

Rapport du Projet d'image

Reconnaissance et indexation
de forme

Patrice Coudert et Frédéric Lang

Contents

Introduction et structure du rapport

Le principe du projet est de créer une signature. Pour cela nous avons créé un certain nombre d'indicateurs que nous allons tout d'abord vous présenter. On en profitera pour s'intéresser à leur robustesse vis à vis de l'occlusion, du bruit, ... Ces propriétés seront étudiées à la fois théoriquement et en pratique.

Pour les résultats pratiques, les scripts bash et les graphiques sont présentés dans le dossier pour reproduire ces tests. Même si les graphiques ne seront pas tous dans le rapport, ils sont présents dans le dossier résultat. Les scripts bash servent à reproduire les tests que nous avons effectués.

La suite expliquera comment nous utilisons ces indicateurs pour répondre aux questions. Nous proposons deux options: la première renverra la classe la plus probable et la deuxième renverra les k images les plus proches.

Explications du programme

Nous allons commencer expliquer comment utiliser notre programme. Le programme s'appelle *main*. Il prend deux options. La première est le mode du programme. Les différentes options sont:

- help*: qui affiche les options et les explications de ce qu'elles font.
- similn*: qui renvoie les n images les plus proches au sens de notre distance.
- indn*: qui renvoie l'indicateur n pour l'image en entrée.
- disp*: qui crée un fichier *image.eps* qui montre nos constructions. (voir ci-après pour plus de détails)
- dispDT*: qui crée un fichier *image.eps* qui montre les cercles où la distance au bord est constante.

Pour utiliser notre programme, il faut donc taper le nom du programme, puis le mode d'exécution souhaité, puis si vous voulez une des options d'affichage et enfin le nom du fichier. Voici trois exemples:

```
./main - ind1 - dispapple - 1.pgm  
./main - ind3 - dispDTapple - 1.pgm  
./main - simil8apple - 1.pgm
```

Chapter 1

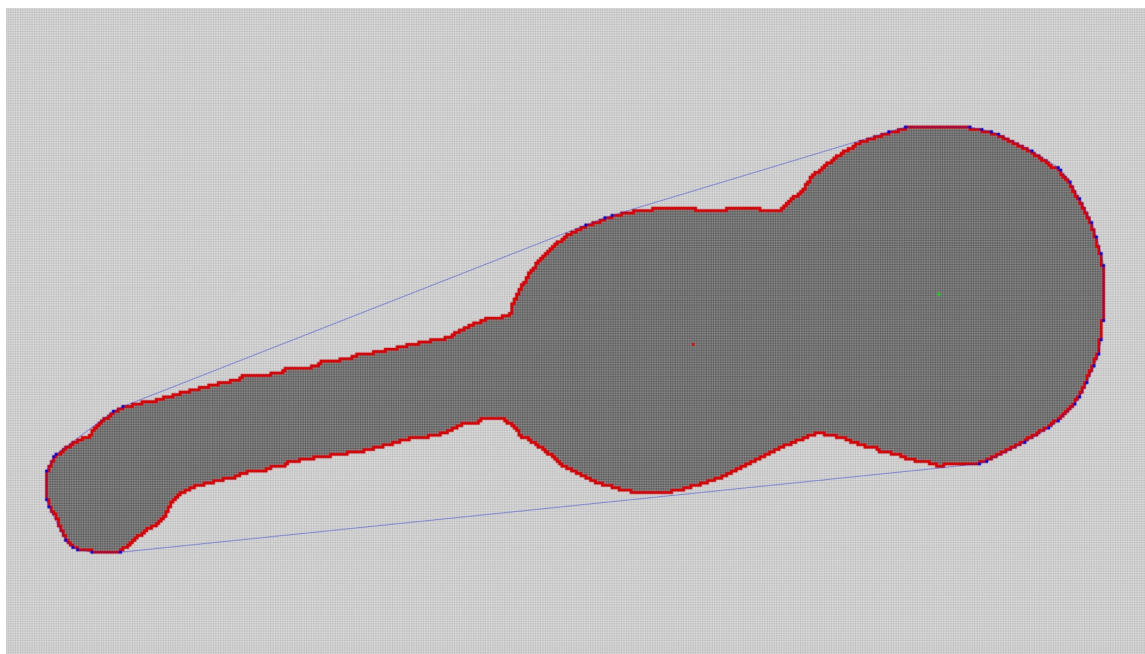
Détails des indicateurs

Introduction

Les indicateurs que nous avons choisi essayent de faire ressortir les caractéristiques principales de l'image. Après avoir regarder les images, on a décidé de grandeur caractéristiques simples et assez différentes selon les classes. À partir de la sortie graphique du programme, la cohérence des indicateurs peut être vérifiée visuellement ce qui nous permet de vérifier facilement nos programmes. Les option *-disp* et *-dispdt* permet de réaliser ces vérifications visuels.

Nous avons fait le choix de nous intéresser qu'à la composante connexe principale c'est à dire de plus grande taille. Les composantes connexes sont calculées en topologie 4. On donne un exemple du changement d'image qu'impose notre choix sur *devie1 - 9.pgm*. La composante principale est en noir et les composantes supprimés sont en jaune. La raison de ce choix est double. Premièrement on enlève des taches qui ne sont pas très caractéristiques de l'image. En effet ces taches peuvent résulter du bruit de notre image ou sont de taille minime et pas caractéristique de la classe. Deuxièmement nos fonctions qui calcule le contour et l'enveloppe convexe ne sont pas résistant à ces problématiques. On discute des améliorations possibles liés à ceci dans le section limitation et améliorations possibles.

L'image ci-dessous est le résultat de notre algorithme pour l'image *guitar - 1.pgm*. Le contour de l'image est en rouge. Celui-ci est pris en topologie 4. Le contour de l'enveloppe convexe apparaît en bleu. Le barycentre apparaît en bleu et le point où la distance au bord est maximal en vert. Ces résultats sont obtenus en ajoutant l'option *-disp* à n'importe quel image. La deuxième image est obtenu sur la même image avec la commande *-dispDT* est montre les cercle où la distance au bord est constant.



Nous allons présenter un par un les dix indicateurs que nous avons choisi. Un script est présent pour calculer un indicateur sur toutes les images de la base de données. Pour l'utiliser il suffit de le lancer avec le numéro de l'indicateur. Pour chacun nous allons ensuite justifier théoriquement et en pratique de leur résistance/sensibilité au bruit, à un changement d'échelle et à une rotation. Toutes ces sessions ne sont pas développées en détail. Mais, pour chaque indicateur, trois tests ont été réalisés grâce aux scripts *resistance* et les courbes sont dans le dossier correspondant. Pour refaire ces tests, il suffit (après avoir compilé le programme) d'exécuter le script bash par la commande suivante:

```
bash resistance_bruit1.sh apple 3 2
```

Cette commande lance vingt tests sur l'image *apple - 3.pgm* pour regarder l'influence du bruit pour l'indicateur 2. Ensuite pour tracer les courbes il suffit d'utiliser le script *gnuplot* dans le dossier *resultat/bruit/*. Dans ce fichier, il suffit juste de changer le nom du fichier dont vous souhaitez la courbe.

1.1 PERIMETRE

Cet indicateur mesure la longueur du contour de l'image. Il est destiné à différencier les formes très courbées (*lizzard*) de celle qui se rapproche plus d'un disque (*apple*). Il est calculé en extrayant le contour de la forme avec la fonction *contour* implémentée dans le fichier *utils.cpp*. Cette fonction est celle qu'on a codée lors du TD5. Le contour obtenu correspond à une topologie 4. Pour ne pas le rendre sensible à un changement d'échelle, on divise par la racine de la taille de l'image.

Resistance au bruit

Le contour est par nature assez sensible au bruit. En effet, le bruit se situant majoritairement au bord, c'est le contour qui est le premier modifié. Pour limiter ce problème, on ne garde que la plus grande composante connexe en topologie 4. Ainsi, même si on s'attend à ce que la taille du contour augmente, le fait de supprimer les composantes connexes en topologie enlève un peu le bruit et donc la taille du contour reste acceptable. Les test pratique confirme ce fait on voit un augmentation assez nette de notre indicateur.

Resistance à un changement d'échelle et à une rotation

L'orientation de notre figure n'a que très peu d'impact sur notre contour. Lorsque la taille de l'image est assez grande, tourner l'image conserve la taille du contour même si à certain endroit particuliers, le contour varie un tout petit peu. les résultats pratiques confirme ce fait. De même un changement d'échelle n'influe pas sur cet indicateur du fait de la division par la racine carré de la taille de l'image.

1.2 DISQUECIRCONS

: Cet indicateur nous est venu à l'idée lorsque on a vu les classes hammer, lmfish, glass, ... et les classes fish, apple, ... La longueur de l'image par rapport à sa taille est très différent. A Partir de la position du barycentre et de la distance du point le plus éloigné de celui-ci, calculé par les fonctions **barycentre** et **distFarthestPoint** dans **utils.cpp**, on calcule le rapport entre la surface du disque circonscrit à la forme et la surface de de la forme.

1.3 NBSEG

: Le nombre de segment qui délimitent l'enveloppe convexe de l'image. Les points qui relient ces segments sont calculé par l'algorithme de Graham implémenté par la fonction **convexHull** dans **convexhull.cpp**. Pour la première étape de l'algorithme où sont triés les points selon un ordre lié à un point **p0**, le tri n'a d'abord pas fonctionné car des points étaient alignés avec **p0**, mais ce problème a été corrigé en prenant en compte la distance avec celui-ci.

1.4 MAXSEG

: Cet indicateur mesure la taille du plus grand segment délimitant l'enveloppe convexe.