PG208, Projet n°4 : Traitement vidéo temps réel

Bertrand LE GAL*, Serge BOUTER†et Clément VUCHENER‡

Filière électronique 2^{eme} année - Année universitaire 2011-2012

1 Introduction

1.1 Objectif du projet

L'objectif du projet est de mettre en œuvre les notions appréhendées durant l'enseignement de de PG208. Durant cet enseignement vous avez normalement acquis les notions de base de la programmation orientée objets appliquées au langage C++. Le point de départ du projet est constitué du présent document. Ce dernier correspond au cahier des charges spécifié par le client (votre enseignant). Afin de mener à bien ce projet, vous allez devoir :

- lire le cahier des charges,
- identifier les besoins exprimés par le client,
- comprendre ce que vous devez développer,
- élaborer un modèle de solution,
- développer et mettre au point votre application.

Pour mener à bien les quatre premières étapes, il vous est demandé de mettre en pratique les notion d'UML vues en cours. Cela vous permettra de structurer vos idées et de les partager (avec votre binôme et votre enseignant). De plus, étant donné que le projet est un projet de conception objets, il vous est demandé dans les deux dernière étapes de mettre en œuvre les notions vues en cours (héritage, classes, polymorphisme, surcharge, etc.).

1.2 Déroulement du projet

La durée estimée du projet est d'environ 18 heures. Six séances de TP de 3 heures sont programmés dans votre emploi du temps. Toutefois, elles ne sont pas toutes encadrées : les 2 premières et les 2 dernières

*bertrand.legal(at)ims-bordeaux.fr
†serge.bouter(at)u-bordeaux1.fr
‡clement.vuchener(at)inria.fr

séances sont encadrées tandis que les 2 séances intermédiaires sont planifiées mais se dérouleront sans encadrement.

Le sujet du projet a été écrit de manière à prendre en considération les différences de niveaux entre les groupes. Pour cela le sujet du projet est écrit sous forme de cahier des charges à tiroir. Cette particularité vous permettra d'avancer à votre rythme.

1.3 Évaluation de votre travail

La notation du projet se basera sur :

- La qualité de l'analyse du cahier des charges. Cette analyse doit être réalisée à l'aide d'UML.
- L'utilisation adéquate des possibilités offertes par le langage C++ (héritage, surcharge, ...).
- Le respect des fonctionnalités spécifiées dans le cahier des charges.

L'évaluation du projet sera réalisée en deux parties :

- Lors d'une présentation orale de votre projet.
- Dans un rapport papier détaillant le développement de votre application.

Lors de la présentation orale de votre application, vous êtes en charge de présenter à votre enseignant (le client) l'application développée. Cette présentation d'une durée approximative de 10 minutes vise à démontrer au client que vous avez développé ce qu'il vous a commandé. Pour cela, vous présenterez vos choix de conception, vous expliquerez les différentes étapes par lesquelles vous êtes passées et enfin une réaliserez démonstration des fonctionnalités de l'outil (démonstration que vous aurez pris soin de préparer).

En ce qui concerne le rapport écrit, une attention particulière sera portée à la présentation des choix d'implantation que vous aurez réalisés lors de votre développement. Ces choix ainsi que votre cheminement devront être détaillés clairement à l'aide du langage UML. De plus une analyse des avantages et inconvénients du langage C++ par rapport au langage C devra être réalisée.

2 Cahier des charges

2.1 Fonctionnalités de base

Comme nous l'avons déjà précisé dans les parties précédentes, l'objectif du projet est de développer une application de traitement vidéo temps réel. La base de l'application permettant d'extraire une vidéo depuis un fichier et de l'afficher à l'écran vous est fournie par votre enseignant. Votre rôle sera d'implanter dans cette application une série de traitements vidéo permettant à l'utilisateur d'analyser en temps réel les images qui sont affichées.

Les traitements vidéo à intégrer dans l'outil seront de différentes natures permettant par exemple de filtrer un canal de couleur, détecter les contours de l'image, etc. La liste ci-dessous présente une liste exhaustive des traitements à implémenter au minimum dans votre application :

- **Filtrage du canal rouge** (le canal rouge garde sa valeur et 0 pour les autres)
- **Filtrage du canal bleu** (le canal bleu garde sa valeur et 0 pour les autres)
- **Filtrage du canal vert** (le canal vert garde sa valeur et 0 pour les autres)
- **Transformation en nuance de gris « simple »**, tous les canaux adoptent la valeur fournie par l'équation donnée ci dessous :

$$Red'_{(x,y)} = \frac{Red_{(x,y)} + Green_{(x,y)} + Blue_{(x,y)}}{3}$$
$$Green'_{(x,y)} = \frac{Red_{(x,y)} + Green_{(x,y)} + Blue_{(x,y)}}{3}$$
$$Blue'_{(x,y)} = \frac{Red_{(x,y)} + Green_{(x,y)} + Blue_{(x,y)}}{3}$$

- **Transformation en nuance de gris « fiable »**, tous les canaux adoptent la valeur fournie par l'équation donnée ci dessous :

$$Red'_{(x,y)} = 0.299 \times Red_{(x,y)} + 0.587 \times Green_{(x,y)} + 0.114 \times Blue_{(x,y)}$$
$$Green'_{(x,y)} = 0.299 \times Red_{(x,y)} + 0.587 \times Green_{(x,y)} + 0.114 \times Blue_{(x,y)}$$
$$Blue'_{(x,y)} = 0.299 \times Red_{(x,y)} + 0.587 \times Green_{(x,y)} + 0.114 \times Blue_{(x,y)}$$

- **Sous échantillonnage (poinçon)**, l'objectif est de diminuer d'un facteur 2 la taille de l'image. Le sous échantillonnage est réalisé à l'aide de la fonction donnée ci-dessous :

$$Red'_{(x,y)} = Red_{(2\times x,2\times y)}$$
$$Green'_{(x,y)} = Green_{(2\times x,2\times y)}$$
$$Blue'_{(x,y)} = Blue_{(2\times x,2\times y)}$$

- **Sous échantillonnage (linéaire)**, l'objectif est de diminuer d'un facteur 2 la taille de l'image. Le sous échantillonnage est réalisé à l'aide de la fonction donnée ci-dessous :

$$Red'_{(x,y)} = \frac{Red_{(2\times x,2\times y)} + Red_{(2\times x+1,2\times y)} + Red_{(2\times x,2\times y+1)} + Red_{(2\times x+1,2\times y+1)}}{4}$$
$$Green'_{(x,y)} = \frac{Green_{(2\times x,2\times y)} + Green_{(2\times x+1,2\times y)} + Green_{(2\times x,2\times y+1)} + Green_{(2\times x+1,2\times y+1)}}{4}$$
$$Blue'_{(x,y)} = \frac{Blue_{(2\times x,2\times y)} + Blue_{(2\times x+1,2\times y)} + Blue_{(2\times x,2\times y+1)} + Blue_{(2\times x+1,2\times y+1)}}{4}$$

– **Sous échantillonnage (cubique)**, l'objectif est de diminuer d'un facteur 2 la taille de l'image. Le sous échantillonnage est réalisé à l'aide de la fonction donnée ci-dessous :

$$Red'_{(x,y)} = \sqrt[2]{\frac{Red^2_{(2\times x,2\times y)} + Red^2_{(2\times x+1,2\times y)} + Red^2_{(2\times x,2\times y+1)} + Red^2_{(2\times x+1,2\times y+1)}}{4}}$$

$$Green'_{(x,y)} = \sqrt[2]{\frac{Green^2_{(2\times x,2\times y)} + Green^2_{(2\times x+1,2\times y)} + Green^2_{(2\times x,2\times y+1)} + Green^2_{(2\times x+1,2\times y+1)}}{4}}$$

$$Blue'_{(x,y)} = \sqrt[2]{\frac{Blue^2_{(2\times x,2\times y)} + Blue^2_{(2\times x+1,2\times y)} + Blue^2_{(2\times x,2\times y+1)} + Blue^2_{(2\times x+1,2\times y+1)}}{4}}$$

– **Sur-échantillonage (poinçon, linéaire, cubique)**, l'objectif est d'augmenter d'un facteur 2 la taille de l'image. Le sur échantillonnage est réalisé à l'aide des fonctions données ci-dessous :

$$Red'_{(2\times x,2\times y)} = Red_{(x,y)}$$

$$Red'_{(2\times x+1,2\times y)} = Red_{(x,y)}$$

$$Red'_{(2\times x,2\times y+1)} = Red_{(x,y)}$$

$$Red'_{(2\times x+1,2\times y+1)} = Red_{(x,y)}$$

– **Détection de contour (filtrage passe haut)**, la détection de contour peut être appliquée sur une image couleur ou transformée en nuance de gris. Le filtrage sera assuré par une convolution assurée par les matrices données ci-dessous pour les filtres à implémenter :

$$M_0 = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$M_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$M_5 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$M_6 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$M_7 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$M_8 = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

$$M_9 = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

– **Flou (filtrage passe bas)**, le filtrage de l'image peut être appliquée sur une image couleur ou transformée en nuance de gris. Le filtrage sera assuré par une convolution assurée par les matrices données ci-dessous pour les filtres à implémenter :

$$B_0 = \frac{1}{8} \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$B_1 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 8 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$B_2 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$B_3 = \frac{1}{8} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

– **Adaptation automatique des niveaux de couleur** afin d'exploiter toutes les possibilités d'affichage. Cette transformation se base sur la réalisation d'un histogramme pour chacun des canaux de l'image : on calcul les valeurs minimum et maximum des valeurs pour chaque canal et ensuite on adapte les valeurs des composantes de chaque pixel à l'aide de la formule suivante :

$$C_u = \frac{255}{max - min}$$

$$u'_{(x,y)} = C_u \times (u_{(x,y)} - min)$$

– **Motion blur**, afin de donner une exagération des mouvements dans la vidéo. Pour réaliser ce filtre il est nécessaire de mémoriser les N images précédentes de la séquence et d'en réaliser une somme pondérée. Dans votre cas, nous allons considérer que la somme des 2 dernières images avec l'image courante est suffisante. La somme pondérée sera réalisée à l'aide de l'équation suivante :

$$P'_{(x,y)} = \frac{1}{2} \times P_{(x,y)} + \frac{3}{10} \times P_{2(x,y)} + \frac{2}{10} \times P_{3(x,y)}$$

– **Transformations complexes**, nous souhaitons aussi mettre en ?uvre des transformations plus complexes réalisant successivement plusieurs transformations. Vous devrez par exemple développer les « traitements suivants » :
  – Sortie = Image sous échantillonnée cubiquement d'un facteur 4 ;
  – Sortie = Image sous échantillonnée par 2 avec détection de contours
  – Sortie = Image en nuance de gris floutée et agrandie d'un facteur 2 (linéairement)
  – Sortie = Image avec détection de contour réalisée sur une image floutée et passée en nuance de gris.
– **Autres transformations** en fonction de vos envies ? En fonction du temps restant et de vos connaissances, vous pouvez ajouter à cette liste de nouvelles transformations ou des compositions de ces dernières : comme par exemple la squelettisation, la détection d'objets, etc.

Afin d'insérer ces différents filtres dans l'outil vous devrez bien réfléchir à la question suivante : quels sont les points communs entre les différentes transformations (uniformisation des classes afin de bénéficier de la compatibilité) ?

2.2 Extension de l'application

Dans un second temps vous devrez modifier votre conception afin de permettre à l'utilisateur de réaliser un assemblage de filtres vidéo à appliquer à l'image avant son affichage. Dans cette partie vous devrez comprendre et gérer les concepts de programmation visuelle associés à la bibliothèque QT. Vous devrez faire en sorte que l'utilisateur puisse choisir parmi tous les filtres développés les éléments qu'il souhaite voir appliqué à l'image, en ajouter, en supprimer ou changer leur ordre d'application afin de rendre plus flexible l'outil.

3 Point de départ de la conception

Le point de départ de votre conception vous sera fourni par votre enseignant lors de la première séance de projet. Le code source qui vous sera distribué assure les fonctions suivantes :

– Création d'une interface graphique permettant d'ouvrir un fichier vidéo et d'en lancer la lecture avec ou sans application d'un filtrage X.

– Mise en marche et arrêt de l'affichage de la vidéo. Mode pas à pas dans le cas ou la vidéo est actuellement arrêtée.
– Passage du mode lecture d'une vidéo au mode extraction d'image depuis une webcam (ou pas...).
– Mode de limitation de l'extraction à 24 images par seconde ou mode libre allant jusqu'à 1000fps au maximum.

Basez vous sur l'ensemble de ce travail afin d'étendre les capacités de l'outil dans le but de répondre à toutes les attentes du client.

4 Outils de développement

Afin de mener à bien ce projet, vous allez devoir changer d'environnement de développement. La bibliothèque objets QT nécessite des scripts de compilation (makefile) que le logiciel Eclipse ne sait pas gérer, nous allons donc nous rabattre sur QtCreator qui est développé par TrollTech (maison mère de Qt). Vous pourrez télécharger une version gratuite des bibliothèques QT, du compilateur g++ ainsi que de l'éditeur à l'adresse suivante :

`http://www.qtsoftware.com/downloads`

5 Classes utiles

La bibliothèque QT offre des centaines de classes toutes plus utiles les unes que les autres ? Ci-dessous je vous en cite un certains nombre qui peuvent vous servir dans le cadre de ce projet, mais beaucoup d'autres sont aussi utilisables (cette liste est non exhaustive) :

– **QString** : Gestion des chaines de caractères
– **QHash** : Structure de données associative (pratique pour le cache)
– **QVector** : Structure de données itérative
– **QFile** : Objet permettant de lire, d'écrire et de vérifier l'existence d'un fichier (**QFileInfoList** + **QFileInfo**),
– **QByteArray** : Object permettant de manipuler des tableaux de données 8 bits en mémoire,
– **QDir** : Object représentant un répertoire (consultation des fichiers présents, vérification de l'existante du répertoire, etc.).

Pour toutes les parties concernant la programmation graphique en Qt vous trouverez de très bon tutoriaux sur Internet, de plus le code source fourni par votre encadrant devrait vous permettre d'en comprendre les bases ?