Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement

2021-2022

Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV

Entreprise
Polytech

Tuteur entreprise
Mathieu DELALANDRE

Étudiant
Hugo PERVEYRIE (DI5)

Tuteur académique
Mathieu DELALANDRE

4 avril 2023

Liste des intervenants

Entreprise

Polytech
64 avenue Jean Portalis
37200 Tours, France
polytech.univ-tours.fr

Nom	Email	Qualité
Hugo PERVEYRIE	hugo.perveyrie@etu.univ-tours.fr	Étudiant DI5
Mathieu DELALANDRE	mathieu.delalandre@univ-tours.fr	Tuteur académique, Département Informatique
Mathieu DELALANDRE	mathieu.delalandre@univ-tours.fr	Tuteur entreprise

Avertissement

Ce document a été rédigé par Hugo PERVEYRIE susnommé l'auteur.

L'entreprise Polytech est représentée par Mathieu DELALANDRE susnommé le tuteur entreprise.

L'Ecole Polytechnique de l'Université de Tours est représentée par Mathieu DELALANDRE susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assume l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Hugo PERVEYRIE, *Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV*, Projet Recherche & Développement, Ecole Polytechnique de l'Université de Tours, Tours, France, 2021-2022.

```
@mastersthesis{
    author={PERVEYRIE, Hugo},
    title={Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université de Tours},
    address={Tours, France},
    year={2021-2022}
}
```

Document mis en forme sous LaTeX
grâce à la classe de document `polytech.cls` V2.5.2(2019/08/07)

Table des matières

Table des figures

1 Introduction

1 Acteurs, enjeux et contexte

Le Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT) est composé de 48 membres de la faculté (Professeurs et Professeurs Assistant), 25 doctorants et sept docteurs.

Les préoccupations scientifiques du LIFAT sont nombreuses. Il souhaite concevoir et développer des modèles, méthodes et algorithmes ainsi que fournir des ressources et des logiciels pour extraire des informations et récupérer des connaissances à partir de données en intégrant l'interaction homme-machine. Le LIFAT résout également des problèmes d'optimisation combinatoire avec la volonté d'obtenir de bons résultats en un bon temps de calcul.

Le laboratoire est actuellement organisé en 3 groupes de recherches :

- Base de données et Traitement des langues naturelles (BdTln)
- Recherche opérationnelle, Ordonnancement et Transport (ROOT)
- Reconnaissance des Formes et Analyse d'Images (RFAI)

Le projet RFAI6 est un Projet de Recherche et de Dévelopement (PRD) réalisé par Hugo PERVEYRIE, étudiant en 3ème année de cycle ingénieur en Informatique. Ce projet est encadré par Mathieu DELALANDRE, chercheur au LIFAT et enseignant à Polytech Tours. Comme son nom l'indique, ce projet est proposé par l'équipe RFAI du LIFAT.

Le projet se déroule sur une partie de la Station TV, un projet du LIFAT qui fait l'objet de plusieurs stages étudiants ayant pour but le déploiement d'applications de traitement d'images et d'intelligence artificielle.

2 Objectifs

La station TV a un programme qui permet de capturer 24 flux TV simultanément à un taux maximum de 20 images par seconde. Ce programme utilise le SDK de AVerMedia pour récupérer les images grâce aux cartes d'acquisition AVerMedia.

Ces images sont enregistrées sur les disques dur de la station, ce qui ne permet pas un fonctionnement du programme sur le long terme. On rencontre rapidement un problème de stockage en raison de la volumétrie demandé par l'écriture de 480 images par seconde (20×24).

Les images sont récupérées et enregistrées en JPEG avec les paramètres de compression totalement gérés par AVerMedia sans possibilité de changer ce taux de compression. Les images étant utilisés à des fins d'entrainement pour une intelligence artificielle, un contrôle de la qualité de l'enregistrement est indispensable.

Le personnel du LIFAT souhaitant pouvoir faire tourner le programme de capture sur plusieurs jours, il faudrait pouvoir filtrer les images capturées avant de les enregistrer afin de limiter l'espace disque utilisé. Ces images étant utilisées à des fins d'entrainement d'intelligence artificielle, on souhaite aussi pouvoir gérer l'enregistrement des images afin de changer le paramètre de compression d'images JPEG et la résolution de l'image.

Une amélioration de ce programme nécessite donc :

— L'utilisation d'une librairie d'analyse d'images compatible avec AVerMedia
— Une définition de critères de qualité pour les images
— Un filtrage des images récupérés en fonction de plusieurs critères de qualités prédéfinies
— Une gestion de la sauvegarde d'image

Pour résumer, l'objectif principal du projet est donc d'utiliser une solution d'analyse d'image couplée avec le kit de développement AVerMedia afin de pouvoir réduire la capacité disque utilisé par le programme de base. On souhaite aussi pouvoir manipuler la qualité et le taux de compression des images JPEG capturées.

3 Bases méthodologiques

On utilisera des diagrammes de GANTT pour le suivi du projet, notamment avec l'utilisation de l'outil GanttProject.

Les spécifications du projet étant clairement définies, j'ai choisi d'utiliser un cycle en V itératif en méthodologie de projet. Cela implique une réunion de pilotage entre chaque accomplissement de grosses parties du projet.

Un suivi des tâches est nécessaire et est effectué avec l'aide de la méthode Kanban qui nécessite l'utilisation d'un gestionnaire de tâches comme Trello.

Concernant le programme à améliorer, nous sommes donc contraints d'utiliser C++ car la structure de base utilisant le SDK de AVerMedia a été développée en C++.

2 Description générale

1 Environnement du projet

1.1 Station TV

L'environnement du projet est la station TV. Elle est composée de deux stations de travail bien différentes.

On retrouve donc :

- La station de travail Dell Précision T7600
- La station de travail Dell PowerEdge T640
- Une baie de 8 tuners attribués à la première station de travail
- Une baie de 24 tuners attribués à la seconde station de travail

Les baies de tuners récupèrent les différentes chaînes de la TNT grâce à une antenne présente sur le site de Polytech.

La différence entre les deux stations TV est assez simple. La station de capture vidéo va récupérer des fichiers audio/vidéo des 8 chaînes qui seront traitées plus tard par d'autres applications et la station de traitement en temps réel va récupérer les 24 chaînes et générer un nombre d'images choisies pour ces 24 chaînes. Cette génération d'images est beaucoup plus couteuse en ressources qu'une simple capture vidéo et la station à donc plus de capacité de traitement.

Pendant ce projet, j'ai été amené à améliorer le programme de capture présent sur la station de travail Dell PowerEdge T640. Cette station sera donc décrite plus en détail.

Vous retrouverez ci-dessous un schéma afin de mieux comprendre le cheminement de l'information entre la réception du signal TNT et la réception puis traitement des données par les 2 stations TV :

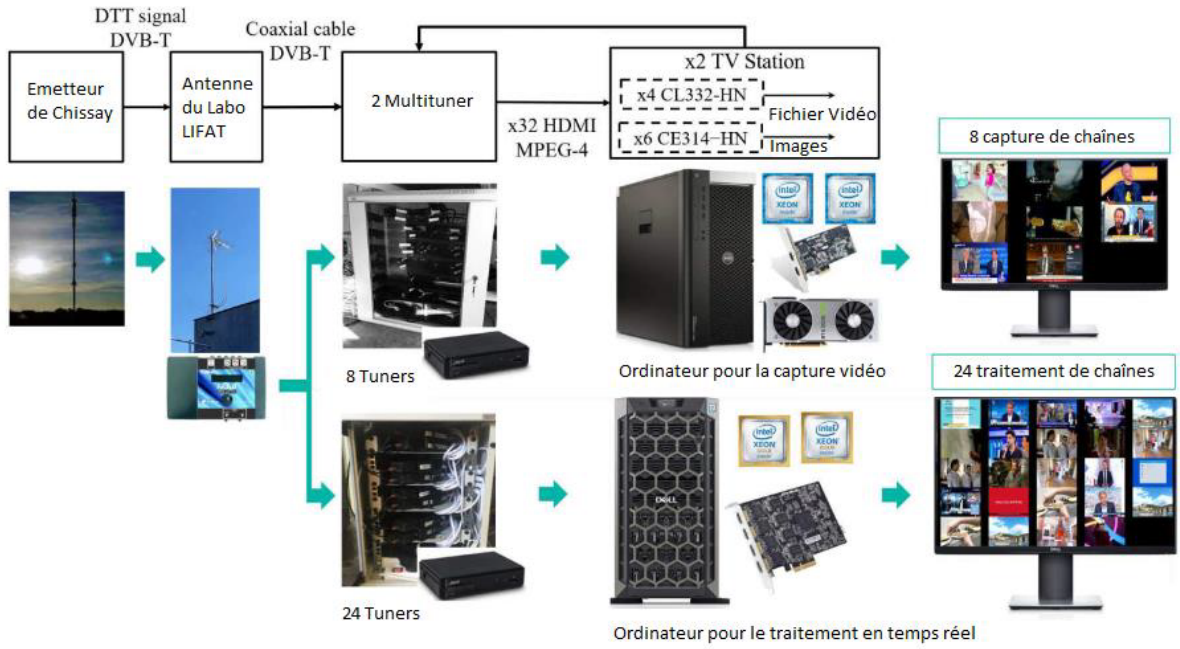


Figure 2.1 – *La station TV*

1.2 Dell PowerEdge T640

1.2.1 Composants

La station de travail DELL PowerEdge T640 est composée de 2 processeurs Intel Xeon Gold 5218R (2×20 cœurs cadencés à 2,10 GHz) et de 32Go de RAM. Elle possède cinq cartes d'acquisition AVerMedia CE314-HN et une carte d'acquisition AVerMedia plus récente. L'architecture disque est assez complexe, elle sera décrite dans la section suivante.

1.2.2 Architecture disque

La station de travail Dell PowerEdge T640 est composée initialement de deux disques dur Dell 1,2 To cadencées à 10 000 tours par minute de 3.5 pouces en interface SAS.

La station de travail T640 est équipée d'un contrôleur de réseau de disque Dell PERC H330. Un contrôleur de réseau de disques est un périphérique qui gère les disques physiques et les présente à l'ordinateur sous la forme d'unité logique. Il implémente presque toujours une interface RAID ce qui le rend plus généralement connu sous le nom de contrôleur RAID.

Un projet Architecture des Systèmes et Réseaux (ASR) est aussi actuellement en cours sur la station de travail pour ajouter 2 disques de 12To. Ces deux disques vont être implémentés avant la phase de développement du PRD et notre programme de capture travaillera donc sur ces deux disques. Le projet ASR s'occupera intégralement de cette partie-là.

1.2.3 Carte de capture PCIe Full HD HDMI 4 canaux

Le modèle des cartes de capture est AVerMedia CE314-HN. Ce sont des cartes de capture PCIe Gen1x4 (1 Go/s) équipées de 4 ports HDMI avec une entrée audio intégrée. Avec ses entrées de

source vidéo HDMI, chaque carte est capable de prendre en charge une capture de vidéo en temps réel allant jusqu'à la résolution Full HD en 60 images par seconde (Image par Seconde (FPS)). Lorsqu'on utilise les 4 ports, on peut capturer en Full HD en 30 FPS. Les images capturées sont nativement en 8 bit par pixel et en format de pixels YUV. Ce modèle supporte le multicartes et nous permet donc d'en embarquer 4 dans la station de travail afin de capturer 4×6 flux soit 24 flux.

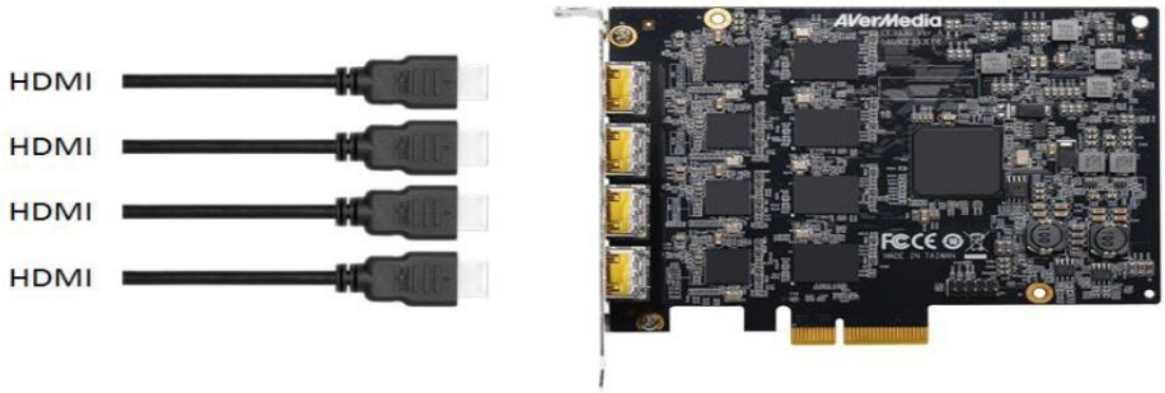


Figure 2.2 – *Carte AVerMedia CE314-HN*

1.3 Dell Precision T7600

1.3.1 Composants

La station de travail Dell Précision T7600 est, elle aussi, en architecture biprocesseur mais possède quant à elle 2 processeurs Intel Xeon E5-2620 (2 x 8 cœurs cadencé à 2,10 GHz). Cette machine est utilisée pour la capture de vidéo, elle possède 4 cartes de capture vidéo AVerMedia CL332-HN. Ces cartes sont moins performantes puisqu'elles peuvent seulement enregistrer des vidéos et ceux sur 2 ports uniquement. Pour des besoins de traitement vidéo, la station est équipée d'une carte graphique GeForce RTX2070 Super.

2 Caractéristiques des utilisateurs

Les utilisateurs du programme de capture de flux sont des chercheurs du LIFAT. Ce sont des utilisateurs qui sont familiers avec le milieu de l'informatique et qui savent utiliser une console.

Il ne sera donc pas nécessaire d'implémenter une interface graphique. Il sera toutefois indispensable de documenter le code afin que des futurs informaticiens puissent comprendre le code.

3 Fonctionnalités du système

Le système en lui-même resterait celui du programme de capture avec quelques changements au niveau des paramètres d'entrée et un nombre d'images en paramètres de sortie qui serait allégé.

Le système de capture d'image filtrerait chaque image, après réception, basé sur le contraste, l'entropie et la différence par rapport à la dernière image capturée sur son flux. Cela résulterait en un bon allègement de la capacité disque utilisée. Le système étant totalement automatisé après définition des paramètres. Les utilisateurs n'interagissent presque pas avec celui-ci.

La structure interne du programme est précisément décrite ci-dessous.

4 Structure générale du système

Comme expliqué lors des chapitres précédents, un système existe déjà et nous souhaitons l'améliorer.

Le système est composé d'un programme présent sur la station de travail Dell PowerEdge T640 qui capture les images sur les différents flux avec comme paramètres :

— Le nombre de flux à capturer (24)
— Le nombre d'images par seconde à capturer par flux (20)
— Le format des images à enregistrer (JPEG)

On souhaite ajouter à ce système des filtres lors de l'enregistrement de nos images et une gestion du taux de compression de l'image JPEG.

De nouveau paramètres à ajouter serait donc le taux de compression JPEG et la résolution d'enregistrement.

Un PRD a déjà été réalisé par Théo REMON, lequel avait mis en place des filtres spatiaux sur les images. Ces filtres permettent de réduire davantage le nombre d'images capturées (plus de 60%). Ils utilisent l'espace spatial de l'image pour fonctionner. Cela posera toutefois un problème dans notre cas dès lors que l'on souhaite filtrer 480 images par seconde (ce nombre correspond au nombre de flux capturés multipliés par le nombre d'images par seconde). On cherche donc à filtrer nos images grâce à un filtre bas niveau. Un calcul optimisé de l'histogramme d'une image pourrait nous permettre d'appliquer quelques filtres bas niveau. Cela nous permettrait également de filtrer les images grâce au contraste et à l'entropie ainsi que d'appliquer un algorithme de duplicata.

Le système, après toutes ces améliorations, capture donc les images et procède à plusieurs filtrages. Après récupération des histogrammes des images, les filtres effectuent des calculs pour récupérer des informations comme le contraste. Si l'image n'est pas retenue après avoir été filtrée, celle-ci ne sera pas enregistrée. Le cas échéant, il faudrait enregistrer cette image avec la compression et la résolution passées en paramètre.

Voici finalement un schéma expliquant le fonctionnement du système :

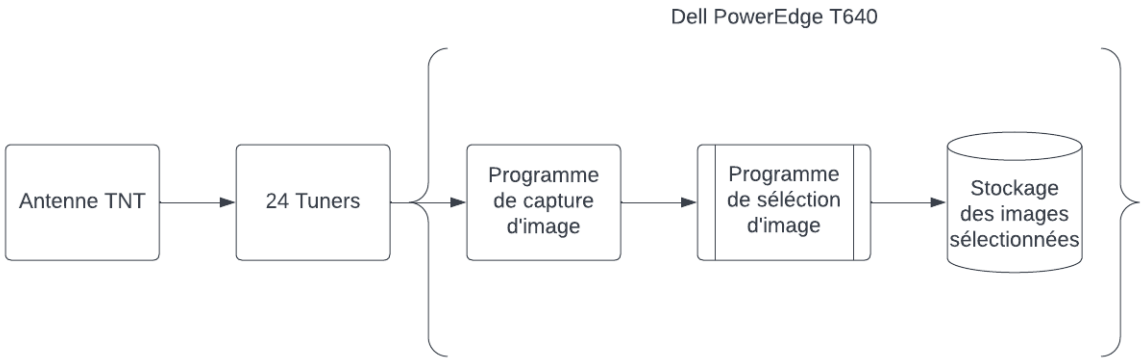


Figure 2.3 – *Diagramme du système de captures d'image*

Le système actuel est similaire, mais ne comporte donc pas de programme de sélection d'image.

3 Veille technologique

1 Création d'un répertoire d'images

Afin d'effectuer les différents tests, nous avons besoin de données correspondant à notre programme de capture d'images. Nous savons que ce programme récupère un flux TV en direct (vidéo) dans l'espace de couleur YUV. Pour simuler cette arrivée d'images, j'ai récupéré un extrait du journal de 20 h sur France 3.

Cet extrait est par la suite réécrit grâce à OpenCV en plusieurs images au format JPEG dans l'espace de couleur YUV. Pour ce faire, j'ai effectué un programme de conversion vidéo RGB vers des images YUV que vous pouvez retrouver ci-dessous :

```
1  VideoCapture cap("C:\\tv.mp4", CAP_MSMF);
2  int i = 0;
3  while (1) {
4          Mat frame;
5          cap >> frame;
6          cvtColor(frame, frame, COLOR_BGR2YUV);
7          if (frame.empty())
8                  break;
9          std::string path = "D:\\Cours\\Projet_R&D\\Sources\\
10         Images\\" + std::to_string(i) + ".png";
11         imwrite(path, frame);
12         i++;
13         waitKey(1);
14 }
15 cap.release();
```

Ce programme m'a donc permis de récupérer plusieurs milliers d'images dans l'espace de couleur YUV. Ci-dessous, vous trouverez un exemple d'une image dans l'espace de couleur YUV.

Figure 3.1 – *Extrait du JT en YUV*

Grâce à ces différentes images, il est maintenant possible de simuler l'arrivée d'un flux TV tel qu'il existe sur la station. Cela va nous permettre de tester notre récupération d'histogramme.

2 Récupération de l'histogramme et mesure du temps

On souhaite récupérer l'histogramme en niveau de gris à partir de matrice des images. La fonction calcHist de OpenCV correspond parfaitement à ce que nous souhaitons faire. Mais il faut néanmoins vérifier sa durée d'exécution car cette fonction devra être exécutée 20 fois par seconde par thread.

Voici une courte version de la récupération d'histogramme avec les informations essentielles :

```
1  Mat image = imread("D:\\Cours\\Projet_R&D\\Sources\\Images\\2025.jpg");
2  Mat hist;
3  cvtColor(image, image, COLOR_YUV2BGR);
4  cvtColor(image, image, COLOR_BGR2GRAY);
5  calcHist( &image, 1, 0, Mat(), hist, 1, &histSize,
6  &histRange, uniform, accumulate );
7
8  Mat histN;
9  hist.copyTo(histN);
10 for (int i = 0; i < histSize; i++) {
11                 histN.at<float>(i) = hist.at<float>(i) /
12   (image.size().width * image.size().height);
13 }
```

Notre histogramme récupéré prend en abscisse la plage de valeurs disponibles en gris et en ordonnée le nombre de pixels qui correspond à une valeur X. Voici un exemple d'histogramme associé à une image du JT en niveau de gris :

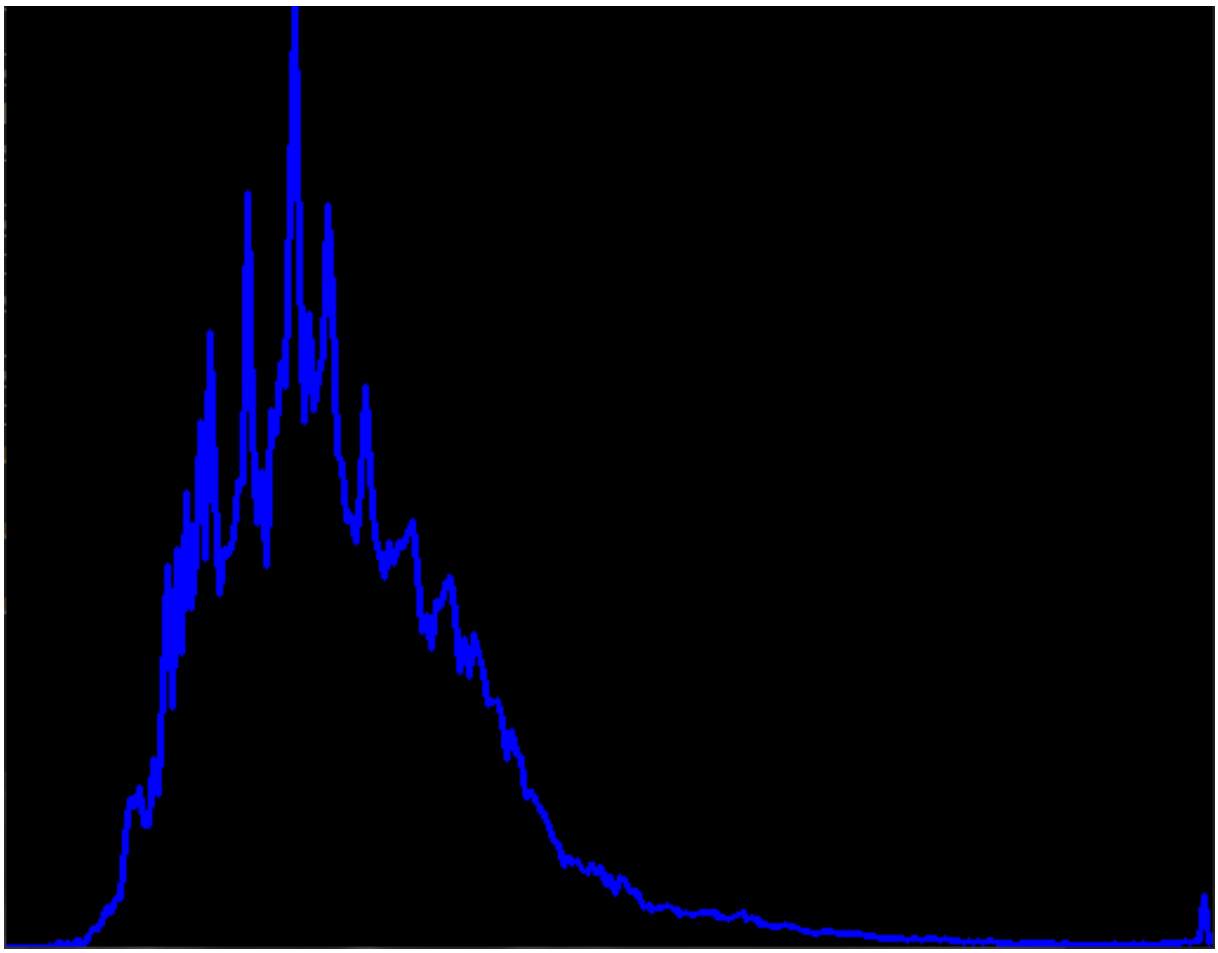


Figure 3.2 – *Histogramme associé à l'image*

Une remarque que l'on pourrait faire sur cette image est qu'elle est plutôt sombre, car les pixels ont tendance à se regrouper vers le début de la plage de valeurs.

En plus de l'histogramme récupéré, il y a aussi le calcul de l'histogramme normalisé. Vous pouvez retrouver ci-dessous un schéma concernant l'histogramme normalisé :

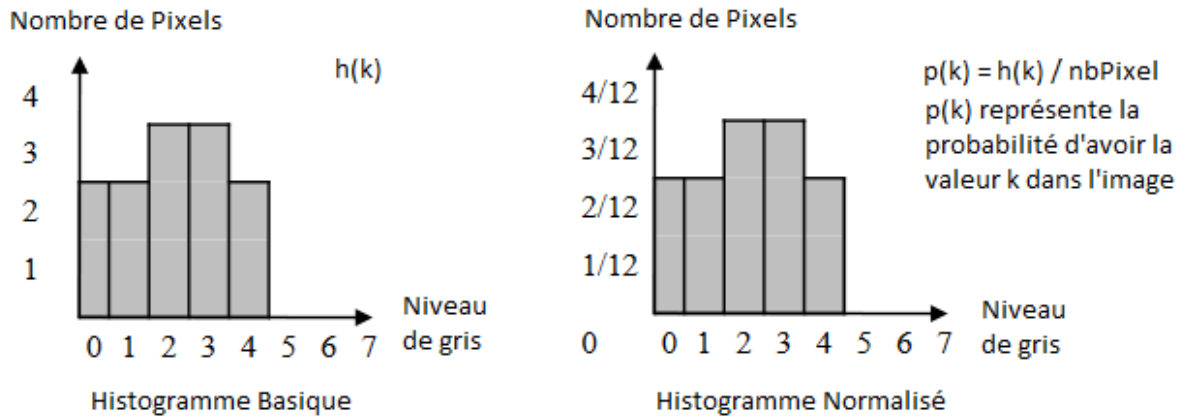


Figure 3.3 – *Définition de l'histogramme normalisé*

Comme décrit précédemment, il est important de connaître la durée d'exécution de la fonction de calcul d'histogramme. En effet, on souhaite être en mesure de réaliser une récupération d'histogramme autant de fois que l'on souhaite étudier les images. On souhaite donc récupérer des

histogrammes 20 fois par seconde et par flux. Les flux étant indépendants, il n'est pas important de préciser que cette fonction sera exécutée 480 fois par seconde.

Avec l'aide de la bibliothèque Chrono sur C++, j'ai pu récupérer le temps d'exécution du programme de récupération d'histogramme qui est d'environ 1500 microsecondes. Si on effectue ce programme 20 fois, on a donc une durée d'exécution de 30 millisecondes.

Cette fonction est donc totalement utilisable dans le cadre de nos analyses en temps réel. Les filtres étant calculés à partir de l'histogramme, leur durée d'exécutions est totalement négligeable.

La dernière donnée utile est la durée d'un enregistrement d'image avec OpenCV. Celui-ci est de 65 millisecondes. Sachant que dans un cas extrême, cinq images seront capturées sur une seconde, cette durée ne posera pas de problème particulier.

3 Filtre sur les histogrammes

Les filtres vont s'appliquer directement sur les histogrammes afin d'éviter des calculs sur le domaine spatial de l'image qui sont beaucoup plus longs.

Il existe des formules pour calculer le contraste et l'entropie à partir des données de l'histogramme. Par exemple, pour récupérer le contraste d'une image, on va calculer l'écart type des données de l'histogramme. Pour l'algorithme de duplication, un calcul de la distance entre deux histogrammes sera effectué afin de noter la possible duplication d'image.

L'objectif pour la suite est de tester les différents filtres puis de définir des seuils afin d'éliminer 95% des images. L'algorithme de duplication doit en éliminer une grande partie. L'algorithme de contraste et d'entropie nous garantissent qu'elles seront par ailleurs utilisables. Les seuils ne devraient pas créer beaucoup de faux négatif.

4 Analyse et conception

1 Les filtres

1.1 Filtre sur le contraste

On dispose d'une formule permettant de calculer le contraste à partir d'un histogramme normalisé. Cette formule a besoin de l'écart type et de la moyenne. Voici les différentes formules utilisées pour calculer le contraste :

$$\text{Moyenne} = \sum_{k=0}^{L}(k * p(k))$$

Ici L correspond à la taille de l'histogramme normalisé et p(k) correspond à la valeur k de l'histogramme normalisé

$$\text{Écart-type} = \sqrt{\sum_{k=0}^{L}(k-m)^2 * p(k)}$$

Ici m correspond à la moyenne

$$\text{Contraste} = 1 - \frac{1}{1+\frac{\sigma^2}{L^2}}$$

Toutes ces formules sont utilisées dans le calcul de contraste de la fonction ci-dessous :

```
1  // Calcul de la moyenne
2  double mean = 0.0;
3  for (int i = 0; i < histSize; i++) {
4          mean += i * histN.at<float>(i);
5  }
6
7  // Calcul de l'ecart type
8  double sD = 0.0;
9  for (int i = 0; i < histSize; i++) {
10         sD += pow(i - mean, 2) * histN.at<float>(i);
11 }
12 sD = sqrt(sD);
```

```
13
14 // Calcul du contraste
15 double contrast = 0.0;
16 contrast = 1 - (1 / (1 + (pow(sD, 2) / pow(histSize, 2))));
```

1.2 Filtre sur l'entropie

On dispose d'une formule permettant de calculer l'entropie à partir d'un histogramme normalisé. Vous pouvez retrouver cette formule ci-dessous :

$$\text{Entropie} = -\sum_{k=0}^{L}(p(k) * \log_b(p(k)))$$

Cette formule est utilisée dans le calcul de l'entropie de la fonction ci-dessous :

```
1 double entropy = 0.0;
2 for (int i = 0; i < 5; i++) {
3         if (histN.at<float>(i) != 0) {
4                 entropy += -1 * (histN.at<float>(i) * log2(histN.at<float>(i)));
5         }
6 }
```

1.3 Filtre de duplicata

Pour le filtre de duplicata, on souhaite être en mesure de calculer la distance entre 2 histogrammes. Ces 2 histogrammes correspondent à celui de l'image actuelle et celui de l'image précédente. La distance de Minkowski est une formule de distance précise et simple. C'est une généralisation de la distance Euclidienne et de la distance de Manhattan. Vous pouvez retrouver la formule ci-dessous :

$$\text{Distance de Minkowski} = \sum_{k=0}^{L}|Ha(k) - Hb(k)|$$

Ici Ha et Hb sont respectivement deux histogrammes normalisés différents.

Cette formule est utilisée dans le calcul de la distance de la fonction ci-dessous :

```
1 double distance = 0.0;
2 for (int i = 0; i < histSize; i++) {
3         distance += fabsf(histN.at<float>(i) - histN.at<float>(i));
4 }
```

1.4 Métrique de qualité

L'objectif est de rechercher des métriques de qualité pour le contraste, l'entropie et la duplication afin de bien éliminer 95% des images.

Des tests caractéristiques vont donc être engagés afin de fixer les seuils de façon empirique. Ces seuils seront retenus selon les contraintes de stockage qui sont propres à la station de travail.

Ces tests caractéristiques nous ont amenés à utiliser une métrique de distance correspondant à des changements de plan afin d'éliminer une très grosse partie des images qui correspondrait

à des doublons. Les métriques de contraste et d'entropie sont encore à fixer et ne serviraient seulement qu'à rejeter des images passant l'algorithme de changement de plan et qui sont de mauvaise qualité.

2 Modélisation proposée

Notre solution va donc utiliser la fonction calcHist pour récupérer les histogrammes et les différentes implémentations de filtres présentés ci-dessus.

Ces différents filtres retournent des booléens en fonction des seuils et lorsqu'une image passe tous les filtres, celle-ci est alors enregistrée. Chacune des 480 images par seconde passera donc tous les filtres et une infime partie d'entre elles sera alors sauvegardée en fonction des résultats des filtres.

Toute cette partie OpenCV sera présente dans une solution prdOpenCV qui correspond à la partie analyse d'image.

5 Mise en œuvre

1 Outils et librairie utilisés

La partie conception étant la continuité de la partie analyse, j'ai utilisé les mêmes librairies, à savoir :

- AVerMedia (SDK présent sur la station de travail)
- OpenCV (4.6.0)
- OpenMP (Librairie présente sur Visual Studio 2022)

RapidXML est une librairie qui a été rajouté lors de la phase conception qui permet de manipuler les fichiers XML pour des fins de configuration. Elle est totalement intégrée aux solutions grâce à un fichier header.

2 Séparation des solutions

Pour la phase de conception, il a finalement été plus judicieux de séparer la solution de capture de la partie analyse d'image en raison d'une intégration complexe.

Le projet a donc été découpé en 2 solutions :

- prdOutput : Solution intermédiaire qui représente la solution de capture d'image avec un paramètre de capture bloquant qui va rejeter 95% des images sans critère quelconque
- prdOpenCV : Solution d'analyse d'image qui va simuler l'acquisition d'AVerMedia et rejeter 95% des images basées sur un algorithme de changement de plan. Inclus les calculs de contraste et d'entropie pour proposer un rejet sur seuil empirique et une analyse complète de ces paramètres

3 Solution intermédiaire

3.1 Présentation de la solution

Cette solution a pour but de préparer la future intégration d'OpenCV a la solution de capture d'image. Il y a donc la partie biprocesseur qui est totalement gérée et la mise en place de

paramètres de capture qui permettent la modification du fonctionnement de la capture. La fonction captureStreamImage a été totalement refaite pour préparer l'arrivée d'OpenCV et donc l'arrêt de l'utilisation d'AVerMedia sur cette partie. À terme, seule cette fonction devrait être modifiée puisqu'il faudrait être en mesure d'utiliser les handlers d'AVerMedia avec OpenCV.

3.2 Présentation des classes

3.2.1 Classe StreamImage

La classe StreamImage permet d'initialiser et de détruire les flux TV récupérés par le SDK de AVerMedia. Elle comporte donc des méthodes afin de manipuler les Handlers de chaînes le plus proprement possible afin d'éviter toute éventuelle erreur au lancement du programme. Cette classe a, en majeure partie, été réalisé lors d'un précédent stage par Yondga Lin.

La fonction très importante dans cette classe est getAllHandleObject qui nous permet de récupérer tous les Handlers des chaînes qui ont correctement été initialisés. Cette fonction va être utilisée dans le Main pour effectuer par la suite la répartition des Handlers pour chaque thread.

Bien évidemment, ce getter ne fonctionnerait pas sans la fonction seekAvailableChannel qui va effectuer la vérification d'initialisation de chaque flux.

3.2.2 Classe CaptureImage

La classe CaptureImage est la classe représentant la capture d'image. Elle n'a qu'une seule fonction qui est captureStreamImage et qui va prendre de nombreux paramètres pour capturer un thread de façon totalement paramétrable. Voici la liste des paramètres importants :

- hObject : Handler de la chaîne a capturé
- path : Répertoire d'écriture des images capturées
- dwDuration : Durée de la capture
- FPS : Nombre d'images à capturer par seconde
- picEveryNImage : Nombre d'images capturées avant de pouvoir en sauvegarder une

Cette fonction est appelée par chacun des threads pour capturer la chaîne qui lui a été affectée.

Vous pouvez retrouver la boucle de capture qui va sauvegarder 95% des images capturées.

```
1  while (nbImage < nbCapture) {
2
3                  // Accept to save an image every <picEveryNImage> image (BlockedF
4                  if (nbImage % picEveryNImage == 0) {
5                          imageToTake++;
6                  }
7
8                  // Use path to create the filename for AVerMedia function
9          [...]
10
11
12                 // Save an image if we are authorized (BlockedFPS)
13                 if (imageToTake > 0 ) {
14                         // AVerMedia function to save an image
15                         AVerCaptureSingleImage(hObject, &pCaptureSingleImageInfo
16                         imageToTake--;
```

```
17                  }
18
19                  // Pause between every image
20                  std::this_thread::sleep_for(microseconds(delta));
21                  nbImage++;
22          }
```

Cette boucle va faire "picEveryNImage" pauses puis capturer une image avec la fonction AVerCaptureSingleImage et refaire "picEveryNImage" pauses.

L'objectif à terme est d'enlever l'utilisation de AVerCaptureSingleImage pour remplacer cette fonction par l'utilisation d'OpenCV ce qui va permettre d'introduire les différents algorithmes de rejet.

3.2.3 Main

Le Main de la solution de capture d'image à évoluer tout au long de l'année, que ce soit grâce à ce PRD ou à travers le projet ASR qui visait à intégrer l'architecture biprocesseur dans le programme.

Le Main commence par récupérer les variables de configuration présente dans le fichier "config.xml". Il initialise ensuite les variables AVerMedia et les handlers.

L'initialisation des variables de capture et notamment du paramètre de capture bloquant se fait comme suit :

```
1 // Initialization of the blocked framerate
2         int PicEveryNImage, imagePerMinute = 30;
3         PicEveryNImage = (60 * dwFPS) / ((float)imagePerMinute / NB_THREADS);
4         cout << PicEveryNImage << endl;
```

Le nombre d'images par minute représente le nombre d'images global maximum à sauvegarder pour l'ensemble des threads. La variable picEveryNImage qui est utilisé par la classe captureImage est calculé à partir du nombre d'images par minute citée précédemment, le nombre de threads utilisés et le FPS.

Ensuite, il y a l'intégration de l'architecture biprocesseur avec l'affectation des threads aux deux processeurs grâce aux affinités de groupe Windows. Pour plus d'information sur cette partie, je vous invite à consulter le projet ASR réalisé par Romuald Duret et moi-même.

Finalement, chaque thread récupère les variables présentées précédemment et lance la capture d'image grâce à la fonction captureStreamImage de la classe CaptureImage.

4 Solution d'analyse d'image

4.1 Présentation de la solution

La solution d'analyse d'image simule le flux des cartes d'acquisition d'AVerMedia afin de recréer le fonctionnement de capture d'un flux. La capture utilise OpenCV afin de rejeter 95% des images pour sauvegarder le reste.

4.2 Présentation des fonctions

4.2.1 Les filtres

Les filtres OpenCV ont déjà été présentés dans la partie Analyse et conception et n'ont pas été modifiés lors de la phase de mise en œuvre.

4.2.2 convertRGBtoYUV

La fonction convertRGBtoYUV va simuler l'acquisition des flux TV en enregistrant toutes les images d'une vidéo RGB au format YUV. Par la suite, ces images seront récupérées par les autres fonctions comme si elles venaient d'être réceptionnés par les cartes AVerMedia. Cette fonction a été introduite lors de la veille technologique.

4.2.3 getImage

Cette fonction va récupérer les images créer grâce à la fonction convertRGBtoYUV afin de simuler le système de capture sur un seul flux.

Cette fonction débute avec la définition des paramètres de capture OpenCV présent ci-dessous :

- hist, histN, oldHistN, image : les matrices OpenCV utilisés pour les calculs
- FPS, ImagePerMinute : le FPS et le nombre d'images maximum sauvegardable par minute
- nbImage : le nombre d'images capturées
- imageSinceChange : le nombre d'images depuis le dernier changement de plan
- ImageToTake : le nombre d'images que le programme a le droit de sauvegarder
- PicEveryNImage : le nombre d'images à capturer avant de pouvoir en sauvegarder une

Le calcul de PicEveryNImage a déjà été présenté pour la solution précédente.

Le programme va donc entamer la capture des images et va intégrer un algorithme de changement de plan qui à chaque changement de plan va effectuer plusieurs actions. Le programme va d'abord voir s'il a le droit de sauvegarder la dernière image du plan précédent et agir en conséquence. Le changement de plan étant passé, on va ensuite contrôler la possibilité d'un changement de plan progressif avec l'aide de la variable imageSinceChange. Cette mesure permet d'éviter la capture de doublon sur un changement de plan progressif. Ainsi, si un plan ne dure pas plus de 4 images, on va considérer qu'un changement de plan progressif est en cours et qu'il ne faut absolument pas sauvegarder d'image sur cette période. Ce seuil de 4 images a été fixé de façon empirique.

Concernant le seuil de distance qui nous permet d'affirmer qu'il y a eu un changement de plan, il a aussi été fixé de façon empirique à 0,1.

Finalement, cette fonction va s'arrêter lorsqu'il n'y a plus d'image à lire sur la simulation de flux.

4.2.4 studyParameter

La fonction studyParameter est une version allégée de getImage qui ne va plus prendre en compte la sauvegarde des images, mais qui va récupérer les valeurs de contraste et d'entropie. Elle récupère aussi la distance de Minkowski entre une image T et une image T-1. Ces 3 valeurs sont écrites dans un fichier CSV en fonction d'une variable delta. Cette variable delta représente le FPS. La vidéo utilisée pour simuler le flux étant de base à 25 FPS, le nombre de FPS final pris en compte par cette fonction sera égale à :

$FPS = 25/delta$

Cette fonction est utilisée afin d'analyser les paramètres et d'analyser l'influence du FPS sur les différentes distances de Minkowski.

4.3 Analyse des paramètres

L'analyse des paramètres va être découpé en 2 étapes. On va tout d'abord déterminer l'impact du FPS sur les distances récupérées grâce au paramètre delta. Par la suite, on va observer les valeurs récupérées pour le contraste et l'entropie avec un paramètre delta fixé à quatre afin de définir empiriquement un seuil qui permettrait de capturer seulement les images correspondant aux 75% meilleures en termes de qualité.

4.3.1 Étude de l'impact du FPS

L'objectif de cette étude est de documenter l'impact du FPS sur la distance calculée entre deux images. Afin de mener cette étude, une vidéo de 10 minutes a été utilisée. Pour un paramètre delta fixé à 1, cette vidéo va générer 18 900 images tandis que pour paramètre delta fixé à 4, cette vidéo va générer 4 600 images. Ce grand écart est volontaire, car cela devrait générer un changement important sur les distances calculées si le FPS a un éventuel impact.

Voici les distances calculées en fonction du paramètre delta :

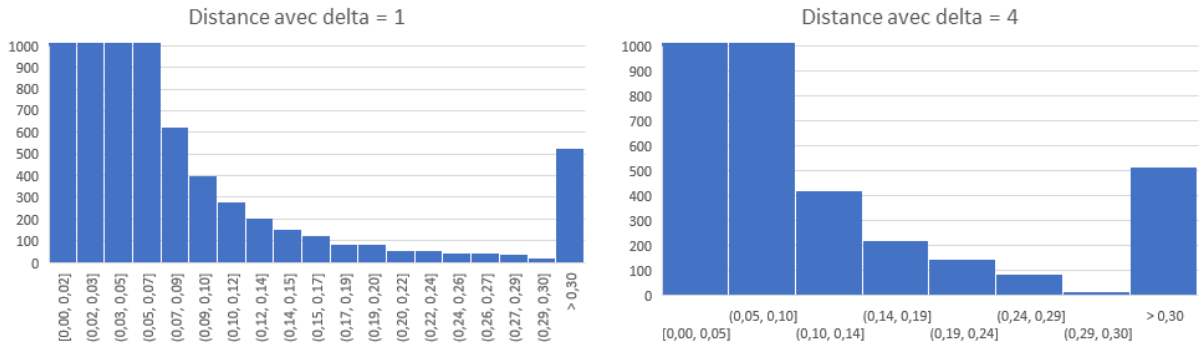


Figure 5.1 – *Les différentes valeurs de distance en fonction de delta*

Un seuil de 0,3 a été fixé sur les histogrammes, car avant ce seuil, les valeurs ont tendance à être trop concentrés pour être intéressantes.

On remarque qu'il y a autant d'image dépassant le seuil de 0,3 de distance, que ce soit avec un paramètre delta égal à 1 ou à 4. Cette égalité décrit clairement un impact du FPS sur le calcul des distances. De plus, la moyenne des distances lorsque le paramètre delta est fixé à 1 est de 0,054 alors que la moyenne est à 0,136 lorsque le paramètre delta est fixé à 4.

Cet impact est logique puisque les images sont plus écartées dans le temps et ont donc plus de chance d'avoir des différences.

Il est compliqué d'utiliser un seuil plus élevé que 0,3 pour repérer les changements de plan, et ce, quelque soit le delta fixé, car certains changements étant progressif (d'où le nombre élevé de valeurs) il est important de les détecter dès le début. Dans le cas contraire, on pourrait se retrouver avec une image floue. Un seuil permettant d'éviter ce genre de problème se situerait donc entre 0,2 et 0,25.

4.3.2 Étude du contraste et de l'entropie

Cette étude a été menée avec un paramètre delta fixé à 4 afin d'éliminer un maximum de doublon et éviter un surplus de données similaires. Ce qui nous intéresse ici, c'est d'étudier les valeurs du contraste et d'entropie afin de trouver des seuils nous permettant de récupérer les 20 % d'images les plus qualitatives. L'étude a été réalisée sur une vidéo de 10 minutes, ce qui nous a permis de récupérer un total de 4 600 images.

Un seuil qui récupère 20% des images serait donc un seuil qui récupère environ 900 images.

Voici les valeurs de contraste récupérées :

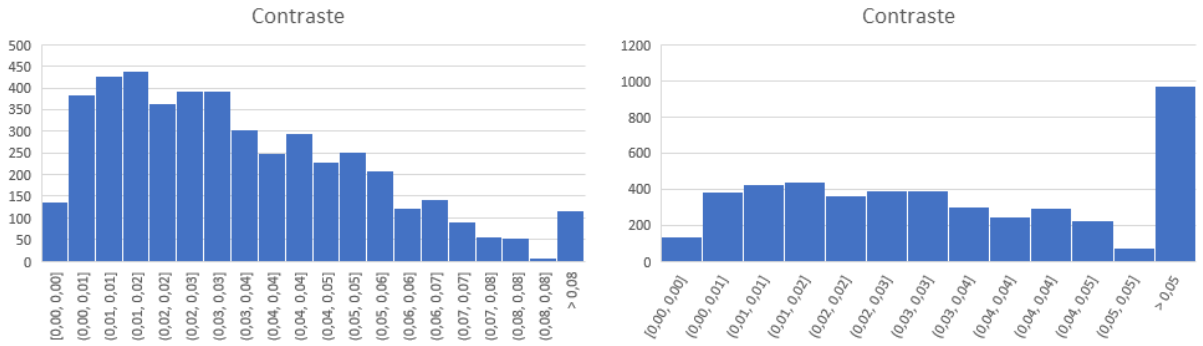


Figure 5.2 – *Les différentes valeurs de contraste*

On peut voir sur le graphique de gauche que les valeurs de contraste sont bien répartis et suivent presque une fonction gaussienne, mais il n'y a pas assez de plan pour lisser les "bruits" de certain plan. Récupérer 25% des valeurs correspondrait à choisir les images ayant un contraste plus élevé que 0,05 comme indiqué sur le graphique de droite.

Voici les valeurs d'entropie récupérées :

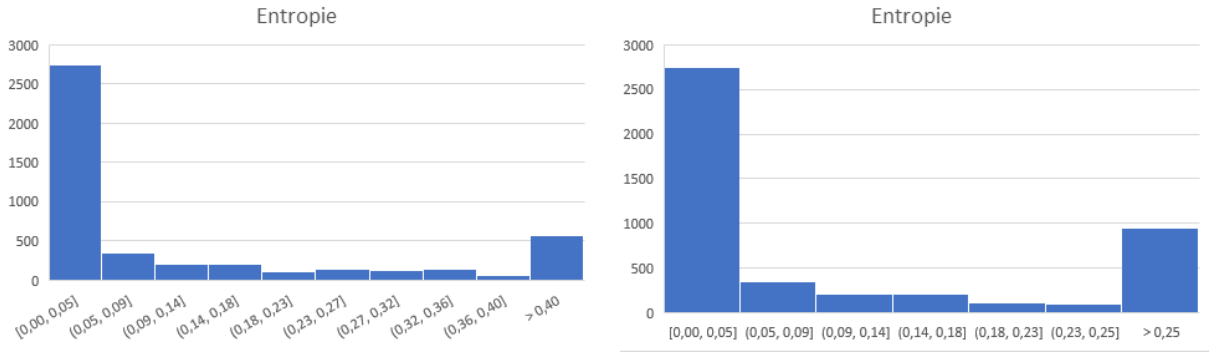


Figure 5.3 – *Les différentes valeurs d'entropie*

On peut voir sur le graphique de gauche que les valeurs d'entropie sont extrêmement concentrées, ce qui nous permet de facilement rejeté une très grosse partie des images. Récupérer 25% des images correspondrait à choisir les images ayant une entropie plus élevée que 0,25, comme indiqué sur le graphique de droite.

6 Bilan et conclusion

1 Bilan du semestre 9

Durant le semestre, j'ai effectué toute la partie spécification et analyse. J'ai aussi programmé la récupération d'histogrammes et les différents filtres basés sur les histogrammes. Un gros jeu de données a aussi été réalisé avec l'aide d'OpenCV afin de calculer, par la suite, les métriques de qualité.

Il me reste donc le calcul des métriques de qualité et l'intégration de l'amélioration dans le programme de capture existant.

Pour le semestre 10, je commencerai par rechercher les différentes métriques de qualité. Avec l'aide de la veille technologique ainsi que de l'analyse et de la conception des filtres, je déploierai une solution finale de filtre d'images. Le programme sera alors terminé et il ne me restera plus qu'à l'intégrer au programme existant.

Pour plus d'informations sur la durée de chaque tâche et l'organisation, vous retrouverez en annexe la gestion de projet concernant le semestre 10.

2 Bilan du semestre 10

Durant le semestre, j'ai effectué toute la partie conception. J'ai préparé une solution intermédiaire prdOutput qui illustre le fonctionnement final de l'application sans l'intégration d'OpenCV. Les fonctionnalités OpenCV décrites lors du semestre 9 sont intégralement présentes dans la solution prdOpenCV.

La suite du projet serait la manipulation des valeurs d'acquisition d'AVerMedia pour pouvoir les utiliser avec OpenCV et ainsi inclure toute la partie filtre des images.

3 Bilan sur la qualité

L'application est entièrement commentée et détaillée et possède une documentation Doxygen afin de permettre à d'autres personnes de reprendre le code sans perdre beaucoup de temps sur la compréhension du code.

Les fonctionnalités sont bien découpées pour faciliter la compréhension du code et permettre la réutilisation de chaque partie précise très simplement.

4 Bilan auto-critique

La partie spécification de mon projet s'est bien déroulé et la partie conception s'est plutôt bien déroulé même si j'aurais pu fixer plus de réunions avec mon encadrant pour mieux avancer sur le projet. La partie gestion de projets aurait pu être mieux réalisé avec plus d'expérience.

Annexes

A Planification, gestion de projet

1 Évolution du projet

1.1 Outils utilisés

Pour effectuer les différents diagrammes de Gantt, j'ai utilisé le logiciel GanttProject. C'est un logiciel que je connaissais déjà et qui m'est donc familier.

Pour la gestion des tâches, j'ai utilisé Trello qui est simple d'utilisation et permet un suivi efficace des tâches restantes et des tâches réalisées.

Le logiciel de versionnage Git sera utilisé lors du S10 pour contrôler au mieux l'avancement et l'intégration des différents modules/filtres au programme de capture.

1.2 Semestre 9

1.2.1 Diagramme de Gantt

Le diagramme de Gantt initial pour la planification de ce projet lors du semestre 9 :

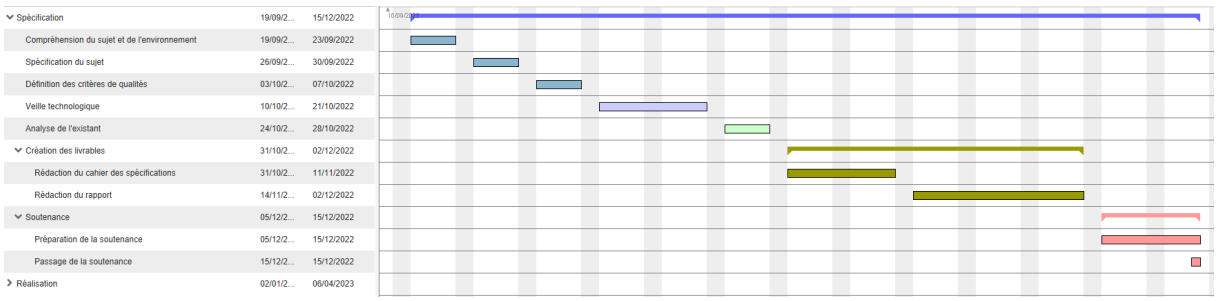


Figure A.1 – *Le diagramme de Gantt S9 Initial*

Le diagramme de Gantt final de ce projet pour le semestre 9 :

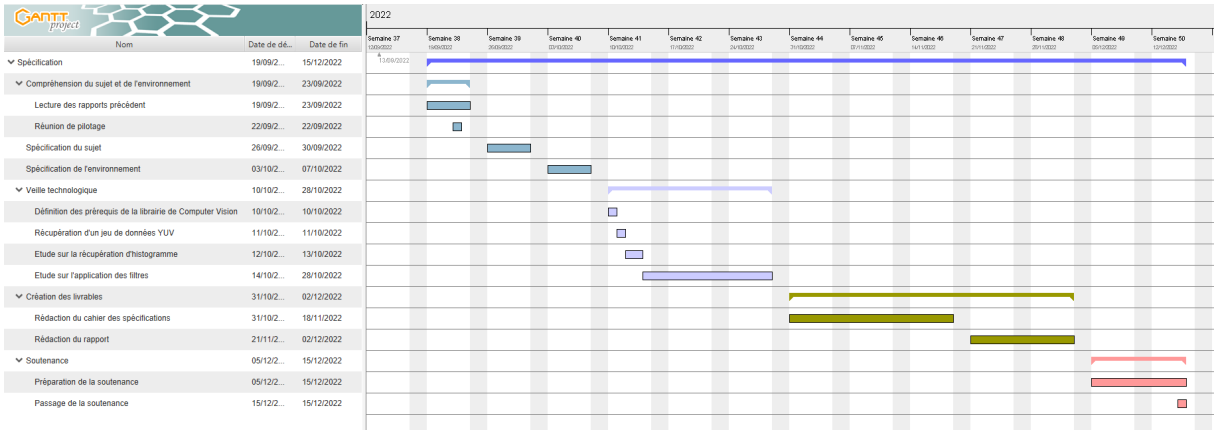


Figure A.2 – *Le diagramme de Gantt S9 Final*

1.2.2 Gestion de tâches

La gestion des tâches final du projet pour le semestre 9 :

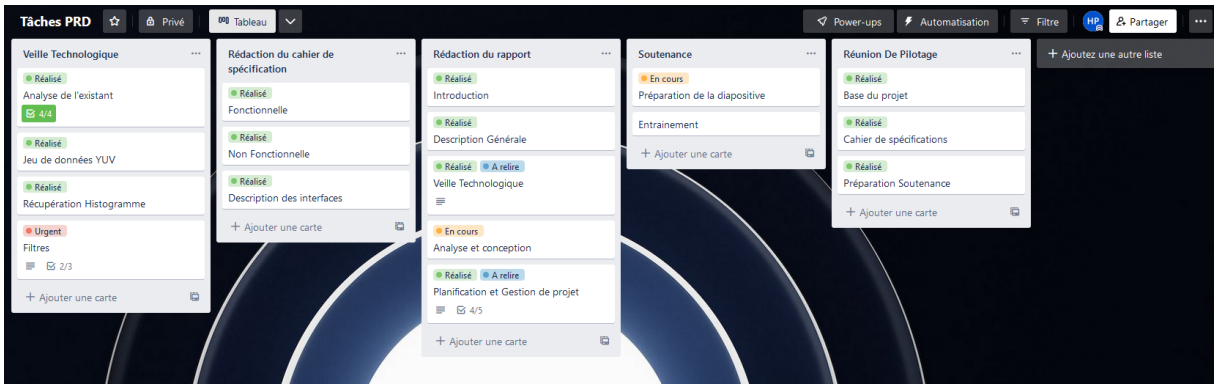


Figure A.3 – *Gestion de tâches Semestre 9*

1.3 Semestre 10

1.3.1 Diagramme de Gantt

Le diagramme de Gantt initial pour la partie développement de ce projet lors du semestre 10 :

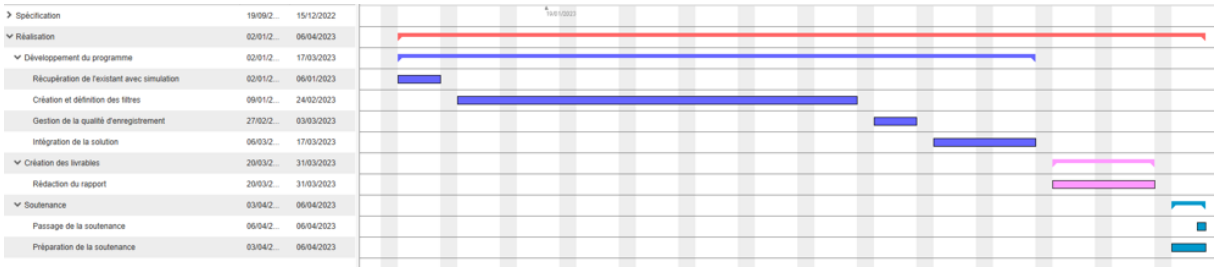


Figure A.4 – *Le diagramme de Gantt S10 Initial*

Le diagramme de Gantt final pour la partie développement de ce projet lors du semestre 10 :

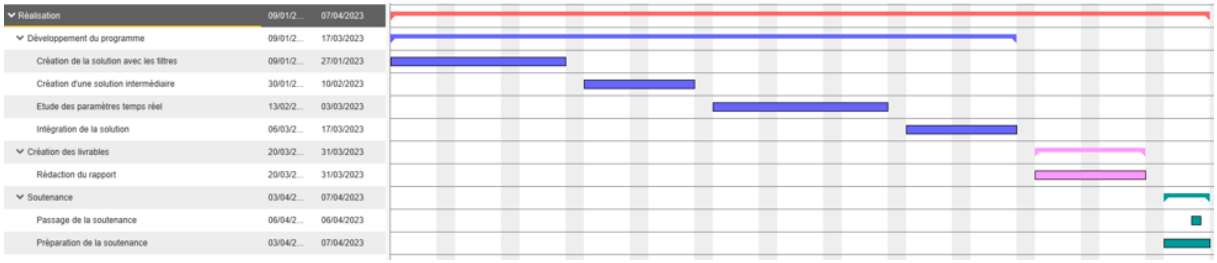


Figure A.5 – *Le diagramme de Gantt S10 Final*

1.3.2 Gestion de tâches

La gestion des tâches prévue du projet pour le semestre 10 :

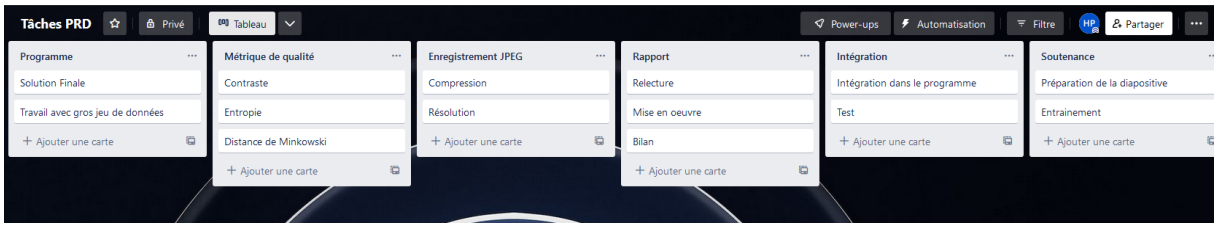


Figure A.6 – *Gestion de tâches Semestre 10*

2 Analyse de la gestion de projet

Globalement, la gestion de projet s'est plutôt bien déroulé. Lors du semestre 9 j'ai eu quelques problèmes, car j'avais mal mesuré le temps de travail de certaines tâches. Ce problème a été résolu en réattribuant plus efficacement les tâches à travers le temps et en découpant mieux les tâches pour mieux établir des temps de travail pour les tâches. Ce découpage n'était quand même pas aussi précis que celui présent sur Trello car il faut pouvoir se retrouver sur un Gantt.

Finalement, je dirai que le plus gros manquement sur ma gestion de projet était la parallélisation des tâches. Je n'ai en effet pas établi d'antécédent pour les tâches, ce qui m'aurait permis d'en paralléliser certaines et peut-être de travailler plus efficacement.

B Description des interfaces

1 Interfaces matérielles/logicielles

L'amélioration du programme de capture n'aura pas d'interaction avec le matériel. Le programme de capture quant à lui interagit avec les 6 cartes d'acquisition via le SDK de AVerMedia.

2 Interfaces homme/machine

Le programme n'a pas vocation à avoir d'interface. Ce sera un programme C++ exécutable.

Les paramètres seront passés au programme grâce à un fichier de configuration XML.

3 Interfaces logiciel/logiciel

L'amélioration du programme de capture n'aura pas d'interaction avec un logiciel. Il y aura néanmoins des interactions avec la librairie OpenCV. Le programme de capture quant à lui interagit avec le SDK de AVerMedia

C Cahier de Spécifications

1 Spécifications Fonctionnelles

1.1 Fonctionnalités à développer

1.2 Définition de la fonction 1 : Récupération d'histogramme

Présentation de la fonction 1 :

Récupère, à partir d'une image dans l'espace de couleur YUV, son histogramme dans l'espace de couleur RGB. Cette fonction devra être optimisée le plus possible, car elle peut être assez lourde si elle est exécutée un grand nombre de fois par seconde.

Description de la fonction 1 :

- Entrées :
  - Une image en espace de couleur YUV appelée X
- Sorties :
  - L'histogramme de l'image X en niveau de gris
- Algorithme simple :
  - Transforme l'image de YUV à RGB
  - Transforme l'image de RGB à GRIS
  - Retourne l'histogramme et l'histogramme normalisé de l'image en nuance de gris

1.3 Définition de la fonction 2 : Filtre sur le contraste

Filtre une image en fonction de son contraste avec l'aide d'un seuil prédéfini afin de ne récupérer que les images les plus qualitatives. Ce filtre va éliminer les images avec un contraste faible, donc les images avec tous leurs pixels concentrés dans la même zone de couleur.

Description de la fonction 2 :

- Entrées :
  - Un histogramme en niveau de gris normalisé H correspondant à l'image X
  - Un seuil définissant un contraste minimum
- Sorties :
  - Un booléen indiquant si l'image doit ou ne doit pas être stocké
- Algorithme simple :
  - Calcul de la moyenne de l'histogramme normalisé
  - Calcul de l'écart type de l'histogramme normalisé
  - Calcul du contraste à partir de l'écart type
  - Retourne un booléen Vrai si le contraste est supérieur au seuil, sinon Faux

1.4 Description de la fonction 3 : Filtre sur l'entropie

Filtre une image en fonction de son entropie avec l'aide d'un seuil prédéfini. Ce filtre va éliminer les images avec une entropie trop basse, donc les images avec un trop grand nombre de pixels concentrés dans la même zone de couleur.

Description de la fonction 3 :

- Entrées :
  - Un histogramme en niveau de gris normalisé H correspondant à l'image X
  - Un seuil définissant une entropie minimum
- Sorties :
  - Un booléen indiquant si l'image doit ou ne doit pas être stockée
- Algorithme simple :
  - Calcul de l'entropie grâce à l'histogramme normalisé
  - Retourne un booléen Vrai si l'entropie est supérieure au seuil, sinon Faux

1.5 Description de la fonction 4 : Filtre sur la différence d'image

Filtre une image en fonction de la différence par rapport à l'image précédente. Cet algorithme de duplicata va filtrer un grand nombre d'images sur des programmes lents comme un JT et sera moins efficace sur des programmes d'action.

- Entrées :
  - Un histogramme en niveau de gris normalisé H(T) correspondant à l'image X(T).
  - Un histogramme en niveau de gris normalisé H(T) correspondant à l'image X(T-1).
  - Un seuil définissant une différence minimum.
- Sorties :
  - Un booléen indiquant si l'image doit ou ne doit pas être stocké
- Algorithme simple :
  - On compare l'histogramme X(T) et l'histogramme X(T-1) avec l'aide de la distance de Minkowski
  - Si cette valeur est supérieure au seuil, on retourne Vrai, sinon Faux

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

2.1.1 Langage de programmation

Le programme de capture d'image a été développé en C++. Le programme étant une amélioration de celui-ci, il sera donc aussi en C++.

2.1.2 Bibliothèque à utiliser

Les images sont récupérées par les cartes d'acquisition avec l'aide du SDK de AVerMedia. Cette partie est déjà implémentée dans le programme de capture. Notre programme nécessite une bibliothèque d'analyse d'images capable de récupérer la matrice de pixel de AVerMedia et de l'utiliser pour récupérer un histogramme et effectuer des analyses dessus.

La librairie OpenCV permet d'effectuer toutes ces opérations et est beaucoup documentée.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

L'algorithme de récupération d'histogrammes et l'utilisation des différents filtres seront effectués 480 fois par seconde (Nombre de Flux * FPS). Nous devons donc avoir un programme optimisé capable d'effectuer un tel nombre d'opérations par seconde.

D Document d'installation et d'utilisation

1 Installation

Pour faire fonctionner les deux solutions du PRD, il faut télécharger OpenCV et utiliser Visual Studio 2022 (version utilisée lors du PRD).

Les binaries de OpenCV doivent être renseignés dans la variable d'environnement PATH :

— <OpenCVFolder>/x64/vc15/bin
— <OpenCVFolder>/build/bin

La librairie RapidXML est déjà importé dans le projet en tant que header et OpenMP est inclus de base dans Visual Studio 2022 et n'a qu'à être activé.

2 Configuration

Il faut maintenant se rendre dans la configuration de chaque projet Visual Studio 2022 pour paramétrer OpenCV et OpenMP :

— Répertoires VC++ → Répertoire Include : <OpenCVFolder>/build/include
— Répertoires VC++ → Répertoire de bibliothèques : <OpenCVFolder>/build/x64/vc15/lib
— C/C++ → Langage → Prise en charge OpenMP : Oui

Afin de facilement configurer les paramètres des applications, il y a un fichier XML dans le dossier de chaque solution Visual Studio 2022. Ce fichier s'appelle "config.xml" et on peut y définir :

— nbThread : Nombre de threads pour faire tourner l'application (merci de choisir un nombre pair pour éviter tout problème lors de l'affectation des groupes)
— RepertoireCpu1 : Le dossier d'écriture des threads de capture tournant sur le 1er CPU
— RepertoireCpu2 : Le dossier d'écriture des threads de capture tournant sur le 2d CPU

Certaines de ces options ne sont pas disponibles pour la partie analyse d'image, car elle ne nécessite pas plusieurs threads et donc plusieurs processeurs.

3 Utilisation

Après configuration des projets l'utilisation de ces derniers se fait avec Visual Studio 2022 en build Release afin d'éviter les messages de debug de OpenCV.

E Glossaire

JPEG JPEG est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe. . 2, 6, 7, 35

RAID Le RAID est un ensemble de techniques de virtualisation du stockage permettant de répartir des données sur plusieurs disques durs afin d'améliorer soit les performances, soit la sécurité ou la tolérance aux pannes de l'ensemble du ou des systèmes. . 4

RGB Espace colorimétrique d'une image en 3 composantes. R représente le rouge, G le vert et B le bleu. La superposition de ces 3 composantes forment les images et vidéos que nous sommes habitués à regarder. . 7, 17, 27

SDK Un SDK (Software Developement Kit) est un ensemble d'outils fourni avec une plateforme matérielle. Dans notre cas, ce sont les cartes d'acquisition AVerMedia. . 1, 2, 26, 29

YUV Espace colorimétrique d'une image en 3 composantes. Y représente la luminance (luminosité) de l'image et U et V la chrominance (couleur) de l'image. . v, 5, 7, 8, 17, 27

F Acronymes

ASR Architecture des Systèmes et Réseaux. 4, 16

BdTln Base de données et Traitement des langues naturelles. 1

FPS Image par Seconde. 5, 15–18, 29

LIFAT Laboratoire d'Informatique Fondamentale et Appliquée de Tours. 1, 2, 5

PRD Projet de Recherche et de Dévelopement. 1, 4, 6, 16

RFAI Reconnaissance des Formes et Analyse d'Images. 1
ROOT Recherche opérationnelle, Ordonnancement et Transport. 1

Projet Recherche & Développement

Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV

Hugo PERVEYRIE

Encadrement : Mathieu DELALANDRE

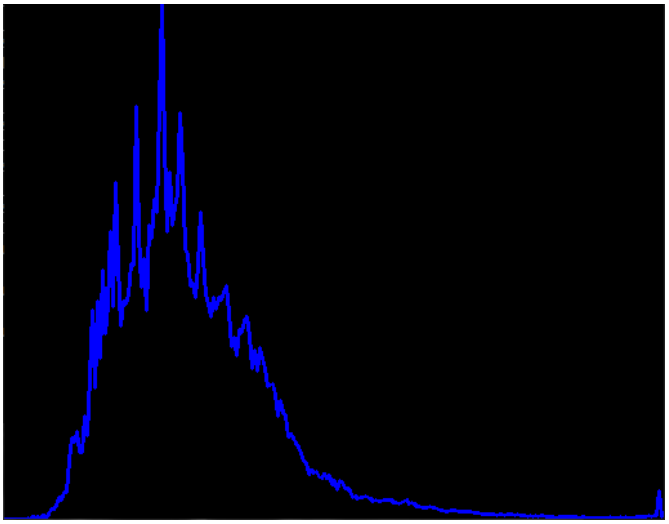
En collaboration avec Polytech

Objectifs

- Répondre à des contraintes de capacité
- Filtrer des images sur des critères de qualité
- Calcul de métrique de qualité

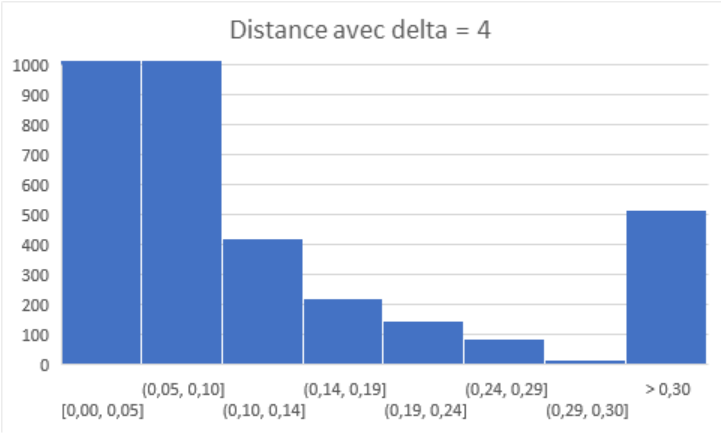
Mise en œuvre

1. Analyse d'histogrammes avec OpenCV
2. Filtre sur le contraste, l'entropie et la duplication
3. Définition empirique des critères de qualité



Résultats attendus

Un programme de capture intelligent capable de récupérer des image de qualité sur plusieurs mois à la suite.



Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Projet Recherche & Développement

Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV

Hugo PERVEYRIE

Encadrement : Mathieu DELALANDRE

En collaboration avec Polytech

Objectifs

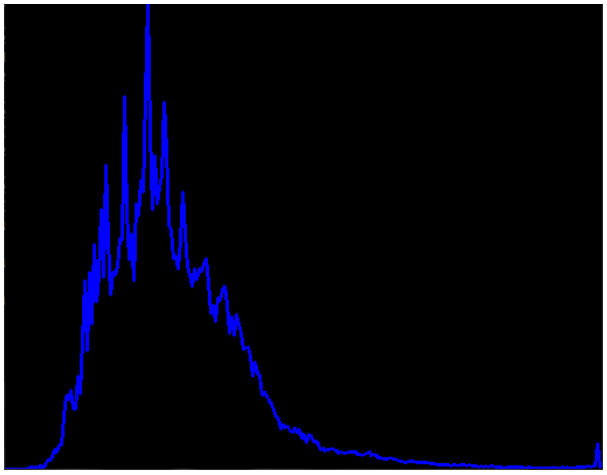
- Répondre à des contraintes de capacité
- Filtrer des images sur des critères de qualité
- Calcul de métrique de qualité

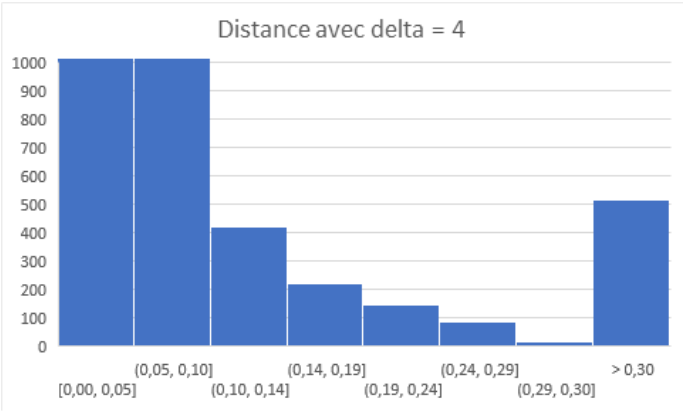
Mise en œuvre

1. Analyse d'histogrammes avec OpenCV
2. Filtre sur le contraste, l'entropie et la duplication
3. Définition empirique des critères de qualité

Résultats attendus

Un programme de capture intelligent capable de récupérer des image de qualité sur plusieurs mois à la suite.





Ecole Polytechnique de l'Université de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Capture Haute Qualité, Smart et à l'échelle d'images JPEG sur la Station TV

Résumé

Améliorer un programme de capture d'image JPEG existant afin de le rendre intelligent. Filtrer 95% des images récupérées à l'aide de filtres appliqués non pas sur le domaine spatial des images mais directement sur leur histogramme. Cette séquence de filtrage permet de répondre à des contraintes de capacité de stockage.

Mots-clés

Analyse, Histogramme, Filtres

Abstract

Improving an existing JPEG image capture program to make it smarter. Filter 95% of the images using a filter applied not to the spatial domain of the images but directly to their histogram. This filtering sequence is mandatory due to capacity constraints of the workstation.

Keywords

Analysis, Histogram, Filters

Entreprise
Polytech

Étudiant
Hugo PERVEYRIE (DI5)

Tuteur académique
Mathieu DELALANDRE

Tuteur entreprise
Mathieu DELALANDRE