Documentation du projet

Projet analyse d'image PGM en C

Organisation du projet :
Le projet est organisé en plusieurs parties :

- Un dossier "src" qui contient le code permettant la manipulation des images PGM (charger l'image, la modifier, sauvegarder le résultat…) et également l'algorithme permettant de vérifier la conformité des anneaux (dans analysePGM.c).
- Les dossiers "test" et "lib" qui permettent d'effectuer des tests unitaires lancés par le makefile (la librairie utilisée est Unity plus de détails dans la partie Tests unitaires ci-dessous)
- Le dossier "doc" qui contient toute la partie documentation du projet (il y a aussi de nombreuses explications dans les commentaires présents dans le code du projet)

1) Traitement de d'images PGM

Cette partie qui correspond au fichier "pgmImageProcessing.c" dans le dossier "src" permet plusieurs choses :

- Importer l'image pour la stocker dans une matrice
- Pouvoir effectuer des modifications sur cette image (certaines ne sont pas forcément utiles au projet comme la rotation par exemple que j'ai implémentée pour m'entraîner mais la détection des bords peut être utile pour faciliter le travail de l'algorithme)
- Vérifier que l'image est conforme avant d'essayer d'appliquer l'algorithme (si l'image pgm a 4 lignes d'entêtes par exemple cela peut faire échouer le programme)
- Sauvegarder l'image résultant des traitements que l'on lui a appliqué (cela a notamment été utile au pour surveiller que tout fonctionne correctement)

2) Tests unitaires et Makefile

Cette partie permet de vérifier que les conditions sont réunies pour que le programme fonctionne, en cas d'échec d'un test on peut plus facilement identifier l'erreur et la corriger surtout lorsque l'on fourni le programme à quelqu'un d'autre et donc que les conditions d'exécution de celui-ci sont différentes. Pour cela j'ai utilisé la librairie Unity : http://www.throwtheswitch.org/unity qui est simple d'utilisation pour réaliser rapidement des tests unitaires en C. Le makefile est très complet et permet d'effectuer la compilation, les tests et le nettoyage des fichiers compilés.

3) Algorithme pour analyser l'image

Le cœur du projet et ce qui permet donc de définir si les anneaux sont conformes ou non. Il est important de distinguer deux parties :

- La localisation des anneaux avec la fonction findCircle() dans analysePGM.c, c'est un élément essentiel de l'analyse de l'image puisque c'est cette fonction qui trouve les anneaux et les permet ensuite le traitement pour déterminer si ils sont conforment ou non. En principe : on parcours l'image de la gauche vers la droite et lorsque l'on arrive au bout on descends d'une ligne, ainsi on est sûr que le premier pixel noir que l'on trouve soit le pixel qui dépasse au sommet du cercle. En sachant cela on peut découper un carré de 65*65 pixels qui contient le cercle pour le comparer avec une image de référence et le carré découpé est remplacé par des pixels blancs sur l'image originale pour ne pas traiter à nouveau cet anneau lorsque l'on poursuivra la recherche.
- La détermination de la conformité en deux étapes : mandatoryTests() qui va vérifier des propriétées telles que si à partir du haut de l'anneau les 16 pixels suivants sont bien noirs, entre 48 et 64 pixels dessous il y a du noir et au milieu et après 64 pixels en dessous il n'y a plus de noir. comparerImages() qui va calculer la différence entre chaque pixels de l'anneau avec un anneau conforme pour model et si la somme des différences est inférieure à une valeur définie arbitrairement alors on considère qu'il est conforme.
  Calcul de la différence entre deux matrices de pixels :

$$d(M, M') = \sum_{i,j} |x_{i,j} - x'_{i,j}|$$

Résultat du programme :
Il peut détecter plusieurs anneaux conformes, ne fonctionne pas si ils sont trop proches mais sinon le programme est assez fiable même si il y a encore de quoi l'améliorer. (il est aussi capable de donner des indications sur l'erreur).