Travaux pratiques de traitement d'image

Michèle Gouiffès
michele.gouiffes@universite-paris-saclay.fr

Environnement de travail

Le projet fourni est défini dans l'environnement de programmation `Visual Studio (VS)` sous Windows. Vous êtes libre de travailler dans un autre environnement et Système d'exploitation.

Attention ! Le projet Visual Studio a été développé sous VS 2022. Si vous utilisez une autre version de VS, il faut, pour chaque projet, aller dans Projet → Propriétés pour modifier :

- Target Platform Version en spécifiant l'une des versions qui apparaît dans le menu déroulant, par exemple **8.1**
- Platform Toolset en spécifiant la version que vous utilisez. Par exemple **Visual Studio 201 (v140)**

Chaque exercice fait l'objet d'un projet `exo1` à `exo11` et les projets sont regroupés dans une solution sous Visual Studio. Pour chaque exercice, vous devrez compléter une partie du code.

Librairie Opencv. La librairie `opencv` (pour *Open Computer Vision*) offre de nombreux outils de manipulation et d'affichage d'image, ainsi que de nombreux algorithmes de vision par ordinateur parmi les plus couramment utilisés. Vous aurez parfois besoin de la documentation qui est disponible ici : `https://docs.opencv.org/4.3.0/`.

La classe Mat. Pour commencer, voici quelques informations sur la classe `Mat` qui est utilisée pour la représentation des images. `https://docs.opencv.org/4.3.0/d3/d63/classcv_1_1Mat.html`

Classe Mat :
`Mat A;`
`Mat A (5, 3, CV_8U);` Image 8 bits de taille 5 lignes × 3 colonnes
`Mat A (5, 3, CV_8UC3);` Image à trois canaux 8 bits de taille 5 lignes × 3 colonnes .
Autres types de données : `CV_32S, CV_32F, CV_64F` pour `int`, `float` et `double`.

Initialisation :
`A.zeros(5,3, CV_8U)`
`A.ones(5,3, CV_8U)`

Accès aux données :
`im.at<uchar>(i,j)` ; Accès au pixel de ligne `i` et de colonne `j` dans une image 8 bits.
`im.at<Vec3b>(i,j).val[k]` ; Accès au pixel de ligne `( i, j)` du canal `k` dans une image à trois canaux de 8 bits. `Vec3f` pour une images à trois canaux de `float`
`Point2d p(j,i); im.at<uchar>(p);` Définition d'un point de coordonnées entières `p`. Attention car les coordonnées sont inversées.

Dimensions : `A.rows`, `A.cols`, `A.channels`

Fichier de configuration. Pour chaque exercice, vous disposez d'un fichier de configuration `config_exo1.cfg` à `config_exo9.cfg` qui vous permet de modifier les paramètres sans avoir à recompiler le programme. Le tableau **??** explique les principaux paramètres.

TABLE 1 – Principaux paramètres du fichier de configuration.

im_dir	répertoire des images (ex : `im/panneaux/`)
im_name	nom de l'image (en général `im`)
im_ext	extension de l'image (en général `.png` ou `.jpg`)
Ninit	numéro de la première image traitée
Nend	numéro de la dernière image traitée
format	nombre de chiffres significatifs dans la numérotation des images (format=1 pour im1.png et format =2 pour im01.png)
scale	facteur de réduction d'échelle de l'image, en général 1 ou 2
colorspace	espace couleur : RGB, rgb, HSV, Lab

1 Exercice 1 : espaces couleur et histogrammes

1. Analyse d'histogramme 1D.
   - Complétez la fonction `histo_1D` dans la classe `Stats` qui calcule l'histogramme d'une composante couleur `channel` d'une image d'entrée `im` :
     `Mat Stats::histo_1D( const Mat& im, int channel, int th_l, int th_h, bool display)`
   - Vérifiez le le bon fonctionnement de votre code sur les images `im0i.png` du répertoire `im/divers/` avec `i` allant de 1 à 8 (fichier `config_exo1a.cfg`).
   - Comment se caractérise l'histogramme d'une image contenant peu de variétés de couleurs ? Contenant de nombreuses couleurs ?
   - Comment se caractérise l'histogramme d'une image très contrastée ? D'une image peu contrastée ?
2. Couleur. Observez les images en prenant successivement comme espace couleur : RGB, rgb, Lab, HSV. Attention, l'ordre des canaux (ou *channels*, ou composantes) est inversé sous opencv, par exemple dans le cas de RGB, le canal 0 correspond au B , le 1 au G et le 2 au R.
   - À quoi correspondent les images L (de l'espace Lab) et V de l'espace (HSV) ?
   - Quel est l'avantage de rgb par rapport à RGB ? Quel est l'inconvénient ?
   - À quoi correspond la composante H dans l'espace HSV ? La composante S ?
3. Détection de pixels donnés. Considérons maintenant les images de bâtiments vus du ciel `im0i.jpg` du répertoire `im/batiments/` avec `i` allant de 1 à 12 (fichier `config_exo1b.cfg`). L'objectif est de choisir un espace couleur qui faciliterait le comptage des bâtiments.
   - Observez les images en prenant successivement comme espace couleur : RGB, rgb, Lab, HSV. Observez également les histogrammes 2D. Quels espaces couleur sont *a priori* les plus adaptés pour répondre à notre problème, c'est-à-dire qui permet d'isoler les pixels de toit ? Choisir un de ces espaces.
   - Dans cet espace couleur, y a t'il un histogramme 2D dans lequel la couleur de tuile se sépare des autres couleurs de l'image ?
   - Réalisez un seuillage (à l'aide des curseurs) pour détecter les toits. Vous ajusterez vos seuils sur l'image `im01.jpg` et vous testerez ensuite ces seuils sur le reste des images.

2 Exercice 2 : modification d'histogramme

1. Étirement d'histogramme. Complétez la fonction `HistoStretching` de la classe `Color` et testez son fonctionnement sur les images fournies.
   — Pourquoi l'étirement ne permet-il pas d'améliorer la qualité de certaines images peu contrastées ?
   — Quel est l'effet de l'étirement sur des images contenant peu de variété de couleurs ?
   — Même question pour les images bruitées.
2. Égalisation d'histogramme. Testez la fonction `equalizeHist` sur le images fournies.
   — Quel est l'effet de l'égalisation sur les images peu contrastées ?
   — Quel est l'effet de l'égalisation sur des images contenant peu de variété de couleurs ?
   — Même question pour les images bruitées.
3. En déduire les avantages et inconvénients de l'égalisation et de l'étirement.
4. Transformations Log et Exp. Testez les deux fonctions log et exp de transformation non-linéaire d'histogrammes sur le images fournies. Ajustez le paramètre `R`. `Mat Color::ExpTransform(Mat It, int R)`
   `Mat Color::LogTransform(Mat It, int R)`
   — Quel est l'effet de ces deux transformations ?

3 Exercice 3 : représentation fréquentielle

1. Visualisez les spectres de chaque image (fenêtre FFT)
   — Quelle est la forme du spectre lorsque l'image est peu structurée et ne contient pas de motifs répétitifs (`im01.png` et `im02.png`) ?
   — Dans le cas de motifs unidirectionnels (lignes plus ou moins verticales ou horizontales) (`im03.png` et `im04.png`), expliquez la position des maxima locaux dans le spectre.
   — Même question pour les motifs multi-directionnels. (`im05.png` à `im09.png`)
2. Filtrage passe-bas
   — Faîtes varier la fréquence de coupure du filtre passe-bas `Fb` dans le fichier de configuration `config_exo3.cfg` en prenant successivement 2, 10, 50. Observer le spectre et l'image filtrée. Concrètement, à quoi correspondent les basses fréquences de l'image ?
3. Filtrage passe-haut
   — Complétez la méthode `GaussHighPass` (dans `Frequency.cpp`) qui applique le filtrage passe-haut de l'image à l'aide d'un noyau gaussien 2D.
   `Mat Frequency::GaussHighPass(Mat imFFT0, float sigma_c, float sigma_r)`
   — Faîtes varier la fréquence de coupure du filtre passe-haut `Fh` dans le fichier de configuration `config_exo3.cfg` en prenant successivement 10 et 200. Observer le spectre et l'image filtrée. Concrètement, à quoi correspondent les hautes fréquences de l'image ?

4 Exercice 4 : convolution et filtrage dans l'image

1. Complétez la fonction de convolution dans la classe `filtering`
   `Mat Filtering::convolve(Mat I, Mat K)`
2. Testez la convolution en appliquant les filtres de Sobel. Les résultats de ce filtrage sont les dérivées en x et y de l'image. Nous verrons dans l'exercice 8 comment détecter les contours.
3. Complétez la fonction calculant le noyau de convolution gaussien de taille $W \times W$ dans la classe `filtering`
   `Mat Filtering::GaussKernel ()`

colorspace = Lab channel0 = 1 channel1 = -1 channel2 = -1 K = 3 itermax = 10 errormin=0.3	colorspace = RGB channel0 = 0 channel1 = 1 channel2 = 2 K = 10 itermax = 100 errormin=1
Classification en 3 classes dans l'espace couleur LAB en utilisant seulement la composante a, avec comme critères d'arrêt : 10 itérations maximum et une erreur < 0.3	Classification en 10 classes dans l'espace couleur RGB en utilisant les 3 composantes, avec comme critères d'arrêt : 100 itérations maximum et une erreur < 1

TABLE 2 – Exemples de paramètres du k-means.

4. Comparez le filtrage pour différentes valeurs de W en modifiant le fichier de configuration. W doit être impair.
5. Comparez le filtrage gaussien et le filtrage médian pour une même taille de filtre W. Pour cela, vous utiliserez les images bruitées `im0i.png` pour i allant de 1 à 5 dans le répertoire exo4 (utilisez le fichier `config_exo4b.cfg`).

5 Exercice 5 : classification

1. Complétez la fonction k-means, qui créée une image de K classes de pixels :
   `Mat Clustering::kmeans(Mat I)`
   Cette fonction doit permettre de faire une classification sur 1, 2 ou 3 composantes. On précisera les composantes de son choix dans le fichier de configuration `config_exo5.cfg` (voir tableau **??** pour des exemples). L'objectif est de détecter les panneaux routiers rouge. Les pixels des panneaux doivent donc être affectés à une même classe.
2. En utilisant les 3 composantes de l'espace RGB, faites une clustering en 3 classes. Ce résultat peut-il permettre de simplifier le problème de détection de panneaux ?
3. Même question en utilisant l'espace rgb. D'après vous, pourquoi a t'on ce résultat ?
4. En gardant l'espace rgb, testez la classification en 3 classes sur une seule composante. Comparez les résultats obtenus sur r, g et b.
5. Faites varier K afin de déterminer le meilleur paramètre pour notre problème.

6 Exercice 6 : morphologie mathématique

1. Complétez les opérateurs d'érosion et de dilatation dans la classe `Morpho`
   `Mat Morpho::erode(const Mat& src, const Mat&es, int k)`
   `Mat Morpho::dilate(const Mat& src, const Mat&es, int k)` ,
   où `src` est l'image d'entrée, `es` l'élément structurant et k le nombre d'itérations.
2. Vérifiez leur bon fonctionnement sur les images de panneaux routiers fournies. Au préalable, ces images ont été binarisées après une classification $k - means$ puis une sélection de la classe contenant les pixels rouges.
3. En utilisant les fonction d'érosion et de dilatation que vous avez programmées, complétez les fonctions d'ouverture, fermeture et contours. Vérifiez leur bon fonctionnement sur les images fournies.

7 Exercice 7 : détection de contours

Les contours peuvent être détectés soit en détectant les maxima locaux du gradient (dérivées de l'image), soit en détectant les passage par 0 du laplacien (dérivées secondes de l'image).

1. **Détection des maxima de la norme du gradient.** Les contours sont détectés par seuillage simple de la norme du gradient, en utilisant `threshold`. Le seuil est le paramètre `s` du fichier de configuration.
   — Testez différentes valeurs de seuil pour les 5 images du répertoire afin d'essayer de détecter les contours des objets d'intérêt, par exemple s= 10, 25, 50.
   — Quelles sont les limitations de cette approche ?
2. **Détecteur de Canny.** Ce détecteur de contours utilise 4 paramètres `sb`, `sh`, `apertureSize`, `l2`.
   — À quoi servent ces paramètres ?
   — Choisir des paramètres qui vous semblent adaptés.
3. **Détection des passages par zéro du Laplacien.** Comparez les résultats obtenus pour différents niveaux de filtrage. Pour cela, on modifiera le paramètre W dans le fichier de configuration. Il s'agit de la largeur du filtre gaussien, à partir de laquelle le σ de la gaussienne est calculée (σ =W/2).

8 Exercice 8 : étiquetage en composantes connexes

Avant de réaliser une analyse de formes à partir des images binaires obtenues précédemment, il est nécessaire de faire un étiquetage en composantes connexes. La forme de chaque composante pourra ainsi être étudiée par la suite (exercice 9).

1. Complétez la fonction d'étiquetage en composantes connexes `regions` dans la classe `Segmentation`.
2. Vérifiez son fonctionnement sur les différentes images de panneaux routiers.

9 Exercice 9 : analyse de formes, reconnaissance de panneaux routiers

Reconnaissance de panneaux routiers. On souhaite concevoir un algorithme de reconnaissance de panneaux routiers pour l'assistance à la conduite. Les images à traiter peuvent provenir de différentes caméras avec des caractéristiques de capteurs différentes, leur taille et leur qualité est variable. La caméra étant embarquée dans un véhicule, la détection des panneaux peut souffrir de la grande variabilité des conditions d'acquisitions (changements d'éclairage, bruit, météo). Enfin, suivant la distance du panneau, sa taille est bien entendu variable dans l'image et le contenu de l'image également, avec des objets potentiellement de la même couleur que les panneaux à détecter. Vue la grande variété de panneaux routiers, nous limitons l'analyse à trois types de panneaux : les panneaux d'interdiction (rouges) de forme ronde, hexagonale, triangulaire. Vous disposez de 15 images. La figure **??** en montre trois exemples.

La première étape consiste à isoler les pixels correspondant à la couleur des panneaux. Dans un deuxième temps, la classe des pixels rouges est analyse en composantes connexes. On souhaite ensuite reconnaître les formes suivantes : rond, triangle et octogone. La méthode `compute` de la classe `Shape` calcule, à partir d'une image d'étiquettes de régions (ou de contours), les descripteurs de forme suivants :

FIGURE 1 – Exemples d'images de panneaux routiers.

Descripteur	attribut de Shape
aire de la région	`t_area`
périmètre	`t_perim`
centre d'inertie	`t_center`
limites du rectangle englobant	`t_limits`
aire du rectangle englobant	`t_areaRect`
moments spatiaux, centrés et normalisés	`t_moments`
les 7 moments invariants de Hu	`t_hu`
la signature du contours	`t_sig`

Concernant les moments, vous pouvez vous référer au site `https://docs.opencv.org/3.3.1/d8/d23/classcv_1_1Moments.html`.

Calcul de la signature du contour. Dans un premier temps, le centre d'inertie du contour est calculé. Ensuite, la ligne de contour est parcourue de proche en proche et à chaque itération t on calcule la distance euclidienne D du contour par rapport au centre. On obtient une courbe $D(t)$ qu'il s'agira ensuite d'analyser. Dans les codes fournis, ces paramètres peuvent être calculés à partir des régions et contours extraits dans les exercices précédents.

- Comment utiliser la signature pour détecter les cercles ? L'implémentation n'est pas demandée.
- Pourquoi le rapport μ_{02}/μ_{20} ne put-il pas répondre au problème ?
- Le cercle peut être détecté en calculant la *compacité* donnée par le score suivant : $\frac{4\pi A}{P^2}$ où A et P sont respectivement l'aire et le périmètre de la forme. La compacité est égale à 1 pour un cercle parfait. La compacité peut-elle être utilisée ?
- Comment la signature pourrait-elle être utilisée pour reconnaître les 3 formes ?
- Le code fourni propose une solution. Étudiez cette solution. Pour quelles raisons ne fonctionne t'elle pas dans tous les cas ?
- Comment pourrait-on utiliser la transformée de Hough pour répondre à notre problème ? L'implémentation n'est pas demandée.

10 Exercice 10 : comptage de bâtiments vus du ciel

Comptage de bâtiments. La deuxième application concerne le comptage de bâtiments vus du ciel (figure **??**). Certains des bâtiments ont un toit en tuile, de couleur caractéristique. L'objectif est de compter le nombre de bâtiments.

- Dans un premier temps, visualiser les images du projet `exo10` et imaginer une suite de traitements permettant de compter le nombre de bâtiments au toit rouge.
- Programmer cette suite de traitements dans le `main.cpp` et tester

FIGURE 2 – Exemples d'images de bâtiments vus du ciel.