TP Vision par Ordinateur 2

Le but de ce second TP est de s'entraîner à extraire des informations simples d'une image, et de les combiner de façon à détecter des objets élémentaires dans une image, comme la position et le diamètre de balles rouges, par exemple.
Nous verrons également comment interagir avec notre application OpenCV de façon simple mais efficace.

Exercice 1 : **Les gradients**
Pour traiter efficacement une image, on commence par la réduire à l'information qui nous intéresse : par exemple, si on veut détecter des cercles, on sera intéressés uniquement par le contour des objets, le reste étant de l'information superflue.
Un contour est un endroit de l'image ou il y a une variation relativement importante de valeur entre deux pixels voisins.
Les variations de valeurs sont appelées gradients de l'image.
Il existe plusieures méthodes pour les calculer, nous nous interesserons ici à Sobel.
http://en.wikipedia.org/wiki/Sobel_operator

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/imgproc/doc/filtering.html#sobel

Ce qui vous est demandé :
En partant d'une image en niveau de gris, appliqués le filtre de sobel avec:

1. une dérivée en x de 1, et pas de dérivée en y
2. une dérivée en y de 1, et pas de dérivée en x
3. une dérivée en x et y de 1
4. affichez les trois versions

Exercice 2 : **La détection de contours**

Les gradients nous permettent d'évaluer la variation de valeur locale d'une image, mais ne permettent pas de localiser les contours : pour cela on utilisera Canny, qui se base lui même sur Sobel, et qui permet d'avoir une réponse binaire (contours ou non) et précise.
http://en.wikipedia.org/wiki/Canny_edge_detector

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/imgproc/doc/feature_detection.html#canny
http://docs.opencv.org/modules/imgproc/doc/filtering.html#blur

Ce qui vous est demandé :

1. Appliquez Canny sur votre image en niveau de gris.
2. La version d'opencv de Canny ne comportant pas de phase de debruitage, floutez un peu votre image avant d'appliquer Canny.
3. Affichez les deux versions

Exercice 3 : **Afficher en surimpression**

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/core/doc/basic_structures.html#mat-setto
http://docs.opencv.org/modules/core/doc/basic_structures.html#scalar

Ce qui vous est demandé :

1. prendre l'image en niveau de gris
2. la convertir en BGR dans une image ***display***
3. remplir ***display*** de la couleur de votre choix (représentée par un **Scalar**), en utilisant votre image canny comme masque
4. afficher display

Exercice 4 : Changer les valeurs durant l'exécution

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/highgui/doc/user_interface.html#createtrackbar

Ce qui vous est demandé :

On peut changer des valeur au moyen de trackbars. En s'inspirant de l'exemple ci-dessous, ajoutez des trackbars a la fenetre display pour regler la taille du flou (qui doit être impaire et > 1) et le seuil max de canny (dans 1-255, le second seuil sera la moitié de cette valeur).

```
cv::Mat gray;
int threshold = 100;

void compute(int val, void* ptr){
        if(ptr != NULL)
        *((int*)ptr) = val;
      cv::Mat thres = gray > threshold;
      cv::imshow("threshold image", thres);
}

int main(void){
         cv::Mat img = cv::imread("C:/lena.jpg");
         if(img.empty())
                  img = cv::Mat(512,512,CV_8UC3);
         cv::imshow("input", img);

         gray= cv::Mat(img.size(), CV_8UC1);
         cv::cvtColor(img, gray, CV_BGR2GRAY);
         cv::imshow("Gray", gray);

         compute(0,NULL);
         cv::createTrackbar("threshold","threshold image",
                                    &threshold,255, compute, &threshold);
       cv::waitKey(-1);
}
```

Exercice 5 : Détecter les cercles dans une image
Maintenant que nous savons détecter des contours, nous pouvons nous servir de point de départ de ceux-ci pour détecter des formes simples. Une méthode très connue et utilisée, permettant de détecter des lignes ou des cercles, est la transformée de Hough.
http://en.wikipedia.org/wiki/Hough_transform

Nous nous intéresserons ici à détecter des cercles, mais rien ne vous empêche de tester la détection de lignes par la même occasion.

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/imgproc/doc/feature_detection.html#houghcircles
http://docs.opencv.org/modules/imgproc/doc/feature_detection.html#houghlines
http://docs.opencv.org/modules/core/doc/drawing_functions.html#circle

Ce qui vous est demandé :

1. dans votre méthode **compute,** rajoutez la détection de cercles en utilisant **houghcircle** (partir de l'image en niveau de gris)
2. affichez les résultats, en dessinant les cercles obtenus dans votre image display au moyen de la méthode de dessin **circle**
3. ajoutez des trackbars dans display pour choisir :
    a. la valeur de flou utilisée
    b. le seuillage de canny utilisé
    c. le rayon minimum des cercles
    d. le rayon maximum des cercles
    e. le seuillage de l'accumulateur
    f. le nombre de cercles affichés

Exercice 6 : Calculer la distance a une couleur

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#min
http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#absdiff

Ce qui vous est demandé :

1. Extraire le canal Hue de votre image, comme vu dans le TP précédent.
2. Créer une variable globale huetarget, dont la valeur est entre 0 et 180.
3. Calculer la distance entre l'image Hue et huetarget. Comme la teinte est une valeur cyclique (0 et 180 représentent le rouge toutes les deux), la formule est :

   distance = *min (|Hue-targethue|,|Hue-(targethue+180)|)*

Exercice 7 : Combinons le tout

Les méthodes qui nous intéressent :
http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#countnonzero
http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#sum
http://docs.opencv.org/modules/core/doc/drawing_functions.html#puttext

Ce qui vous est demandé :

Pour chaque cercle détecté, calculez la distance moyenne de ses pixels a huetarget, puis affichez la valeur dans display.