Vision et traitement d'images embarqué
Optimisation des temps de calcul et processeur ARM
Filtre Médian- Sobel (OpenCV/Linux/PC)

IGM_M2-SIS

1. Introduction

1.1 Objectif

L'objectif de ce projet est d'implanter et optimiser le filtre Médian et le filtre Sobel dans une chaîne de traitement d'image en mettant en œuvre les techniques d'optimisations suivantes :

- optimisation algorithmique (algorithme de complexité inférieure au sens mathématique, des structures de données adaptées),
- optimisation pour les processeurs RISC (déroulage de boucle),
- optimisation pour une utilisation efficace des ressources matérielles (accès aux données, gestion de la mémoire).

1.2 Organisation et évaluation

Le travail est à réaliser par équipe de 2 personnes.

Rédigez un rapport qui présentera :

- L'architecture du système utilisé (processeur, fréquence, hiérarchie mémoire, système d'exploitation etc...),
- Les algorithmes et les implantations réalisées,
- Les explications et justifications des choix algorithmiques et d'implantation,
- Les résultats obtenus pendant toutes les étapes du développement du projet (images, copie écran, …),
- Les graphes, courbes et tableaux montrant les résultats de profilage et les gains de temps d'exécutions en % pour chaque opérateur et l'ensemble de l'application,
- En annexe, le code développé avec les commentaires.

2. Mise en œuvre du projet

1. Avant de commencer il faut créez un répertoire de travail local sur votre machine au niveau de votre compte **« /home/…/nom_utilisateur/ »**.
2. Téléchargez le fichier source « RK_Sabre.cpp », ce fichier servira comme version de base pour votre développement. Ce projet est déjà paramétré pour utilisation de la librairie OpenCV.
3. Sous Linux, la compilation de votre projet est assurée via gcc :

```
g++ `pkg-config opencv --cflags` RK_Sabre.cpp -o project `pkg-config opencv --libs`
```

3. Travail à réaliser

3.1 Opérateur à implanter

Pour réaliser la détection de contours à partir d'un flux vidéo, il s'agit d'implanter et optimiser les opérateurs :

- Filtre Médian,
- Filtre de Sobel.

Note : Les étapes d'acquisition et d'affichage des images sont assurées avec les fonctions OpenCV et sont fournies. L'opérateur de conversion en niveaux de gris est également fourni dans la version de base du projet « RK_Sabre.cpp ».

3.2 Étapes du développement à respecter

a. Préparation théorique

- Étudiez les ressources et caractéristiques matérielles du système que vous utilisez : fréquence de fonctionnement du processeur, caractéristiques (nombre des unités de calculs et leurs types), système de mémoire (taille, hiérarchie).
- Pour les opérateurs à implanter, estimez le nombre théorique d'opérations par pixel et le nombre d'opérations par image. Estimez le nombre d'accès aux données par pixel et par seconde, en fonction de la cadence souhaitée (ex. 30 images par seconde) et la taille d'image donnée.

b. Implantation et Optimisation du filtre Sobel

- Réalisez une première version sans optimisation de la chaîne de détection de contour sur PC.
- Portez la même version **« basique »** sur la carte SABRE Lite i.MX 6Quad development kit built to Freescale® SABRE Lite design ;
- Effectuez des mesures des temps d'exécution pour se comparer avec les performances obtenues sur le PC.
  https://www.element14.com/community/community/designcenter/single-board-computers/sabrelite
- En utilisant l'outil « **gprof** », évaluez et interprétez les résultats obtenus.
  https://sourceware.org/binutils/docs/gprof/
- Evaluez et interprétez les résultats en exécutant le programme compilé avec différentes options de compilation (O1, O2 ou O3), afin d'assurer les meilleures performances – cherchez à comprendre quelles techniques d'optimisation sont utilisées par le compilateur pour optimiser le code.
- Afin d'accélérer les temps de calcul, développez différentes versions d'implantation des filtres, en apportant à chaque fois une nouvelle technique d'optimisation (optimisation

algorithmique, déroulage de boucle, réorganisation du code, optimisation des accès aux données, gestion de la mémoire cache, multi-thread, …).

- Pour chaque version :
  - Evaluez le nombre d'opération par image, évaluez le nombre d'accès aux données.
  - Effectuez les mesures des temps d'exécution de chaque opérateur; créez un tableau de profilage qui mettra en évidence le temps de chaque opérateur et le pourcentage de ce temps par rapport à l'ensemble de la chaine de traitement.
    https://fr.wikiversity.org/wiki/Fonctions_de_base_en_langage_C/time.h
    https://software.intel.com/fr-fr/articles/timer-temps-execution-C

Note : Pour chaque étape de développement, ne pas oublier de vérifier que le résultat obtenu est correcte (par exemple comparer les résultats obtenus avec ceux des fonctions OpenCv).