de la lumière puis une filtration temporelle afin de rendre le flux d'entrée plus stable. Ces deux étapes de filtration permettent d'améliorer le rendement aussi bien de la transformation que de la compression.

Du côté casque et utilisateur, Une fois que le flux reçu est décompressé et transformé au format Z16 le plugin Realsense Unity permet d'intégrer dans la scène virtuelle un rendu de nuage de points dynamique ainsi que certaines options de filtrage (temporel, spatial et alignement). Sont utilisés les filtres temporels et d'alignement. Le filtre d'alignement permet de compenser le non alignement physique des capteurs couleur et profondeur sur la caméra avant l'affichage du nuage de points en continu.

## Validation

### Expérience de validation

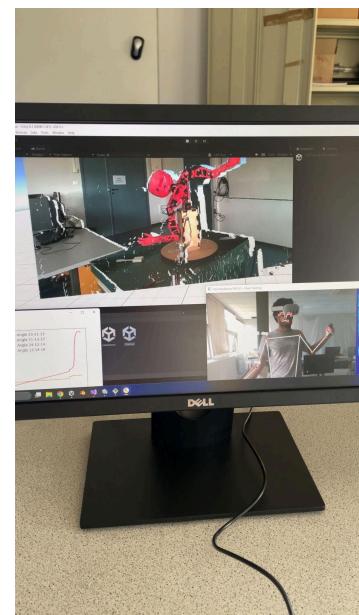
L'expérience de validation a été menée sur plusieurs personnes ne connaissant pas le stage et plusieurs remarques ont été retenues.

*Concernant les fonctionnalités utilisateur :*

Toutes les fonctionnalités ont pu être utilisées, un guide d'utilisation expliquant un protocole d'utilisation a été rajouté à l'intérieur du casque. On notera que les principales difficultés étaient liées à l'utilisation du casque et la prise en main des manettes et à l'adaptation à l'impression 3D.

*Estimation visuelle de la qualité visuelle :*

Le nuage de points donne une réelle impression de profondeur, on remarque que certains objets ne sont pas sur le même plan ce qui était le principal objectif.



L'utilisateur peut s'approcher, s'éloigner, changer d'axe d'observation par rapport au nuage de points dans une certaine mesure. Toutefois certaines limites apparaissent lorsque l'utilisateur s'approche "trop", le nuage de points paraît moins dense et lorsque ce dernier regarde le nuage de points par le côté et se situe à la perpendiculaire de l'axe d'observation de la caméra par rapport au nuage de points, on remarque que les objets situés en arrière-plan deviennent invisibles derrière ceux au premier plan. Ces limites ne sont pas un problème car elles sont hors périmètre du stage comme défini dans le cahier des charges.

### Mesures des performances système

**Le protocole de mesure** des performances système est pensé de la manière suivante : Un premier programme permet d'enregistrer en local le flux volumétrique de la caméra après filtration sur deux fichiers un contenant les données au format raw RGB et l'autre au format raw Z16 sur un ordinateur. Ces flux enregistrés constituent la base de données.

Trois enregistrements ont été réalisés avec une résolution de 1280x720 pixels à 30 images par seconde pour les deux formats. Un second programme permet de simuler la chaîne complète de transformation, compression et transport en émission et en réception. On lit les fichiers de références et on écrit de la même manière les vidéos reçues. Puis à l'aide d'un troisième programme, on vient mesurer le PSNR entre les images en profondeur Z16 avant et après envoi. Ne sont pas présentés les écarts entre les images couleur reçue et démontrées, car sinon cela reviendrait à prouver l'efficacité de l'implémentation de Gstreamer des algorithmes de compression vidéo en eux même. Cependant quelques mesures ont tout de même été effectuées, ces dernières montrent selon l'indicateur déjà implémenté, un PSNR aux alentours de 45 db ce qui est gage de qualité de la reconstruction de l'image.

### Mesures obtenues :

*Débit* :  $30 \pm 2$  fps à 1280x720 pixels pour le flux profondeur et pour le flux couleur.

*Latence* : 20 ms en moyenne (Mesurée depuis l'application Unity).

Débit binaire sans compression :  $1280 \times 720 \times 30 \times (3 \times 8 + 16)$  bits/s soit 1,1 Gbit/s

*Vidéo 1 (3 mins)* :



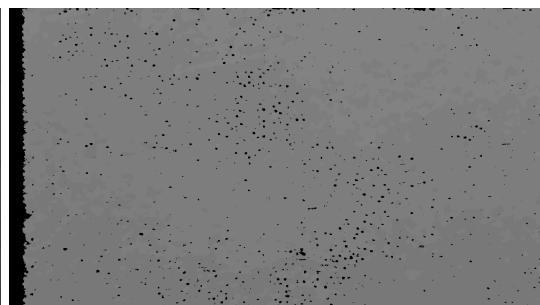
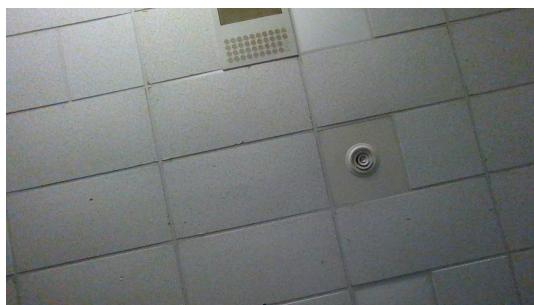
Cette scène a été choisie en raison de ses **textures complexes et détaillées**.

*Débit binaire moyen : 26.98 Mbps*

*Taux de compression : 41:1*

*PSNR profondeur : 40.21*

*Vidéo 2 (3 mins) :*



Cette scène comporte uniquement des **zones uniformes**

*Débit binaire moyen : 20.606 Mbps*

*Taux de compression : 53:1*

*PSNR profondeur : 44.40*

*Vidéo 3 (5 mins):*

**Scène dynamique**, où lors de la prise de la vidéo on fait bouger la caméra constamment.

*Débit binaire moyen : 29.42 Mbps*

*Taux de compression : 37:1*

*PSNR profondeur : 41.84*

*Vidéo 4 (3 mins):*



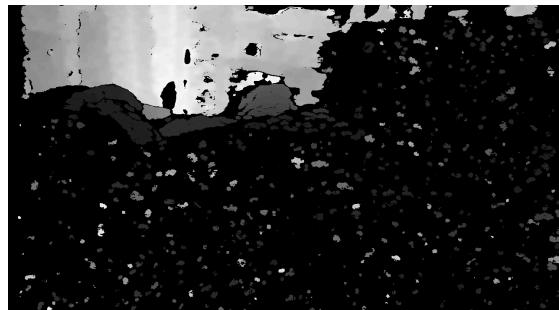
Scène comportant des **éléments trop loin** de la caméra pour que la profondeur soit détectée.

*Débit binaire moyen : 32.26 Mbps*

*Taux de compression : 34:1*

*PSNR profondeur : 38.40*

*Vidéo 5 (3 mins):*



Scène comportant des **éléments trop proches** de la caméra pour que la profondeur soit détectée.

*Débit binaire moyen : 21.25 Mbps*

*Taux de compression : 52:1*

*PSNR profondeur : 39.70*

### Retour sur expérience

Le projet mené avec mon camarade stagiaire a été une succession de phases de recherche et de tests pour résoudre premièrement mais pas des moindres, les nombreux problèmes techniques lors de tests de solution de compression, de plugin gstreamer sur Unity qui est classé comme “Bad Plugin” sur leur site car peu documenté, de lecture de log, mais également de tests d’optimisation de la pipeline gstreamer, de recherches de techniques de compression pour les médias immersifs. J’ai pu m’appuyer et développer certaines capacité en compression vidéo, toucher du doigt quelques techniques de vidéo 3D : nuages de points dynamiques, voxelisation,

système multivue, programmer des algorithmes de transformations d'images en C++ ainsi que certaines compétences en diffusion vidéo avec gstreamer. Ce stage fut le début de l'apprentissage du C++ pour ma part, language qui semble répandu et dont la dimension bas niveau est primordiale pour le développement de telles applications exigeantes en ressources.

## Conclusion

Les objectifs fixés initialement ont pu être atteints malgré le temps imparti relativement court par rapport à l'ampleur de l'application demandée. Des compromis ont dû être réalisés entre la qualité visuelle et le débit afin d'obtenir une latence compatible avec les exigences de téléopération immersive. La solution développée respecte pleinement le cahier des charges : le flux vidéo volumétrique est transmis avec une latence faible, permettant une interaction naturelle, et la qualité visuelle obtenue permet à l'utilisateur de percevoir correctement la profondeur et la disposition spatiale des objets dans l'environnement du robot. Le débit réduit grâce à la compression H.264 assure une transmission fluide adaptée au casque VR, et toutes les fonctionnalités prévues pour l'utilisateur, telles que la visualisation dynamique du nuage de points et le déplacement dans la scène 3D, ont été implémentées et validées dans l'environnement Unity.

Certaines limites subsistent, comme le champ de vision restreint ou l'absence d'exploitation complète de la redondance entre les flux couleur et profondeur lors de la compression, mais elles n'empêchent pas la satisfaction des critères essentiels définis dans le cahier des charges. Ces aspects pourraient être améliorés à l'avenir par l'ajout d'une seconde caméra ou par des optimisations logicielles.

Ce stage m'a permis de développer des compétences techniques et méthodologiques variées, allant de la programmation C++ et du traitement vidéo à l'intégration de flux volumétriques dans Unity, tout en consolidant ma capacité à planifier et suivre un projet complexe. Il m'a également offert une vision concrète des défis liés à la téléopération immersive et à la diffusion de flux volumétriques, ouvrant la voie à des perspectives sur réseaux 5G/6G et applications robotiques en conditions réelles.

## Bibliographie

- [1] ITU-T Recommendation G.114, "One-way transmission time," Int. Telecommunication Union, Online, Nov. 2003.
- [2] ISO/IEC JTC 1/SC 29/WG 7. 2020. V-PCC codec description. Technical Report N00100. MPEG. Online.
- [3] J. M. Boyce et al., "MPEG Immersive Video Coding Standard," in Proceedings of the IEEE, vol. 109, no. 9, pp. 1521-1536, Sept. 2021, doi: 10.1109/JPROC.2021.3062590.
- [4] Heikki Tampio, Joni Ra sanen, Marko Viitanen, Alexandre Mercat, Guillaume Gautier, and Jarno Vanne. 2024. uvgrTP 3.0: Towards V3C Volumetric Video Communication. In 15th ACM Multimedia Systems Conference (MMSys '24), April 15–18, 2024, Bari, Italy. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3625468.3652185>
- [5] Louis Fréneau, Guillaume Gautier, Alexandre Mercat, and Jarno Vanne. 2025. uvgVPCCenc: Practical Open-Source Encoder for Fast V-PCC Compression. In ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3712676.3718336>
- [6] ISO/IEC JTC1/SC29/WG11. 2020. Common test conditions for V3C and V-PCC. Technical Report N19518. MPEG. Online.
- [7] MPEG. 2024. Test Model Category 2 Version 24. Online. <https://github.com/MPEGGroup/mpeg-pcc-tmc2>.
- [8] Fabrizio Pece, Jan Kautz, and Tim Weyrich. 2011. *Adapting Standard Video Codecs for Depth Streaming*. In *Joint Virtual Reality Conference of EGVE - EuroVR (JVRC '11)*, 59–66. The Eurographics Association. <https://doi.org/10.2312/EGVE/JVRC11/059-066>
- [9] *mray/mrayGStreamerUnity: GStreamer Integration with Unity using a Native plugin*. GitHub. Mis à jour le 11 juin 2022. Licence MIT.
- [10] Google. libyuv: an open source library for YUV scaling and conversion. Online. Accessed: Sept. 24, 2025. Available: <https://chromium.googlesource.com/libyuv/libyuv/>

## Glossaire

AR, VR :

- AR (Augmented Reality / Réalité augmentée) : superposition d'éléments virtuels sur le monde réel.
- VR (Virtual Reality / Réalité virtuelle) : immersion complète dans un environnement numérique.

*RGB (RVB)* : Format de stockage des couleurs basé sur trois canaux (Rouge, Vert, Bleu), souvent sur 8 bits chacun (24 bits au total par pixel).

*YUV* : Format de représentation des couleurs séparant la luminance (Y) et les composantes de chrominance (U et V). Permet des compressions efficaces (ex. YUV420, YUV422).

*RGBD* : Format d'image stockant, en plus des trois canaux de couleur (R, G, B), une carte de profondeur (D) donnant la distance de chaque pixel par rapport à la caméra. Utilisé pour la vision par ordinateur, la reconstruction 3D et la réalité augmentée.

*SLAM (Simultaneous Localization and Mapping / Localisation et cartographie simultanées)* : Technique permettant à un système (robot, casque AR/VR, véhicule autonome) de se situer dans un environnement tout en construisant une carte de celui-ci.

*UVG (Ultra Video Group)* : Groupe de recherche spécialisé dans l'étude et la standardisation des technologies de compression et de codage vidéo avancées.

*MPEG (Moving Picture Experts Group)* : Groupe d'experts de normalisation qui définit des standards de compression audio et vidéo, comme MPEG-2, MPEG-4, HEVC, VVC, etc.

*V-PCC (Video-based Point Cloud Compression)* : Norme MPEG pour compresser des nuages de points 3D en les projetant sur des images 2D qui sont ensuite compressées avec des codecs vidéo standards.

*MIV (MPEG Immersive Video)* : Norme MPEG pour coder et transmettre des scènes immersives composées de multiples vues vidéo, adaptées aux applications AR/VR.

*PSNR (Peak Signal-to-Noise Ratio / Rapport signal sur bruit de crête)* : Indicateur de qualité de reconstruction d'une image compressée par rapport à l'originale. Plus le PSNR est élevé, meilleure est la qualité perçue.

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX^2}{MSE} \right)$$

Sont ainsi notés :

- (n, m) : la taille de l'image.
- MAX : valeur maximale possible d'un pixel.
- I(i,j) : pixel de l'image originale.
- K(i,j) : pixel de l'image compressée/reconstruite.
- MSE : erreur quadratique moyenne entre l'image originale et l'image reconstruite :

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I(i, j) - K(i, j))^2$$

*UDP (User Datagram Protocol)* : protocole de transport non connecté, ce qui signifie qu'il ne gère ni l'établissement ni la fermeture d'une connexion, et qu'aucune garantie n'est donnée quant à la livraison ou au maintien du flux. Les paquets ne sont transmis que lorsqu'ils sont explicitement envoyés, et ceux qui sont perdus ne sont pas retransmis. En revanche, UDP se révèle particulièrement adapté aux applications nécessitant une faible latence et prend en charge le *broadcasting* ainsi que le *multicasting*.

*RTP (Real-time Transport Protocol)* : construit au-dessus d'UDP, il ajoute des mécanismes supplémentaires tels que la gestion et le contrôle du flux. Conçu pour la transmission en temps réel (audio, vidéo), il tolère la perte de paquets afin de préserver la fluidité de la communication.

*RTCP (RTP Control Protocol)* : protocole compagnon de RTP qui permet de **surveiller la qualité de transmission**, fournir des statistiques sur les paquets envoyés/reçus, la gigue et la perte de paquets, et synchroniser différents flux multimédias (audio/vidéo).

*H264* : standard de compression vidéo très répandu, utilisant des techniques comme les I-frames, P-frames et B-frames pour réduire le débit tout en maintenant une qualité visuelle élevée.

*GOP (Group of Pictures)* : séquence de frames dans une vidéo H264, commençant par une I-frame et suivie de P-frames et B-frames. La taille du GOP (nombre de frames) influence la compression, la qualité et la latence.

I-frame : image clé encodée indépendamment, sans référence à d'autres frames.

P-frame : image prédite à partir de la frame précédente, avec seulement les différences encodées.

B-frame : image encodée en utilisant la frame précédente et suivante comme référence pour maximiser la compression.

*MTU (Maximum Transmission Unit)* : taille maximale d'un paquet réseau pouvant être transmis sans fragmentation. En streaming vidéo, il est important pour optimiser le transport UDP/RTP et éviter la fragmentation excessive.

## Annexe

### Gestion de projet

#### Planification et organisation du projet

Le projet s'est déroulé en 4 phases : étude de l'existant, choix techniques, implémentation, tests puis documentation.

**L'étude de l'existant** a permis de réaliser un état de l'art des solutions.

*1 Jalon de 2 semaines* : Recherche de projet open source de transfert vidéo volumétrique en temps réel, recherches des formats de visualisation des données 3D en continu, établissement d'une liste des normes et algorithmes de compression compatibles avec le cahier des charges.

**La phase de choix techniques** a permis de comparer et décider laquelle des solutions serait la solution admise, cette période a duré 4 semaines :

*1er puis 2nd Jalon (1 + ½ semaines)*: Tests des programmes des deux encodeurs références MPEG pour la compression au regard des besoins fonctionnels et des contraintes définies dans le cahier des charges.

*3ème Jalon (1 + ½ semaines)* : Test du programme d'encodage VPCC de uvg et de la bibliothèque de transport uvgRTP au regard des besoins fonctionnels et des contraintes définies dans le cahier des charges.

*4eme Jalon (1 semaine) :*

**La phase d'implémentation** 4 semaines :

*1er Jalon* : Adaptation du plugin gstreamer Unity pour que celui-ci convienne au cahier des charges.

*2nd Jalon* : Implémentation de la transformation Z16 vers YUV de manière paramétrable et avec version où les calculs sont tabulés.

*3ème Jalon* : Intégration de la bibliothèque Libyuv et implémentation de l'émetteur pour l'expérience de validation.

*4ème Jalon* : Intégration dans le système complet et temps pour des ajustements.

**La phase de test** a duré 2 semaines :

*1er Jalon* : Implémentation d'un ensemble de programmes tests qui comprend toute la chaîne de compression et transport et la mesure utilisée pour comparer les tableaux de profondeur (Z16).

*2nd Jalon* : Comparaison de plusieurs configurations de pipelines gstreamer et paramètres de transformation.

#### Communication et coordination

La communication avec le maître de stage s'est réalisée principalement lors de réunions hebdomadaires et parfois par mail. Lors de la phase de recherche, j'ai dû présenter les différentes solutions trouvées puis lors des phases de test expliquer le raisonnement d'admissibilité de telle ou telle solution.

Mon collègue et moi étions dans la même salle de projet, ainsi la communication entre nous s'est faite principalement à l'oral. Pour partager des liens ou détails techniques nous avons utilisé les applications en ligne (notion et draw) qui nous ont permis d'écrire ce qui a été réalisé pour chaque jalon. L'outil de versionnement de code Git et la plateforme Github ont dû être utilisés pour partager et organiser le code développé au cours des jalons.

Un planning en ligne a également été mis en place de manière à pouvoir suivre ce que l'autre faisait en détail

### Gestion des ressources

La gestion des ressources physique et les problèmes de compatibilité ont joué un rôle important lors du choix des solutions implémentées et lors de la réalisation de l'expérience test.

### Gestion des risques et problèmes rencontrés

#### Risques identifiés :

- Débit trop élevé pour le réseau,
- Latence trop importante la téléopération,
- Manque de maturité des standards MPEG.

#### Solutions mises en place :

- Compression avec transformation Z16 → YUV,
- Utilisation de gstreamer plutôt qu'un codec expérimental,
- Tests en local avant de viser la 5G ou un déploiement plus ambitieux.
- Gestion du temps perdu à cause de problèmes techniques (bugs Unity, mise à jour inopinée du casque Meta, installation et tests d'intégration de bibliothèque peu documentée).

### Suivi, validation et livrables

Le suivi du stage a été assuré principalement par le maître de stage au travers de **réunions hebdomadaires** et de points ponctuels avec mon collègue stagiaire. Ces réunions ont permis de faire le bilan des jalons atteints, d'identifier les difficultés techniques et de définir les étapes suivantes. La validation des solutions développées s'est appuyée sur des **tests objectifs et quantitatifs**, mesurant la latence, le débit, le PSNR des flux vidéo et la stabilité du système face aux pertes de paquets.

Les livrables fournis à l'issue du stage comprennent : les **codes sources commentés** (émetteur et récepteur), la **pipeline GStreamer complète**, l'**intégration Unity** pour la visualisation en réalité virtuelle, des **scripts de tests et de mesure de performance**, un **guide d'installation et**

**d'utilisation** pour les programmes et les bibliothèques tierces, ainsi qu'un **guide de développement** permettant à d'autres étudiants de poursuivre et d'améliorer le projet. Ces livrables ont été conçus pour être exploitables immédiatement et servir de base pour des développements futurs, notamment sur des réseaux plus larges ou avec un déploiement robotique réel.

#### Retour personnel sur la gestion de projet

Le stage m'a permis de découvrir les contraintes liées à la planification et au suivi d'un projet complexe. Le temps alloué aux phases de test a été **insuffisant**, ce qui a limité la possibilité de tester exhaustivement toutes les configurations et d'optimiser certains paramètres de la pipeline. Néanmoins, l'organisation par **jalons successifs** a facilité la gestion des tâches, la communication avec le maître de stage et la coordination avec le stagiaire collaborateur.

J'ai appris à **prioriser les développements critiques**, à documenter les choix techniques et à m'adapter rapidement face aux imprévus, comme les problèmes liés à l'intégration du plugin GStreamer dans Unity ou aux mises à jour du casque VR. Cette expérience m'a également permis de développer une meilleure **vision globale de la gestion de projet**, en équilibrant objectifs techniques, contraintes matérielles et limites temporelles, tout en gardant la progression vers les objectifs principaux du cahier des charges.