**Traitement d’image (SIF1033-00)**

**Phase 1**

**Équipe 3**

**Par**

Balla Camara

Philippe Bordeleau-Marchand

Sylvain Bilodeau

Yannick Poirier

**Présenté à**

Tarik Boukhalfi

10 mars 2025

**Introduction**

Ce rapport a pour objectif d’expliquer notre logique ainsi que notre compréhension de la phase 1 du présent projet. Le rapport vise également à détailler les choix d’algorithmes et leur pertinence dans le cadre de l’application. L’application se sépare en deux fichiers python distincts. Le premier, « Fonctions.py » contient toute la logique d’implémentation de l’application. Le deuxième, « Projet\_Phase1.py » contient la partie exécutable du code, à savoir le script.

**Fonctions.py**

Ce fichier python contient plusieurs fonctions que nous allons décortiquer plus en détail. Nous avons d’abord une simple fonction « clear\_run\_console » qui utilise la librairie de pyautogui. Cette fonction permet simplement de libérer l’espace console afin d’améliorer la lisibilité. Elle n’est pas nécessaire au bon fonctionnement de l’application.

**Webcam**

La fonction suivante, « liste\_webcam » permet de détecter toutes les webcams disponibles sur le système de l’utilisateur. Essentiellement, une liste de webcam est initialisée ainsi qu’une boucle infinie pour tester chaque indice de webcam. La fonction tente ensuite d’ouvrir la webcam et de vérifier qu’il n’y a pas d’erreur lors de l’ouverture (une image est bien capturée). Si tout est conforme, l’indice de cette webcam est ajouté à la liste et on libère la webcam. L’application répète ce processus jusqu’à ce qu’il n’y ait plus de webcam trouvée. La fonction va finalement retourner une liste des indices des webcams accessibles.

La prochaine fonction, « select\_webcam » dépends de la liste retournée par la fonction ci-haute. Elle demande à l’utilisateur d’inscrire le numéro d’index correspondant aux webcams disponibles. Dans la plupart des cas, une seule webcam est disponible, donc l’indice à inscrire serait « 0 ». En gros, cette fonction appelle « liste\_webcam » pour obtenir la liste de webcam disponible. On efface la console pour avoir une meilleure visibilité en appelant « clear\_run\_console ». On affiche ensuite la liste de webcams disponibles avec leur index (numéro). On demande à l’utilisateur d’inscrire le numéro de la webcam qu’il souhaite utiliser. L’application s’assure que l’entrée saisie est valide, sinon on affiche un message d’erreur qui invite l’utilisateur à recommencer. Si l’entrée est valide, l’index de la webcam sélectionnée est retourné.

**Source vidéo**

La fonction « source\_video » permet à l’utilisateur de choisir la source vidéo qu’il souhaite utiliser entre la webcam ou un fichier vidéo. Dans un premier temps, la console est nettoyée avec « clear\_run\_console ». Ensuite, si l’utilisateur inscrit « w », l’application appelle « select\_webcam » pour que l’utilisateur puisse choisir une caméra. Si l’utilisateur préfère coller un chemin vers un fichier vidéo source, cette vidéo sera utilisée comme la source. L’application vérifie que le fichier vidéo s’ouvre correctement (que le chemin source existe). Dans les deux cas de source vidéo, on utilise une fonction de openCV, « cv2.VideoCapture » pour ouvrir la vidéo ou la webcam et pouvoir la lire image par image. Des messages d’erreurs conséquents s’affichent en cas de problème.

**Formatage vidéo**

La première fonction, « reso\_cam » permet de modifier la résolution d’une capture vidéo en définissant sa largeur et sa hauteur en pixels. La fonction utilise openCV pour directement modifier la résolution de la capture vidéo selon ce qui est souhaité. Dans le contexte de la phase 1, la valeur largeur sera définis à 1920 et la valeur hauteur à 1080. Cette fonction ne retourne rien et applique directement les modifications sur l’objet vidéo.

La deuxième fonction, « resize\_video » sert à redimensionner une vidéo afin de l’adapter à une largeur et hauteur en pixels spécifiques, tout en préservant son rapport d’aspect. Cela permet notamment de retourner une image sans modification si elle est déjà de la bonne taille, de la redimensionner directement si elle est de mauvaise taille, ou encore de la redimensionner et recadrer si elle est de mauvaise taille avec un ratio différent (soit plus haute ou plus large). Dans le dernier cas, si la vidéo à un ratio différent, elle est redimensionnée tout en conservant son rapport d’aspect. Elle est ensuite recadrée afin de supprimer les parties excédentaires (largeur ou hauteur). Dans le contexte de la phase 1, les dimensions visées sont une largeur de 1920 et une hauteur de 1080. Ainsi, toutes les vidéos seront affichées en 1920x1080 sans distorsions ni bandes noires.

**Logo UQTR**

La première fonction, « resize\_image » a pour objectif d’extraire une région d’intérêt (ROI) qui contient l’image sans fond transparent afin de redimensionner celle-ci pour avoir une hauteur fixe à 100 pixels, tout en conservant son rapport d’aspect. Dans ce contexte, nous allons utiliser l’image du logo UQTR fournie en annexe 1 pour ce présent projet. L’application sépare d’abord l’image en 4 canaux (RGB + Alpha). L’objectif est ensuite d’identifier les limites du contenu visible en analysant les pixels du canal alpha, pour identifier la partie de l’image non transparente. Une extraction de la zone non transparente se produit, définie par les coordonnées extrêmes des pixels visibles (gauche, droit, haut, bas). Ensuite, on redimensionne le ROI à une hauteur fixe de 100 pixels, tout en maintenant les proportions. L’image redimensionnée est retournée. Ainsi, l’image redimensionnée conserve sa propreté visuelle et ses proportions.

La deuxième fonction, « incrustation » permet d’incruster une image (logo de l’UQTR dans notre cas) sur une vidéo à une position spécifique et avec un degré de transparence et de fusion. La fonction sépare d’abord les canaux du logo (RGB + Alpha), puis crée des masques pour isoler les zones transparentes. Un calcul de positionnement est ensuite effectué afin de déterminer où placer le logo sur la vidéo. Dans notre contexte, on souhaite placer le logo à 10 pixels du haut et 10 pixels du bord droit. Le logo est ensuite fusionné avec la vidéo en utilisant un mélange des pixels de la vidéo et du logo, ce qui donne un effet d’incrustation avec le degré d’opacité souhaité (60%). La vidéo modifiée avec le logo incrusté à la bonne position est ensuite retournée.

**Projet\_phase1.py**

Ce fichier contient le script de notre application de traitement vidéo. Toutes les fonctions que nous avons définies ci-haut dans Fonctions.py sont appelées et utilisées dans le script. Essentiellement, le script va d’abord demander à l’utilisateur de choisir sa source vidéo. La capture vidéo est configurée automatiquement à 1920x1080. Ensuite, le script charge et prépare le logo UQTR en retirant le texte dans le bas à l’aide d’un découpage manuel et en redimensionnant le logo pour que sa plus grande dimension soit de 100 pixels.

Le script contient une boucle principale qui lit chaque trame (image) de la vidéo. Si nécessaire, les images sont redimensionnées en 1920x1080. Pendant la lecture, une incrustation du logo UQTR dans le coin supérieur droit avec une opacité de 60% est fait. De plus, la date et l’heure en temps réel sont également ajoutées dans le coin inférieur gauche.

La vidéo qui est traitée est affichée en temps réel pour l’utilisateur. Le programme va se terminer si la vidéo arrive à son terme, si l’utilisateur appuie sur « q » ou encore si la fenêtre d’affichage est fermée. Finalement, toutes les ressources sont libérées.

Ce script permet de traiter et d’afficher en temps réel une vidéo avec une incrustation de logo et une superposition de l’heure et la date, tout en respectant la résolution cible et les spécifications de la phase 1 de ce présent projet.