Cédric Cognard
Paul Van-Elsue

Système mobiles

Compte rendu de TP n°4 et 5 :

Application à un système vidéo

Introduction:

Dans ce TP nous allons apprendre à utiliser la caméra de notre système Android et réaliser divers traitements sur son flux. Nous allons commencer par travailler en Java puis en C++. La librairie qui va nous servir est **OpenCV** car elle est relativement simple d'utilisation et performante.

Afin de pouvoir faire fonctionner l'application sur la tablette il est nécessaire d'y installer **OpenCV Manager**. Pour cela on utilise la commande :

```
adb install -l -r -t -d
/home/polytech/dev/OpenCV-android-sdk/apk/OpenCV_3.1.0_Manager_3.10_armeabi-v7a.apk
```

1. Démarrage avec Eclipse:

Lorsque l'on exécute le code fourni, on constate qu'un filtre est appliqué sur l'image capturée, nous avons une image en nuance de gris. Avec ce code nous avons en moyenne 29,57 frames par secondes. L'application a un cercle de vie relativement précis.

Pour initialiser la vue on appelle *onCameraViewStarted* , puis à chaque frame reçues, elle exécute *onCameraFrame* (pour rendre l'image grise), la fonction est donc exécutée un peu plus de 29 fois en une seconde.

Lorsque c'est nécessaire, *onPause* est appelé et mets l'application en pause (désactive la vue), la fonction *onResume* est utilisée lorsque l'application sort de la pause. Enfin au moment de quitter l'application *onDestroy* est exécuté.

La méthode *onCameraViewStarted* est le listener qui se déclenche au démarrage d'une vue de caméra, tandis que *onCameraFrame* s'exécute à chaque frame reçue.

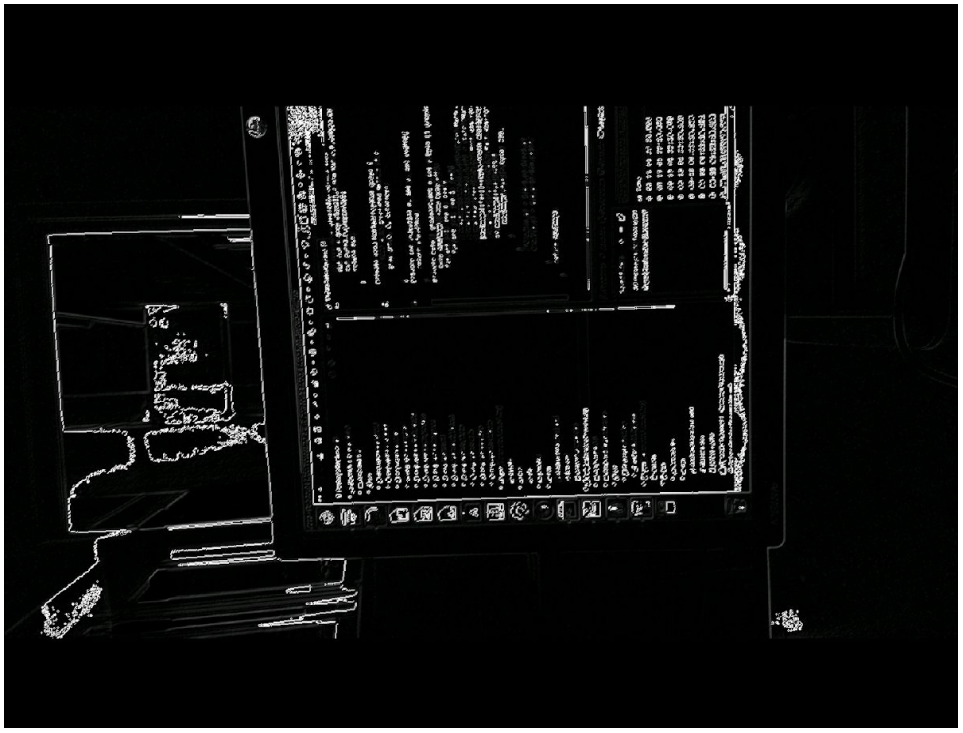
2. Traitement d'images en JAVA:

2.1 Calcul du gradient:

Le gradient permet de rendre les contours plus visibles. Nous avons réalisé ce calcul de la façon suivante:

```
private byte[] Gradiant(byte[] array) {
    int x, y;
    byte gradH, gradV;
    byte[] gradiant_array = new byte[w*h];

    for(x = 1; x < w - 1; x++) {
        for(y = 1; y < h - 1; y++) {
            gradH = (byte) (array[y * w + x - 1] - array[y * w + x + 1]);
            gradV = (byte) (array[(y - 1) * w + x] - array[(y + 1) * w + x]);
            gradiant_array[y * w + x] = (byte) Math.abs(gradH + gradV);
        }
    }

    return gradiant_array;
}
```

Le calcul du gradient vertical et horizontal réduisent les FPS à environ 0.61 en moyenne. En effet le calcul étant plus long, notre tablette HP n'avait pas le temps de faire le traitement rapidement, la vidéo était très coupée.
Nous obtenons le résultat suivant:

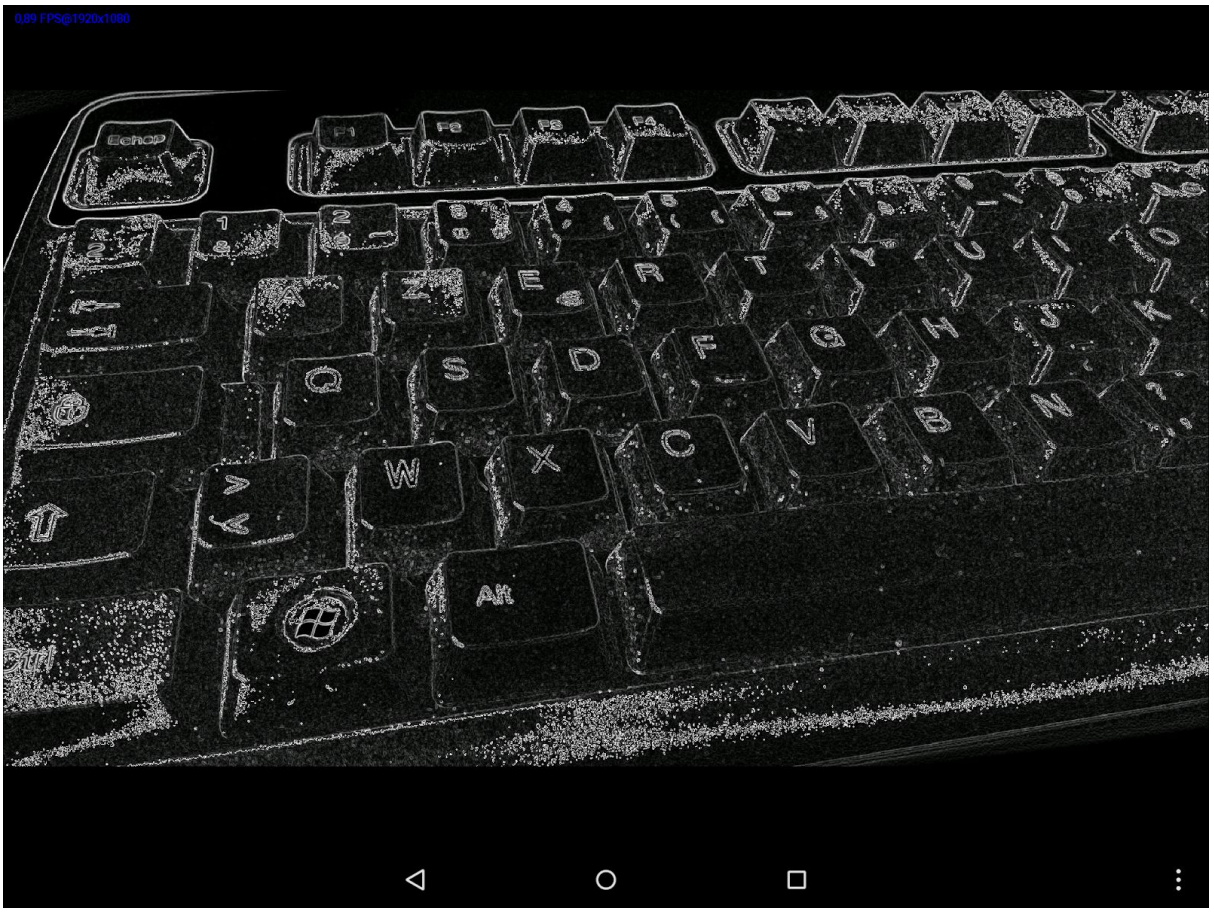


2.2 Filtre de Sobel:

Le filtre de Sobel est un filtre connu, nous avons donc pu trouver diverses documentations en ligne. Ce filtre permet également d'accentuer les contours, mais il le fait de manière plus efficace tout en restant assez simple à implémenter. Nous l'avons fait de la manière suivante :

```
private byte[] Sobel(byte[] array) {
    int x, y, m_x, m_y;
    int filter[][] = new int[3][3];
    byte[] sobel_array = new byte[w*h];

    filter[0][0] = -1;
    filter[0][1] = 0;
    filter[0][2] = 1;
    filter[1][0] = 2;
    filter[1][1] = 0;
    filter[1][2] = 2;
    filter[2][0] = 1;
    filter[2][1] = 0;
    filter[2][2] = 1;

    for(x = 1; x < w - 1; x++)
        for(y = 1; y < h - 1; y++) {
            sobel_array[y * w + x] = 0;

            for(m_x = 0; m_x < 3; m_x++)
                for(m_y = 0; m_y < 3; m_y++)
                    sobel_array[y * w + x] += filter[m_x][m_y] * outarray[(y + (m_y-1)) * w + (x + (m_x-1))];
        }

    return sobel_array;
}
```

Avec le filtre de Sobel, la qualité des contours détectés est bien meilleure mais on arrive à 3 FPS en moyenne, le calcul étant toujours trop long à réaliser. Nous obtenons le résultat suivant:



On peut remarquer que plus le traitement appliqué sur l'image inclus de calculs plus le traitement est long et cela va donc réduire le nombre de frames par seconde. Cependant plus nous avons un grand nombre de calculs plus le résultat obtenu est de bonne qualité.

3. Traitement d'image en C++ :

Dans la suite de ce TP, nous avons essayé de réaliser les mêmes traitements que ceux réalisés précédemment, en changeant le langage utilisé. En effet le C++ ayant pour réputation d'être plus rapide que le Java nous nous attendions donc à obtenir un taux de frames par secondes plus élevé.

Le nom de la fonction dans le code en C est composé du chemin des packages (avec des underscore à la place des points), suivi du nom de la classe et enfin du nom de la méthode qui n'est autre que ProcessFast(). Cette méthode appartient à la classe FrameProcessing. Il est nécessaire de modifier l'entête de la fonction pour que celle ci correspondent à l'architecture du TP4. L'entête final est la suivante:

```
JNIEXPORT void JNICALL Java_fr_polytech_video_VideoActivity_ProcessFast(
```

En effet le code appelant cette fonction est situé dans le package fr.polytech.video. La méthode ProcessFast(...) du code Java va lorsqu'elle sera appelée charger le JNI qui va pouvoir appeler le code de la fonction en C++ ayant le même nom et l'exécuter.

Le problème étant que nous n'avons pas réussi à configurer notre IDE correctement et nous n'avons pas effectué réellement les tests. Même en essayant les programmes de nos camarades, impossible de compiler.

Il est cependant aisé d'imaginer que le code C++ correspondra exactement à une transcription du code Java en C++, mais sera bien plus rapide.

D'après ce que nous avons pu observer chez nos camarades, le calcul du gradient passe désormais à 18 FPS (contre 0.6 avant), ce n'est toujours pas assez mais c'est effectivement bien plus efficace ! Il en va de même pour le filtre de Sobel qui est bien plus rapide.

Conclusion:

Ce TP nous a permis d'avoir une première approche du traitement vidéo sous Android sous deux langages différents. Comme nous le pensions dès le début le code en C++ était plus performant que le code en Java. Cependant il ne faut pas oublier que le manque de performances peut être dû à un défaut de qualité de code, et qu'il faut avant tout savoir optimiser ce que l'on fait avant de chercher à changer de langage. Ce TP nous a aussi permis d'apprendre à "injecter" du code en C++ dans un code en Java, malgré le fait que nous n'ayons pas réussi à configurer notre IDE nous voyons très bien comment cela devrait se passer au niveau du code.