TRAITEMENT D'IMAGES

TP6 : Chaîne complète de traitement d'images

réalisé par LE Viet Man - promotion 15

FONCTIONNEMENT DU PROGRAMME

Compilation des programmes

Pour compiler le programme, il faut modifier la variable INCLUDEDIR dans le fichier Makefile dans le répertoire du programme pour mettre le bon chemin pour le répertoire de OpenCV.

Ensuite, dans le répertoire du programme, vous tapez la commande `make clean`. Et puis, vous tapez la commande `make` pour compiler le programme.

Usage du programme

Le programme possède la commande suivante :

```
./segment <filename>
```

où :

- `<filename>` : le chemin et le nom du fichier d'image

Le programme enverra des images qui contiennent un tel objet segmenté dans l'image originale. Ses noms fini par «_obj#.png» où # : l'indice de l'objet.

Le programme donnera aussi des étapes d'exécution du programme et le nombre des objets segmentés sur la console.

Exemple

- Segmenter l'image objets1.jpg :

```
./segment objets1.jpg
```

CHAÎNE DE TRAITEMENT COMPLÈTE

Comme toutes les autres applications complètes de traitement d'images, mon programme possède trois étapes principales :

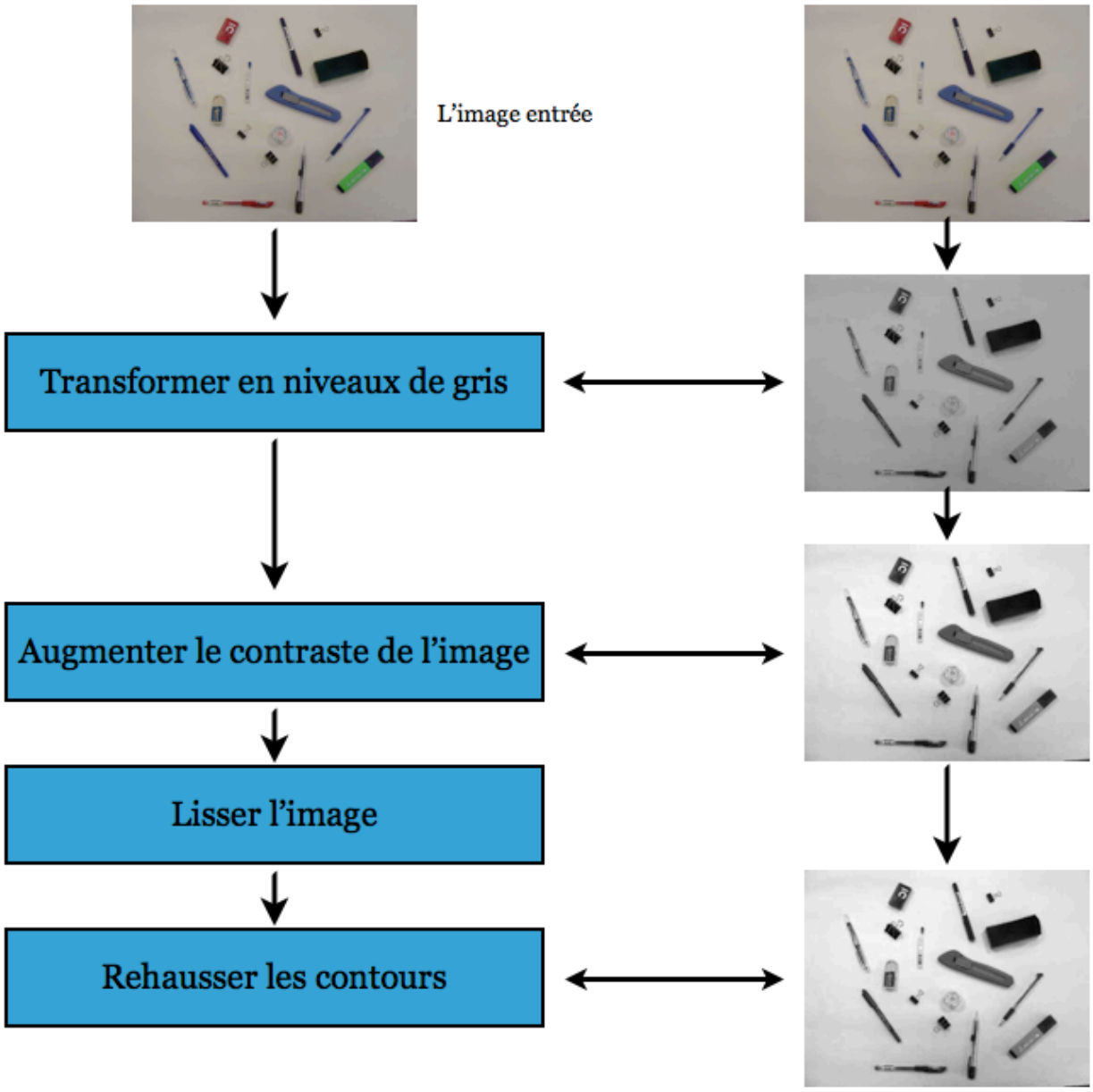
- **Pré-segmentation** : avoir pour but de préparer l'image avant la segmentation. Dans mon programme, je transforme tout d'abord l'image originale en l'image en niveaux de gris. Ensuite, l'image est modifiée automatique le contraste en se basant sur la plus haute valeur de gris dans l'histogramme de l'image. Pour supprimer les bruits, je lisse l'image en utilisant un filtre Gaussien. Enfin, je rehausse les contours dans l'image.
- **Segmentation** : servir à segmenter l'image en des régions isolées.
- **Post-segmentation** : l'image passera des traitements différents pour parfaite les objets segmentés. Cette étape de mon programme a des traitements suivants : effacer les bruits, effacer le fond de l'image, convertir le fond blanc en le fond noir, effacer les bruits au coin de l'image, remplir les trous, éroder les objets, l'étiquetage, créer les masques et prendre les objets segmentés.

Pré-segmentation

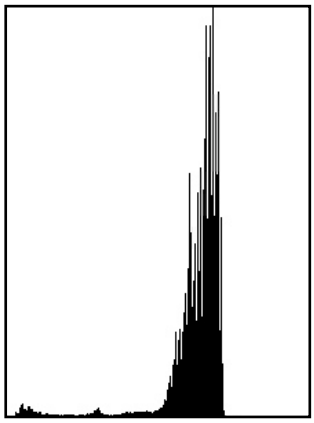
Tout d'abord, j'ai quelques remarques avec quatre images entrées suivantes :

- Toutes les images ont moins de contraste. Donc, ne pas pouvoir détecter tous les contours des objets.
- L'histogrammes des images ne sont pas égalisés et ont la même forme.
- Quand on utilise la fonction cvEqualize de OpenCV avec ces images, les résultats ne sont pas bons et influence le résultat de l'étape Segmentation.
- Plusieurs objets ont la même couleur du fond.
- Le fond de l'image a deux parties : celle claire et celle sombre. Celle sombre a la couleur qui est semblable à l'ombre des objets.
- L'image objets1.jpg a un bruit au coin en bas à droit.

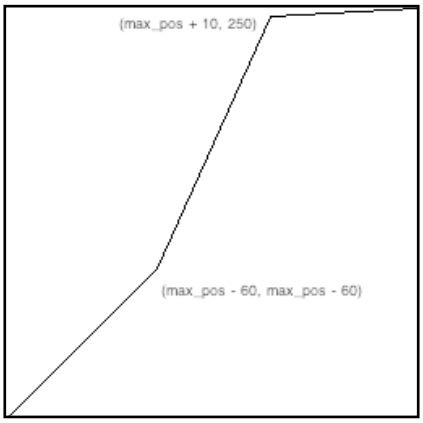
Donc, la tâche de cette étape est d'obtenir le bon contraste de l'image et de clairs contours. Cette étape a quatre phrases suivantes :



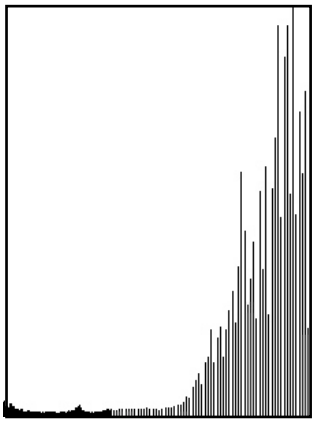
- *Transformer en niveaux de gris* : simplement, j'utilise cvConvertImage de OpenCV avec le paramètre `CV_BGR2GRAY`.
- *Augmenter le contraste de l'image* : pour le faire, j'ai utilisé mon code du TP2 avec quelques modifications. Tout d'abord, je détermine la plus haute valeur de gris (max_pos) dans l'histogramme de l'image. Et puis, je calcule deux points de la fonction de transformation qui ont les coordonnées (max_pos - 60, max_pos - 60) et (max_pos + 10, 250). L'image transformée aura meilleur contraste que le résultat de la fonction cvEqualize et cvNormalise de OpenCV.



L'histogramme de l'image originale



La fonction de transformation



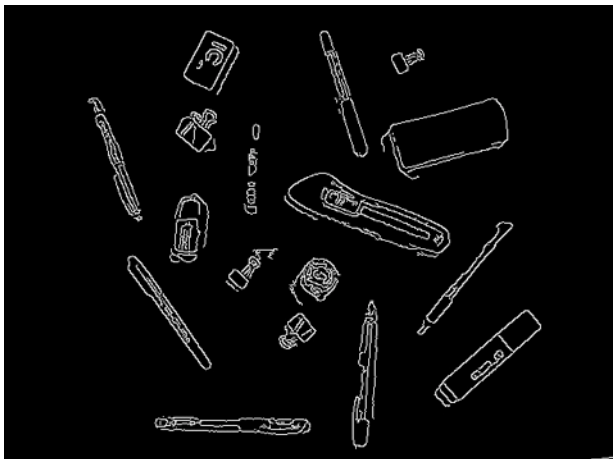
L'histogramme de l'image transformée

- *Lisser l'image* : mais dans le fond de l'image transformée, les nouvelles régions apparaîtra. Pour supprimer ces régions, j'ai utilisé un filtre Gaussien.
- *Rehausser les contours* : Pour détecter efficacement tous les contours des objets, j'ai utilisé un masque de convolution pour rehausser les contours.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Segmentation

Tous les trois algorithmes de segmentation que j'ai évalué dans le TP5, ne sont pas bons pour ces quatre images. L'algorithme k-moyenne et Pyramide segmente une image en se basant sur les couleurs. Donc, ils ne détecteront pas bien des objets. Quand on détermine le grand nombre des régions, le fond de l'image divisera en plusieurs régions. Quand on détermine celui petit des régions, on ne détecte pas quelques objets (ex : la boîte blanche, la coupe-papier, le label jaune du stylo, etc.). L'algorithme watershed ne supporte pas l'image en niveaux de gris. Donc, j'ai utilisé la méthode de détecter des contours pour segmenter des régions. C'est une bonne méthode, mais il existe la redondance des contours ou le manque des contours. Le manque des contours est acceptable parce que l'on peut les restaurer dans l'étape Post-segmentation.

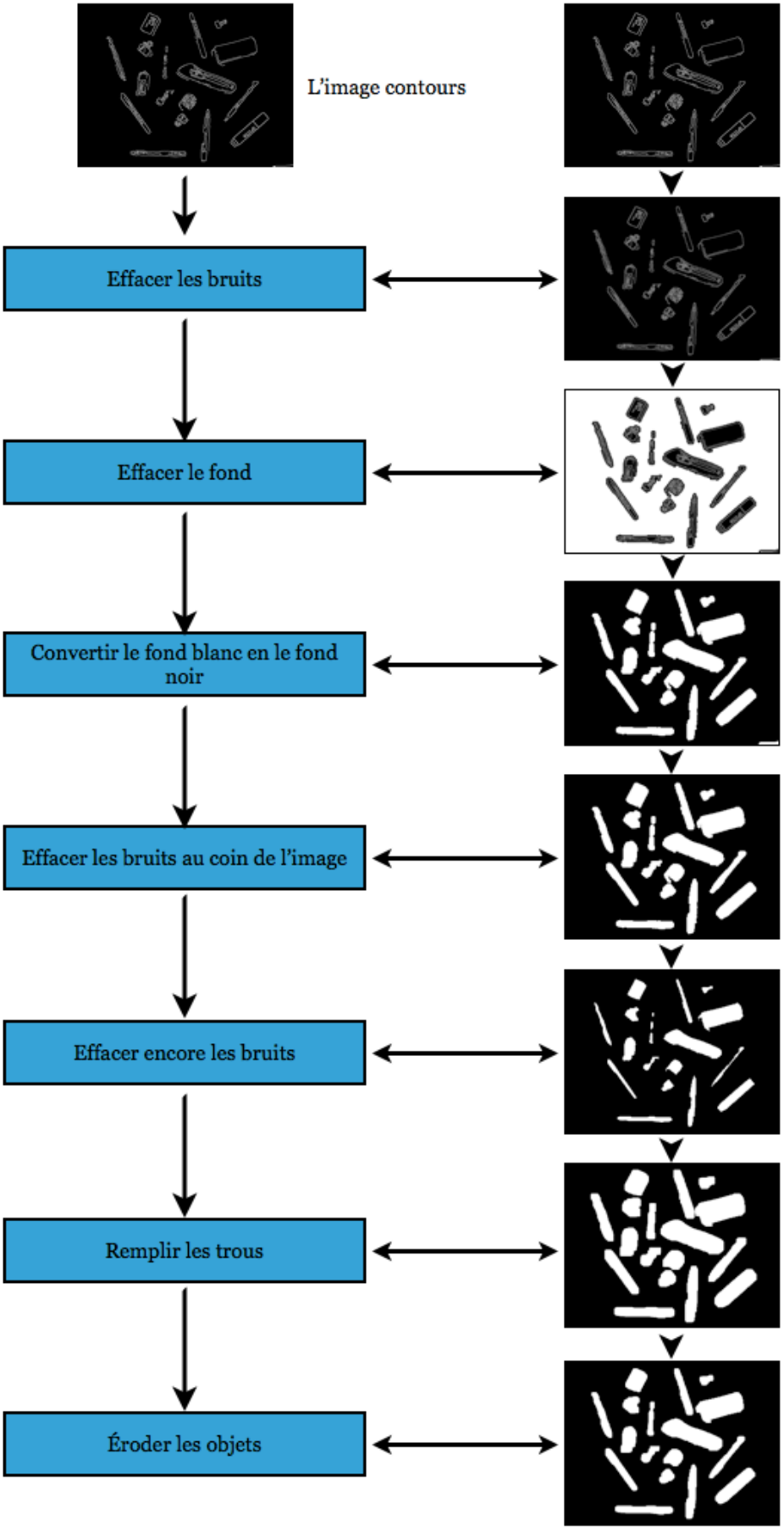


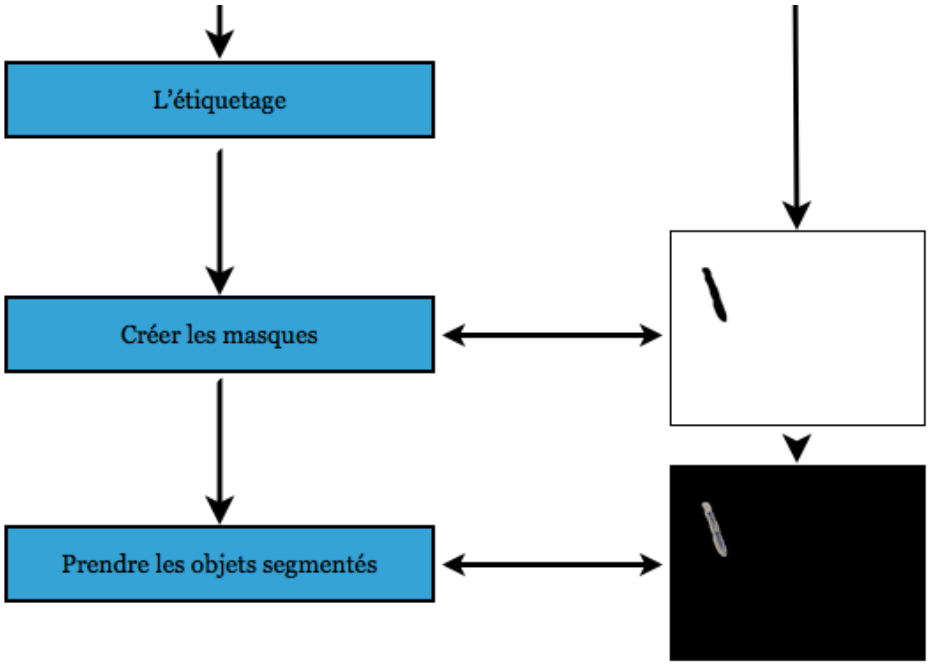
L'image objets1.jpg après l'étape Segmentation

Selon mes tests, la méthode de détection des contours Canny avec les seuils 0 et 160 est la meilleur méthode pour détecter assez les contours nécessaires. Les parties à l'intérieur des contours seront les régions des objets.

Post-segmentation

Cette étape a 10 phases :





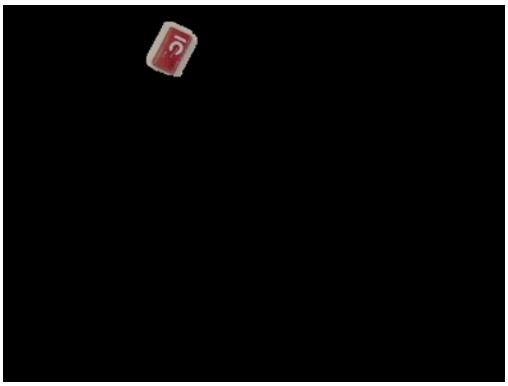
- *Effacer les bruits* : Le résultat de l'étape Segmentation n'est pas encore bon. Les contours ne sont pas fermés. Donc, j'utilise deux fonctions cvPyrDown et cvPyrUp. En effet, l'effet de ces deux fonctions est de supprimer les bruits. Mais il y a encore un autre effet, c'est lier les contours.
- *Effacer le fond* : Après les contours sont fermés, je sépare le fond et les objets en colorant le fond blanc en utilisant la fonction cvFloodFill.
- *Convertir le fond blanc en le fond noir* : après cette phase, les objets sont blancs et le fond est noir.
- *Effacer les bruits au coin de l'image* : dans quelque cas, il existe des régions redondantes qui sont au bord de l'image (surtout l'image objets1.jpg). Donc, j'ai supprimé les régions en utilisant aussi la fonction cvFloodFill.
- *Effacer encore les bruits* : les objets ont encore les pixels redondants que l'on peut seulement supprimer par la méthode de l'érosion. Donc, j'ai exécute l'érosion trois fois.
- *Remplir les trous* : pour le faire, j'ai exécute la dilatation six fois.
- *Éroder les objets* : enfin, j'ai exécuté l'érosion deux fois. Les résultats sont les objets isolés.
- *L'étiquetage* : pour le faire, j'ai utilisé la fonction cvFloodFill.
- *Créer les masques* : correspondre à chaque région segmentée, je crée un masque dont le fond est blanc et les objets sont noirs.
- *Prendre les objets segmentés* : en se basant sur les masques, je prends les objets.

RESULTATS

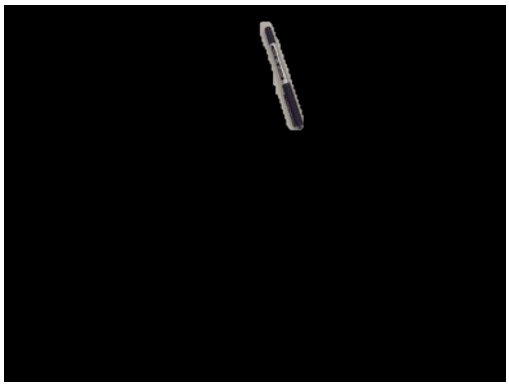
L'image objets1.jpg

On obtient 17 objets suivants :

L’image objets1.jpg

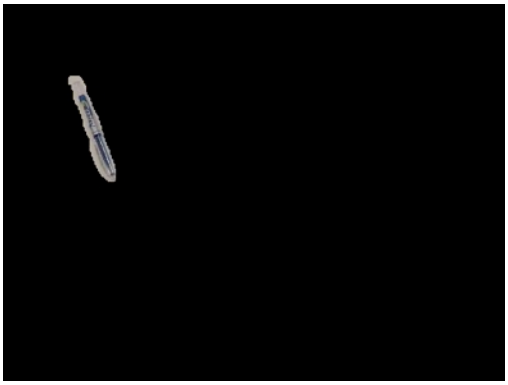


L’objet 1

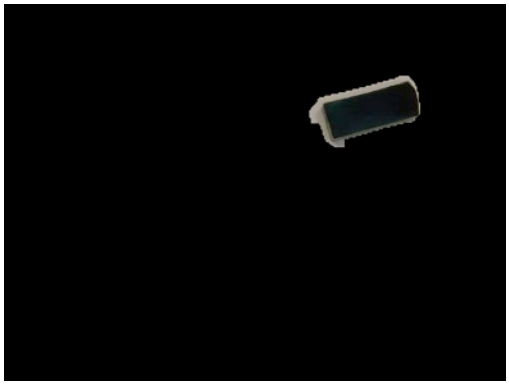


L’objet 2

L’objet 3

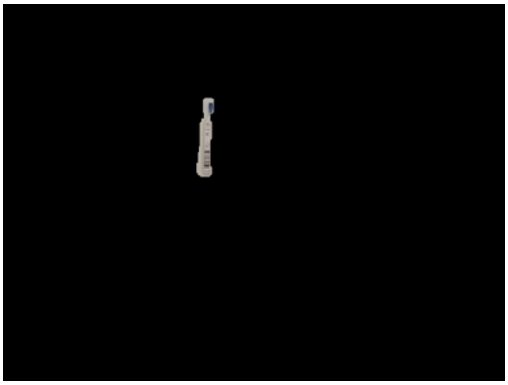


L’objet 4

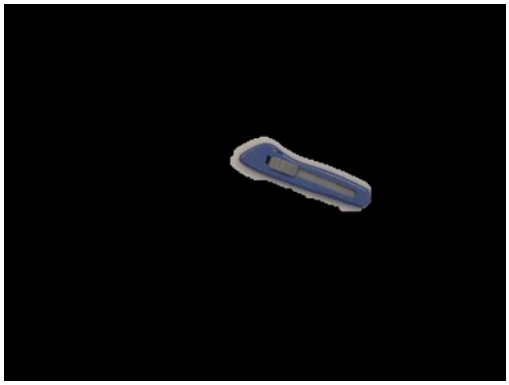


L’objet 5

L’objet 6

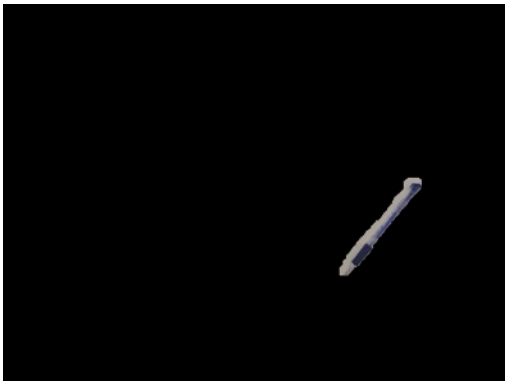


L’objet 7

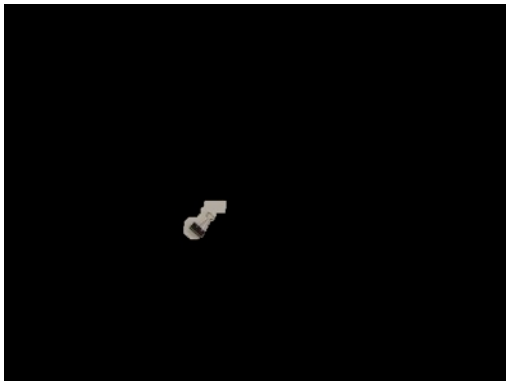


L’objet 8

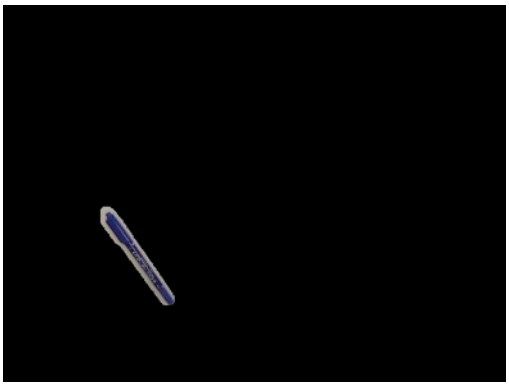
L’objet 9



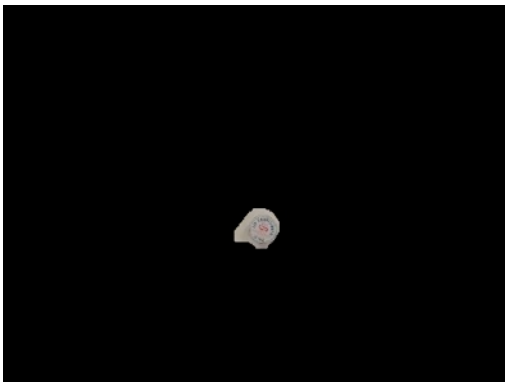
L’objet 10



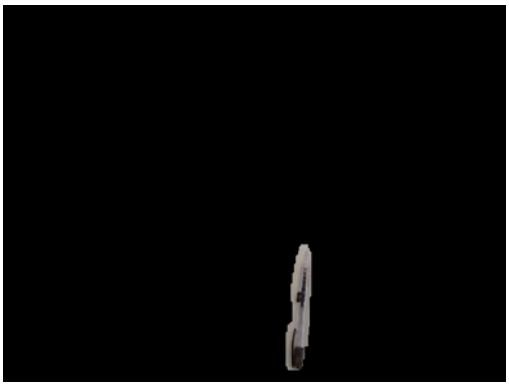
L’objet 11



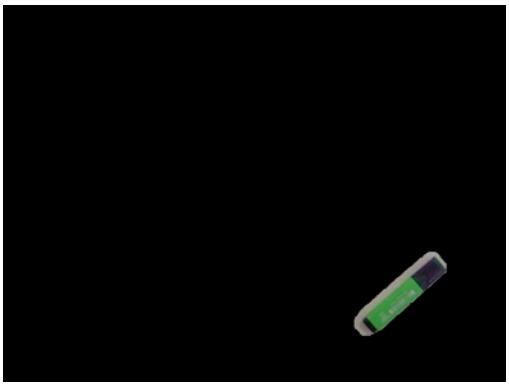
L’objet 12



L’objet 13

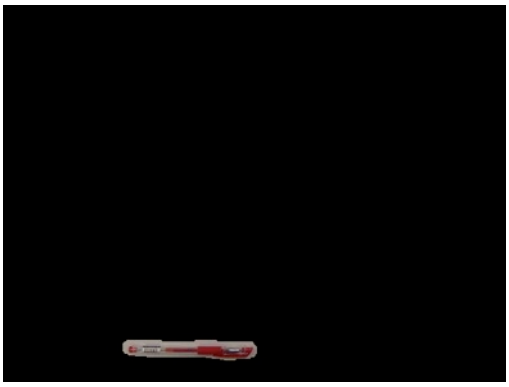


L’objet 14



L’objet 12

L’objet 12

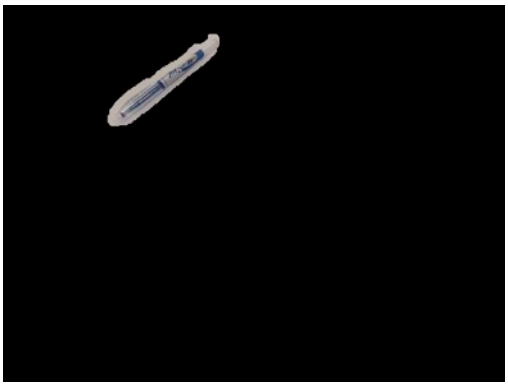


L’objet 12

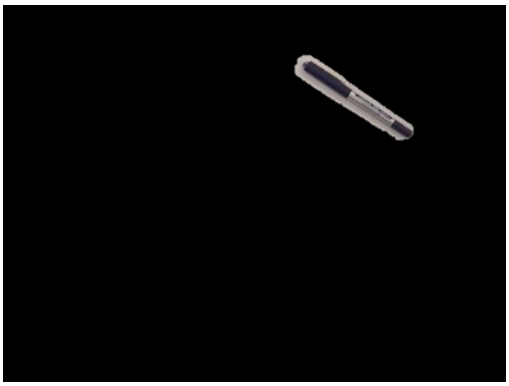
L’image objets2.jpg

On obtient 9 objets suivants :

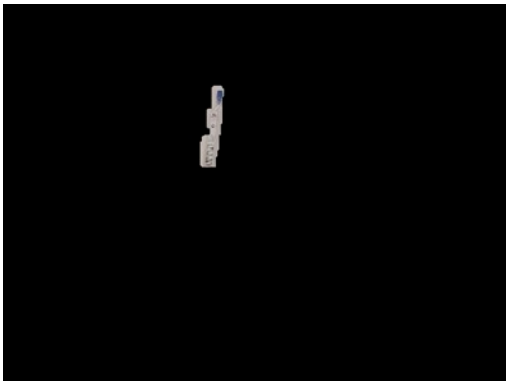
L’image objets2.jpg



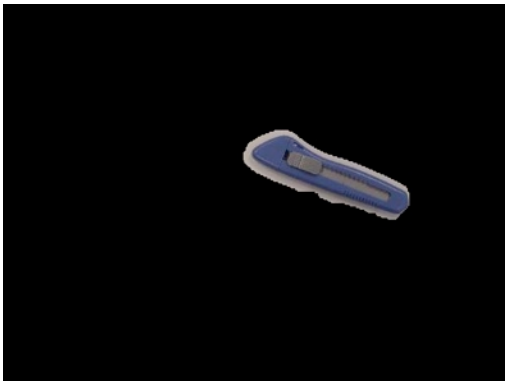
L’objet 1



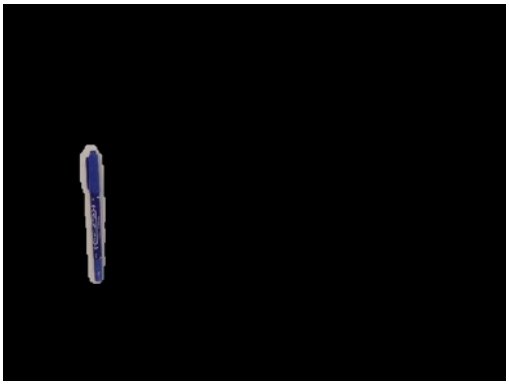
L’objet 2



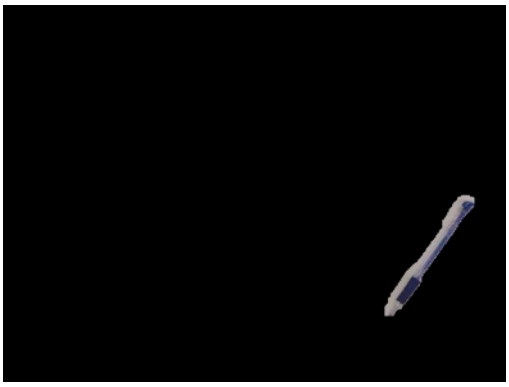
L’objet 3



L’objet 4



L’objet 5



L’objet 6

L’objet 7

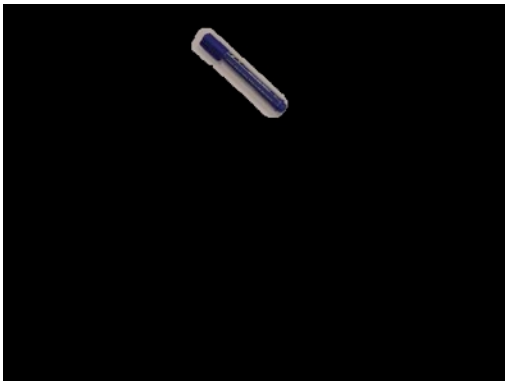
L’objet 8

L’objet 9

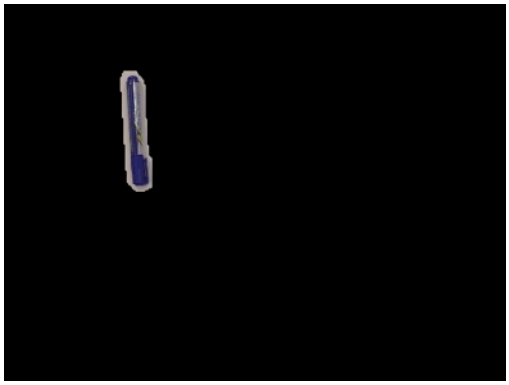
L’image objets3.jpg

On obtient 11 objets suivants :

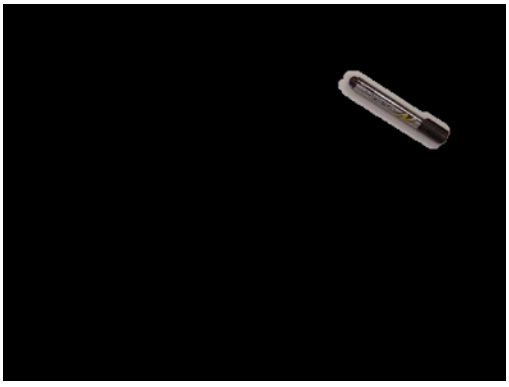
L’image objets3.jpg



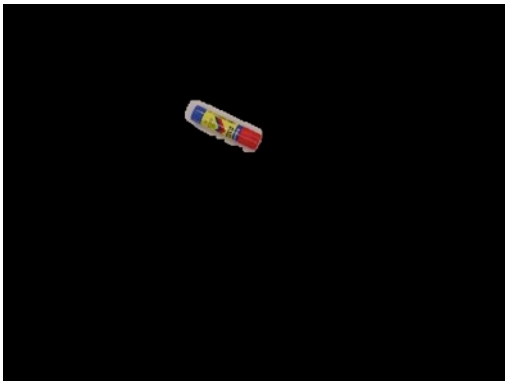
L’objet 1



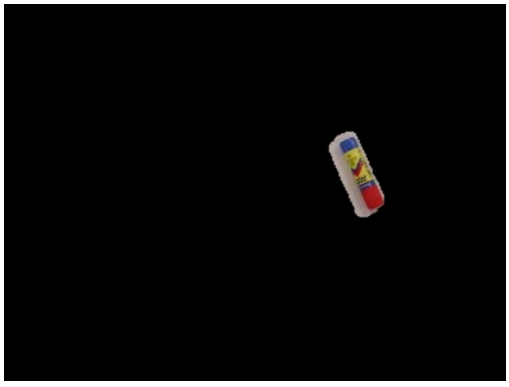
L’objet 2



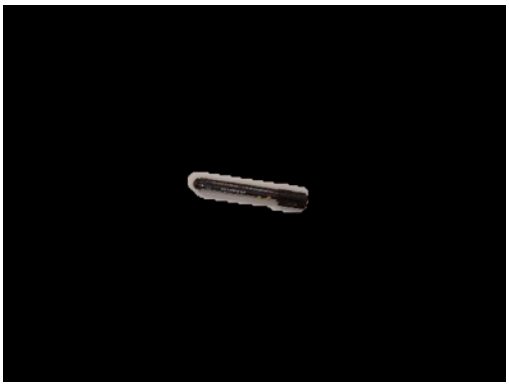
L’objet 3



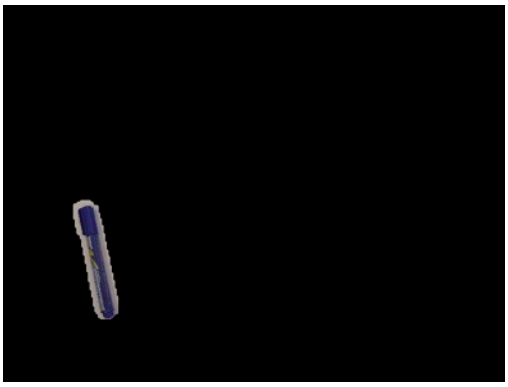
L’objet 4



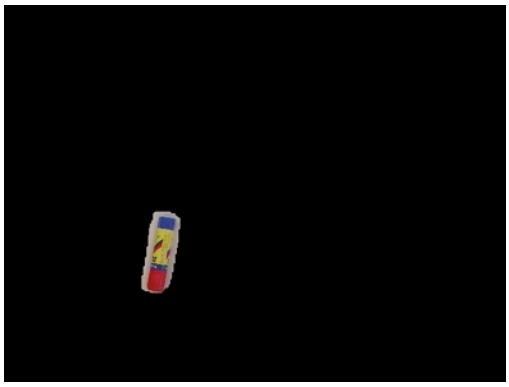
L’objet 5



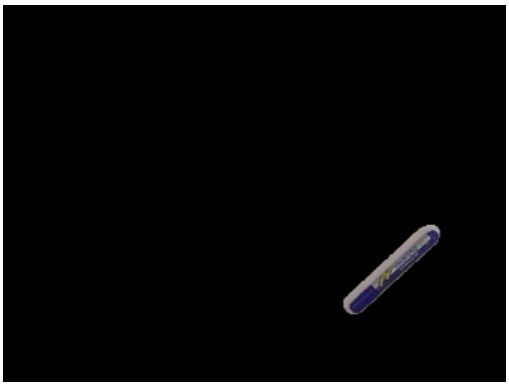
L’objet 6



L’objet 7



L’objet 8



L’objet 9

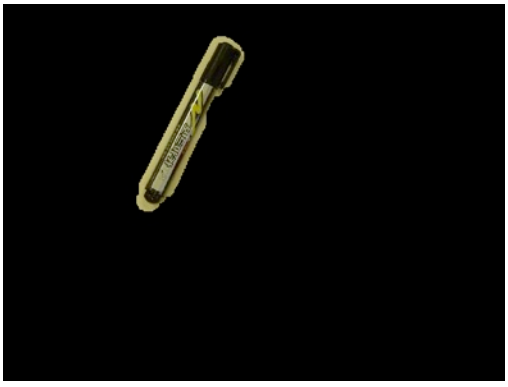
L’objet 10

L’objet 11

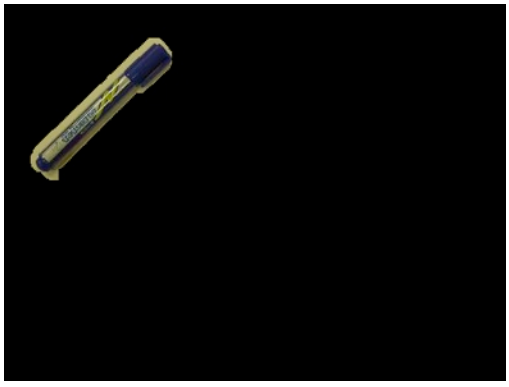
L’image objets4.jpg

On obtient 10 objets suivants :

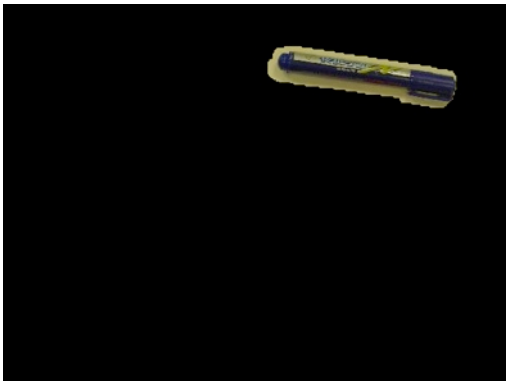
L’image objets4.jpg



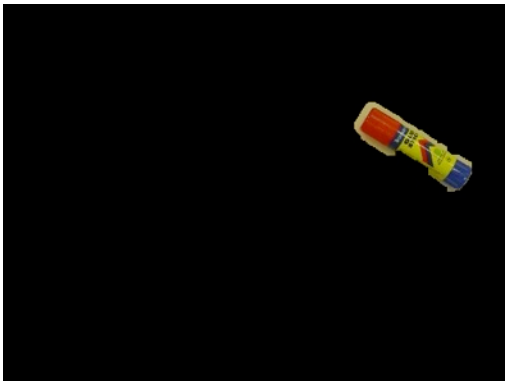
L’objet 1



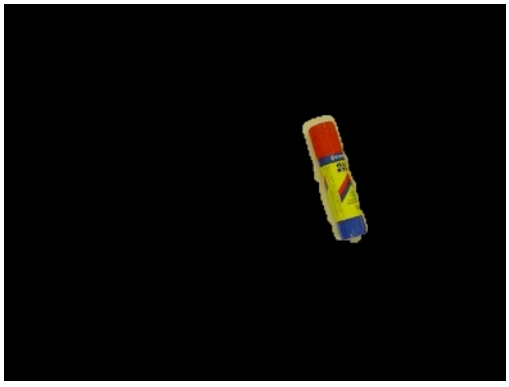
L’objet 2



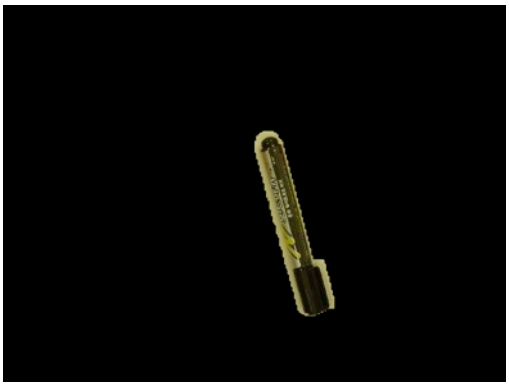
L’objet 3



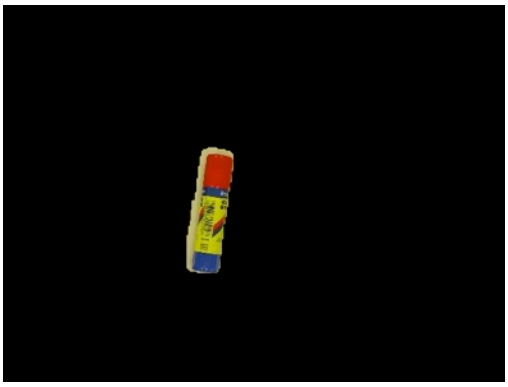
L’objet 4



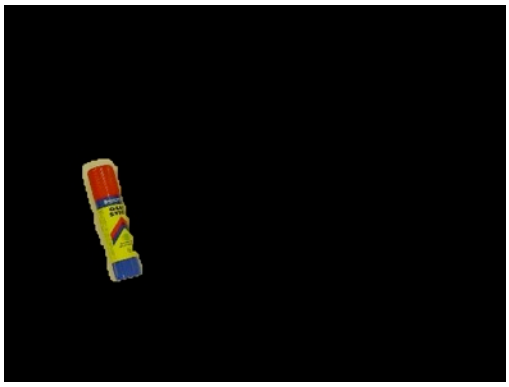
L’objet 5



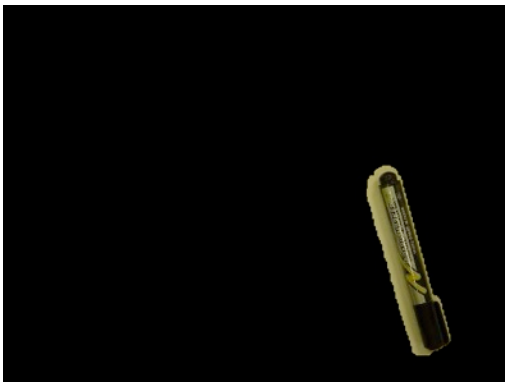
L’objet 6



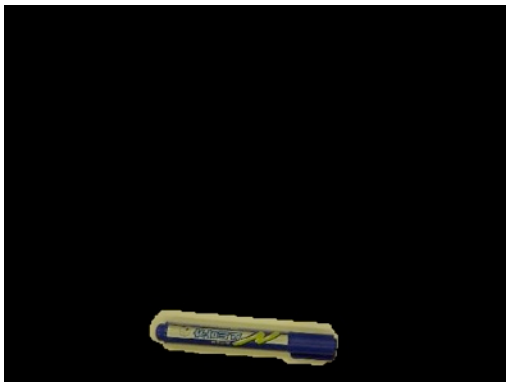
L’objet 7



L’objet 8



L’objet 9



L’objet 10

ANALYSE

Selon moi, les résultats obtenus pour toutes quatre images sont très bons, les objets sont assez précisément isolés, les contours des objets assez précis, les trous remplis et il n’y a pas de bruit dans les résultats. Cela montre que ma chaîne de traitement est concordante avec le problème.

Dans le processus de la réalisation de TP6, le choix des paramètres est très difficile car cela influencera le résultat obtenu. Tout d’abord, la formule de calculer deux points de la fonction de transformation influencera le contraste de l’image. Ensuite, deux seuils de l’algorithme Canny décideront le nombre des contours. S’il y a plusieurs contours, les objets obtenus seront peut-être grosses. S’il y a moins de contours, on ne pourra pas segmenter des régions. Enfin, le nombre de fois de faire la dilatation et l’érosion est aussi important. Si on fait la dilatation plusieurs fois, on pourra remplir tous les trous, mais les régions se seront liées, ou il y a plusieurs pixels redondants dans les objets segmentés. Et si on fait l’érosion plusieurs fois, on pourra créer des trous. De plus, le nombre de fois de faire la dilatation et l’érosion est bon avec cette image, mais ne pas bon avec les autres images.

Dans quatre images entrées, il existe des régions à l’intérieur des autres régions. Les autres méthodes peuvent les résoudre en faisant la dilatation pour remplir les trous ou colorer ces régions par la même couleur. Mais c’est très complète. Ma méthode a évité ces cas en faisant trois phases *Effacer les bruits*, *Effacer le fond* et *Convertir le fond blanc en le fond noir*. C’est très simple et efficace.

Selon moi, trois fois de l'érosion, six fois de la dilatation et puis deux fois de l'érosion sont le meilleur choix pour toutes les quatre images. Le programme peut segmenter bien les objets difficiles à segmenter :



Mais on doit accepter les objets qui ont les pixels redondants :



Il existe les pixels redondants parce qu'il existe la redondance des contours et l'ombre des objets.

Par ailleurs, il existe encore quelques objets qui ont les pixels perdus :



Il existe ces cas parce que les pixels sont ceux dont la valeur de gris est identique avec celle des pixels correspondants au fond de l'image.

CONCLUSION

Toutes les applications de traitement d'images doit avoir trois étapes pré-traitement, traitement et post-traitement. Toutes les trois étapes a aussi le rôle important. Choisir automatique ou manuellement les valeurs pour les paramètres dépend du problème. Ma chaîne de traitement est convenable avec quatre images testées. Elle est complète avec plusieurs phases mais efficace.