Traitement des Données Multimédia

Travaux pratiques

1. Statistiques sur l'image

1.1. Contraste et brillance

Le contraste d'une image est une mesure relative des différences dans l'image. Il est donné par l'écart type des variations de niveaux d'intensité dans l'image.

La brillance, également appelée luminance, est la moyenne de tous les pixels de l'image.

Sous OpenCV, la fonction qui permet de modifier le contraste et la brillance d'une image est :

En C++: void convertTo(oclMat& m, int rtype, double alpha=1, double beta=0) const

Les paramètres de la fonction sont :

- m – The destination matrix. If it does not have a proper size or type before the operation, it will be reallocated
- rtype – The desired destination matrix type, or rather, the depth(since the number of channels will be the same with the source one). If rtype is negative, the destination matrix will have the same type as the source.
- alpha – must be default now
- beta – must be default now

1.2. Améliorer la qualité d'une image

Plusieurs approches sont proposées pour améliorer la qualité d'une image. On considère ici la méthode de l'égalisation d'histogramme.

L'égalisation d'histogramme d'une image numérique est une méthode d'ajustement du contraste de l'image qui utilise l'histogramme. Elle consiste à appliquer une transformation sur chaque pixel de l'image, et donc d'obtenir une nouvelle image à partir d'une opération indépendante sur chacun des pixels. Cette transformation est construite à partir de l'histogramme cumulé de l'image de départ.

En considérant un codage de l'image sur L bits, les étapes de l'égalisation d'histogramme sont les suivantes :

- Calcul de l'histogramme

  $H(i)$; $i \in [0 ; \mathbf{2^L}\text{-}1]$

- Normalisation de l'histogramme

  $Hn(i) = \frac{H(i)}{N}$ avec N le nombre total de pixels de l'image

- Histogramme cumulatif normalisé

  $C(i) = \sum_{j=0}^{i} Hn(j)$

- Transformation des niveaux de gris de l'image

$$f'(x,y) = C(f(x,y)) * (2^{L}\text{-}1)$$

avec f l'image origine et (x,y) les coordonnées d'un pixel

Sous OpenCV, la fonction qui permet l'égalisation de l'histogramme d'une image est :

En C++: void equalizeHist(InputArray src, OutputArray dst) Les paramètres de la fonction sont : – src – Source 8-bit single channel image. – dst – Destination image of the same size and type as src

1.3. Exercice

1. Ecrire un programme qui permet de calculer la moyenne d'une image.
2. Ecrire un programme qui affiche une fenêtre (voir figure 1) permettant de changer deux paramètres d'une image : le contraste et la brillance. Et cela, en affichant à chaque fois, la nouvelle image obtenue et ses valeurs du contraste et de brillance sur l'écran.

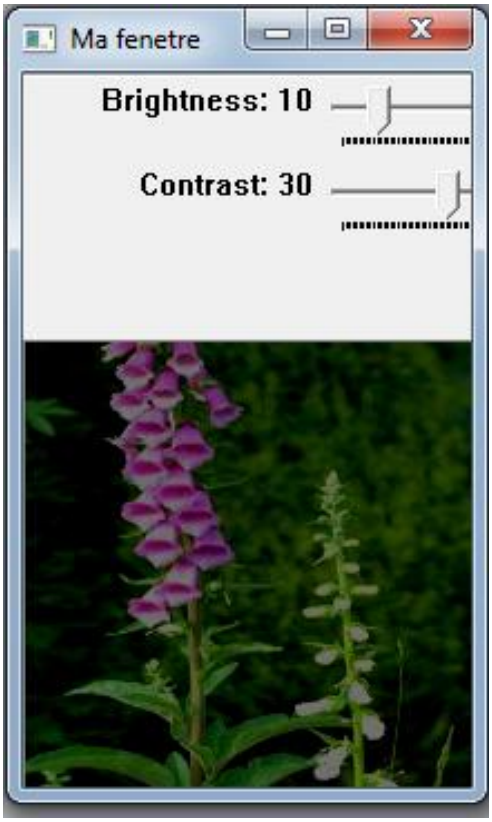

Figure1 : Exemple de la fenêtre permettant de modifier le contraste et la brillance d'une image

3. Faire une égalisation d'histogramme pour chaque sortie (image) de la question précédente.

Remarque : on peut utiliser la fonction « createTrackbar » pour créer un slider. LE prototype de cette fonction est le suivant :

En C++: int **createTrackbar**(const string& trackbarname, const string& winname, int* value, int count, TrackbarCallback onChange=0, void* userdata=0)

Les parametres de la function sont :

- trackbarname – Name of the created trackbar.
- winname – Name of the window that will be used as a parent of the created trackbar.
- value – Optional pointer to an integer variable whose value reflects the position of the slider. Upon creation, the slider position is defined by this variable.
- count – Maximal position of the slider. The minimal position is always 0.
- onChange – Pointer to the function to be called every time the slider changes position. This function should be prototyped as void Foo(int,void*); , where the first parameter is the trackbar position and the second parameter is the user data (see the next parameter). If the callback is the NULL pointer, no callbacks are called, but only value is updated.
- userdata – User data that is passed as is to the callback. It can be used to handle trackbar events without using global variables.

2. Lissage d'une image

Le lissage, également appelé flou (en anglais : *Smoothing ou blurring*), est une opération simple et fréquemment utilisée en traitement d'image. Le lissage peut être utilisé à plusieurs fins. Toutefois, il est communément considéré pour éliminer le bruit de l'image.

L'effet du lissage est de "mélanger" les niveaux de gris dans un voisinage, donc d'engendrer du flou sur l'image. En théorie, le lissage de l'image est effectué en faisant glisser un noyau (filtre ou *kernel*) sur l'image. Le plus souvent, le noyau utilisé est un filtre linéaire. Ainsi, la nouvelle valeur d'un pixel sera calculée en fonction des valeurs du noyau et celle des pixels de l'image originale, qui se trouvent dans la fenêtre déterminée par le noyau. Mathématiquement parlant, nous effectuons une opération de convolution sur une image avec un noyau. Le résultat du lissage dépend du noyau considéré ainsi que de la taille du noyau.

La taille du noyau influence fortement le résultat. Si la taille est trop grande, le lissage risque de rendre l'image brouillée et de supprimer ses petites caractéristiques. Toutefois, si la taille est trop petite, le bruit ne sera peut-être pas éliminer de l'image. Il faut donc choisir la bonne taille du noyau.

Dans cette section, on va voir comment lisser une image avec OpenCV. Dans ce cadre, plusieurs filtres sont proposés, on présentera les plus utilisés.

2.1. Lissage homogène

Le lissage homogène (en anglais : Homogeneous Smoothing ou Normalized Box Filter), est le plus simple. La nouvelle valeur du pixel est la moyenne de toutes les valeurs de l'image qui se trouvent dans la fenêtre déterminée par le noyau.

Le noyau est donné sous la forme suivante : $\frac{1}{taille\_noyau}\begin{bmatrix}1 & \cdots & 1\\ \vdots & \ddots & \vdots\\ 1 & \cdots & 1\end{bmatrix}$

Ce type de lissage est également appelé filtre de filtre moyenneur.

La fonction OpenCV à utiliser est « blur », son prototype est le suivant :

En C++: void **blur**(InputArray src, OutputArray dst, Size ksize, Point anchor=Point(-1,-1), int borderType=BORDER_DEFAULT)

Les paramètres de la fonction sont :

- src – input image; it can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
- dst – output image of the same size and type as src.
- ksize – blurring kernel size.
- anchor – anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
- borderType – border mode used to extrapolate pixels outside of the image.

2.2. <u>Lissage gaussien</u>

Le noyau du lissage gaussien est donné pas une formule mathématique. Il représente la fonction Gaussienne (cloche de Gauss) définie par :

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

La forme matricielle du noyau est :

$$\begin{bmatrix}G(-1,-1) & G(0,-1) & G(1,-1)\\ G(-1,0) & G(0,0) & G(1,0)\\ G(-1,1) & G(0,1) & G(1,1)\end{bmatrix}$$

Sous OpenCV, la fonction qui permet un lissage gaussien est :

En C++: void **GaussianBlur**(InputArray src, OutputArray dst, Size ksize, double sigmaX, double sigmaY=0,
int borderType=BORDER_DEFAULT)

Les paramètres de la fonction sont :

- src – input image; the image can have any number of channels, which are processed independently, but the depth should be CV_8U, CV_16U, CV_16S, CV_32F or CV_64F.
- dst – output image of the same size and type as src.
- ksize – Gaussian kernel size. ksize.width and ksize.height can differ but they both must be positive and odd. Or, they can be zero's and then they are computed from sigma* .
- sigmaX – Gaussian kernel standard deviation in X direction.
- sigmaY – Gaussian kernel standard deviation in Y direction; if sigmaY is zero, it is set to be equal to sigmaX, if both sigmas are zeros, they are computed from ksize.width and
- ksize.height , respectively (see getGaussianKernel() for details); to fully control the result regardless of possible future modifications of all this semantics, it is recommended to specify all of ksize, sigmaX, and sigmaY.

- borderType – pixel extrapolation method (see borderInterpolate() for details).

2.3. Lissage médian

Le lissage médian n'est pas implémenté comme un produit de convolution. Dans ce cas, la nouvelle valeur du pixel est la valeur médiane de toutes les valeurs des pixels inclus dans la fenêtre du noyau (les voisins considérés).

Sous OpenCV, la fonction qui permet un lissage médian est :

En C++: void **medianBlur**(InputArray src, OutputArray dst, int ksize)

Les paramètres de la fonction sont :
- CV_8U, CV_16U, or CV_32F, for larger aperture sizes, it can only be CV_8U.
- dst – destination array of the same size and type as src.
- ksize – aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...

2.4. Lissage bilatéral

Le lissage bilatéral est l'un des filtres les plus avancés pour lisser une image et réduire le bruit. Les trois types de lissages précédents lisseront les contours contenus dans l'image tout en supprimant le bruit. Cependant, le présent filtre peut réduire le bruit de l'image tout en préservant les conteurs. L'inconvénient de ce filtre est qu'il a une plus grande complexité en termes de temps de calcul, pour filtrer l'image.

Sous OpenCV, la fonction qui permet un lissage bilatéral est :

En C++: void bilateralFilter(InputArray src, OutputArray dst, int d, double sigmaColor, double sigmaSpace, int borderType=BORDER_DEFAULT)

Les paramètres de la fonction sont :
- src – Source 8-bit or floating-point, 1-channel or 3-channel image.
- dst – Destination image of the same size and type as src .
- d – Diameter of each pixel neighborhood that is used during filtering. If it is non-positive, it is computed from sigmaSpace .
- sigmaColor – Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighborhood (see sigmaSpace) will be mixed together, resulting in larger areas of semi-equal color.
- sigmaSpace – Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough (see
- sigmaColor). When d>0 , it specifies the neighborhood size regardless of sigmaSpace Otherwise, d is proportional to sigmaSpace .

2.5. Exercice

Ecrire un programme qui permet d'afficher les résultats des quatre types de lissage présentés. En particulier, pour chaque filtre, nous allons considérer les résultats pour des noyaux de taille allant de 1 à 51.