Gprof

Gprof ne vous permet pas d'optimiser le code; C'est un outil de profilage, qui vous permet de mesurer les performances de votre code et de vous indiquer la ou les fonction(s) la/les plus coûteuse(s) dans votre programme (en terme de complexité de calcul, temps d'exécution ou accès mémoire).

pour l'utiliser, il faut :

Autoriser le profilage, en recompilant avec l'option -pg du gcc;

>> g++ -pg `pkg-config opencv --cflags` RK_Sabre.cpp -o RK_Projet `pkg-config opencv --libs`

Exécuter votre programme. Un fichier gmont.out sera créé.

Afin d'accéder aux informations du profiling, il faut analyser ce fichier avec gprof

>> gprof RK_Projet | less

Pour enregistrer les résultats de profilage dans un fichier texte, vous pouvez utiliser cette option,

>> gprof RK_Projet > resultat_profiling.txt

Vous pouvez également affiner l'affichage des résultats en utilisant les options de gprof :

- a suppression des fonctions statiques

- b supprime « verbose » mode

- p supprime « flat profile »

- etc...

Attention, Les résultats obtenus par une seule exécution ne sont pas statistiquement représentatifs. Afin de les préciser, il faut répéter cette exécution plusieurs fois et analyser la moyenne des performances obtenues (enregistrées dans différents fichiers)

Pour pouvoir évaluer chaque partie de votre code, il faut créer des sous fonctions (Acquisition, NG, Median, Sobel, Affichage, ...); sinon l'analyse sera infructueuse, le code évalué utilisera chaque fois 100% du temps d'exécution global et vous ne saurez pas quelle partie est à optimiser en un premier temps.

Pour plus de détails, je vous ai communiqué ce lien dans le sujet :
https://sourceware.org/binutils/docs/gprof/

Sinon, plusieurs tutoriaux existent sur le net, par exemple :
http://www.linuxfocus.org/Francais/March2005/article371.shtml

Rostom Kachouri

Si vous le souhaitez, l'outil Kprof vous permet d'afficher graphiquement les résultats obtenus avec gprof : http://kprof.sourceforge.net/

Sachant que gprof n'est pas le seul outil de mesure de performance, vous pouvez toujours utiliser d'autres outils (Sys/time, perf, ...) ou plusieurs outils en même temps afin de comparer les différents résultats de profilage obtenus.

Il est à noter que la prise en main de ces outils et l'installation des packages nécessaires si besoin font partie du projet.

Toutefois, si vous rencontrez des problèmes d'utilisation et/ou d'installation sur PC ou sur carte, il est toujours possible d'utiliser la librairie time.h dont le lien est également fourni dans le sujet : https://fr.wikiversity.org/wiki/Fonctions_de_base_en_langage_C/time.h

il suffit dans ce cas de :

- inclure cette bibliothèque à votre code #include <time.h>

- déclarer les variables suivantes :

```
long clk_tck = CLOCKS_PER_SEC;

clock_t t1, t2;
```

- utiliser la commande t1 = clock(); juste avant la partie du code dont vous voulez mesurer le temps d'exécution pour récupérer le temps initial en "clock ticks"

- utiliser la même commande t2 = clock(); juste après pour récupérer le temps final

- le temps d'exécution de la partie évaluée est alors : (double)(t2-t1)/(double)clk_tck);

NB. Afin de pouvoir effectuer des comparaisons entre les différentes versions optimisées, il est nécessaire d'avoir des séquences identiques, alors n'oublier pas de modifier votre code en choisissant un nombre de trames fixe à traiter dans votre boucle de traitement d'image et ne pas continuer avec l'option quitter en tapant 'Q' ou 'q'.