Traitement d'images

Cette fiche scientifique est un complément plus avancé à la fiche « Formats d'images ». Le lecteur est invité à consulter la fiche « Formats d'images » dans un premier temps.

L'objectif de cette fiche est de présenter quelques-uns des concepts utilisés en traitement d'images, ainsi que quelques traitements classiques. Certains traitements sont utilisés dans l'activité « Images numériques » (négatif, flou par moyenne), dont l'objectif est d'initier les élèves au traitement d'images en programmant en Python des traitements simples. Le reste de la fiche permet d'élargir les connaissances du lecteur sur ce domaine et peut donner des pistes d'extension de cette activité. Notez que les méthodes proposées ici ne sont que des exemples et qu'il existe de nombreuses autres méthodes ou algorithmes permettant de réaliser le même type de traitement (flou, augmentation de la netteté, *etc.*).

1 Introduction

Le traitement d'images est un domaine entre les mathématiques, l'informatique et la physique, visant à transformer des images et à en extraire des informations. Le but des traitements appliqués aux images peut être varié : améliorer la qualité de l'image, la compresser, la modifier dans un but esthétique ou artistique par exemple.

La suite de cette introduction présente les grands domaines d'utilisation du traitement d'images.

Acquisition. Lors de l'acquisition d'une image avec un appareil photo numérique, le capteur de l'appareil photo transforme la lumière captée en signal numérique. Plus précisément, le capteur d'un appareil photo est composé de millions de photosites, chaque photosite permettant de détecter la quantité de lumière rouge, verte ou bleue (voir figure 1). Une première étape de traitement d'images, le *dématriçage* permet de transformer le signal brut transmis par les photosites en image. Ce dématriçage peut être fait soit directement par l'appareil photo, soit par un logiciel de traitement d'images à partir d'un fichier RAW fourni par l'appareil photo.

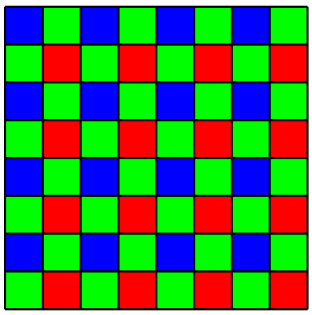


FIGURE 1 – Matrice de Bayer très souvent utilisée pour placer les photosites rouges, verts et bleus sur le capteur d'un appareil photo.

D'autres étapes peuvent être réalisées par l'appareil photo directement ou en post-production. Par exemple, la *balance des blancs* permet d'ajuster les couleurs d'une image en fonction de la couleur de l'éclairage. En effet, la couleur de l'éclairage peut grandement influencer les couleurs captées par l'appareil. Par exemple, les anciennes ampoules à incandescence émettent une lumière un peu jaune qui fait ressortir les couleurs chaudes. Les lampes halogènes, elles, émettent une lumière d'un blanc plus neutre, plus proche de la lumière du soleil. Régler la balance des blancs permet d'ajuster les couleurs pour faire en sorte que les surfaces blanches apparaissent blanches sur l'image, voir figure **??**.

FIGURE 2 – La même photographie prise avec différents réglages de balance des blancs.

Retouches. Une fois l'image obtenue, des logiciels de développement photographique et de retouche comme Photoshop, Gimp ou Lightroom par exemple permettent d'appliquer de nombreux traitements ou modifications aux images : flou, suppression du bruit[1] causé par les capteurs de l'appareil photo, amélioration de la netteté, ajustement du contraste et de la saturation des images, ... L'utilisation de l'intelligence artificielle permet aujourd'hui de réaliser des opérations beaucoup plus complexes (précédemment réservées à des utilisateurs avancés de logiciels de retouches) en quelques clics comme par exemple changer le ciel ou supprimer des éléments d'une image, voir figure 3.

(a) Image de base. (b) Ciel nuageux. (c) Coucher de soleil.

FIGURE 3 – Changement du ciel sur une photographie avec Photoshop grâce à l'intelligence artificielle.

Vision par ordinateur et intelligence artificielle. Certains traitements sont utilisés pour faciliter l'exploitation des images par la suite. Des traitements d'amélioration du contraste ou de détection des contours peuvent par exemple servir à améliorer la qualité et la lisibilité d'images issues de l'imagerie médicale (échographie, scanner, *etc.*). Ces opérations peuvent également faciliter le traitement automatisé des images par un ordinateur. Par exemple, la figure 4 présente les différentes étapes de traitement d'une image acquise par la caméra embarquée d'une voiture afin de détecter les lignes délimitant les voies. Ces informations peuvent ensuite être utilisées par la voiture pour alerter le conducteur en cas de franchissement de ligne ou pour corriger automatiquement la trajectoire de la voiture.

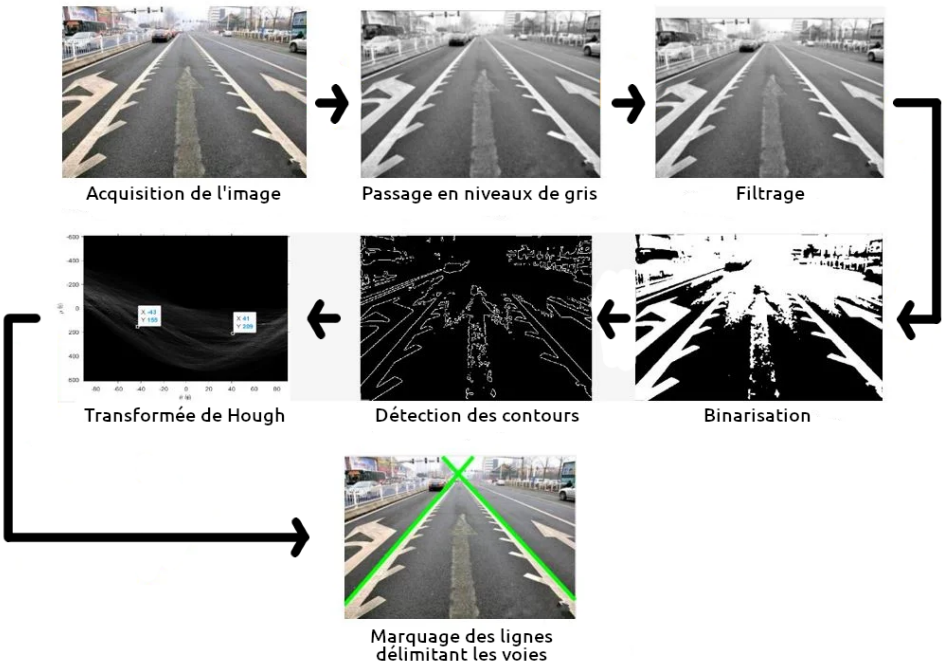


FIGURE 4 – Exemple de processus complet de traitement d'une image acquise par une caméra embarquée par une voiture pour la détection des voies de circulation [XW21].

1. En photographie numérique, le bruit correspond à une dégradation de l'image entre le moment de sa capture et de son enregistrement. Visuellement, le bruit correspond aux tâches colorées, plus claires ou plus foncées générées par les conditions de capture et les imperfections du capteur.

2 Systèmes de représentation des couleurs

Pour représenter les couleurs d'une image, il est possible d'utiliser différents systèmes. Le plus courant est le système *RGB* (*Red, Green, Blue* ou Rouge, Vert, Bleu en français). Dans ce système, chaque couleur est représentée par 3 valeurs entre 0 et 255 : une pour le rouge, une pour le vert et une pour le bleu.

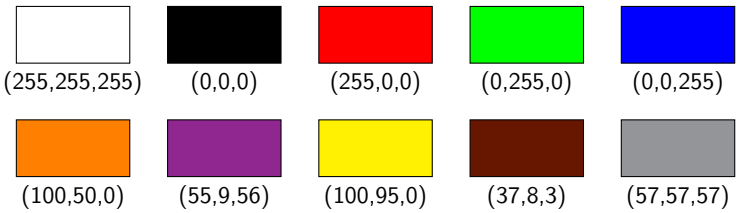


FIGURE 5 – Exemples de couleurs dans le système RGB.

Le système *HSV* (*Hue, Saturation, Value*) ou *TSV* en français (Teinte, Saturation, Valeur) est un autre système de représentation des couleurs. Dans ce système, chaque couleur est représentée par 3 valeurs :

- La *teinte* est un angle sur le cercle des couleurs (voir figure 6). Par exemple, 0^o (ou 360^o) correspond au rouge, 120^o au vert et 240^o au bleu.
- La *saturation* est l'intensité (ou la « pureté ») de la couleur. Elle s'exprime en pourcentage. Plus la saturation est faible, plus la couleur est grisée, « fade ». Au contraire, lorsque la saturation est élevée, la couleur est vive.
- La *valeur* est la « brillance » de la couleur. Elle s'exprime aussi en pourcentage. Plus la valeur est faible, plus la couleur est sombre, 0 correspondant au noir.

Il est possible de passer du système RGB ou système TSV et inversement grâce aux formules présentées dans la suite de cette section. La figure 7 présente la représentation des mêmes couleurs que la figure 5, mais dans le système TSV.

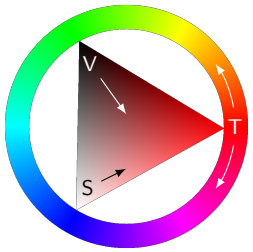


FIGURE 6 – Roue des couleurs du système TSV.

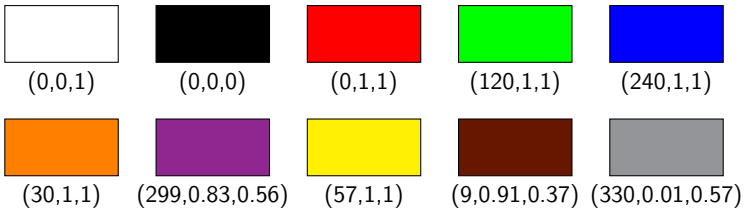


FIGURE 7 – Exemples de couleurs dans le système TSV.

Conversion du système RGB au système TSV. Soit une couleur dont les valeurs sont (r, g, b) avec $r, g, b \in \{0, .., 255\}$ dans le système RGB. Les coordonnées équivalentes dans le système TSV sont (t, s, v) calculées ainsi :

$$t = \begin{cases} 0, & \text{si } max = min \\ (60 \times \dfrac{g-b}{max-min} + 360) \mod 360, & \text{si } max = r \\ 60 \times \dfrac{b-r}{max-min} + 120, & \text{si } max = g \\ 60 \times \dfrac{r-g}{max-min} + 240, & \text{si } max = b \end{cases}$$

$$s = \begin{cases} 0, & \text{si } max = 0 \\ 1 - \dfrac{min}{max}, & \text{sinon} \end{cases}$$

$$v = \frac{max}{255}$$

où $min = \min\{r, g, b\}$ et $max = \max\{r, g, b\}$.

Conversion du système TSV au système RGB. À l'inverse, soit une couleur dont les valeurs sont (t, s, v) avec $t \in [0, 360[$ et $s, v \in [0, 1]$ dans le système TSV. Pour calculer les coordonnées équivalentes dans le système RGB (r, g, b), il faut d'abord calculer des valeurs intermédiaires :

$$t' = \left\lfloor \frac{t}{60} \right\rfloor \mod 6$$
$$f = \frac{t}{60} - t'$$
$$l = v \times (1 - s) \times 255$$
$$m = v \times (1 - f - s) \times 255$$
$$n = v \times (1 - (1 - f) \times s) \times 255$$

Les valeurs (r, g, b) sont ensuite obtenues ainsi :

- Si $t' = 0$, $(r, g, b) = (v \times 255, n, l)$.
- Si $t' = 1$, $(r, g, b) = (m, v \times 255, l)$.
- Si $t' = 2$, $(r, g, b) = (l, v \times 255, n)$.
- Si $t' = 3$, $(r, g, b) = (l, m, v \times 255)$.
- Si $t' = 4$, $(r, g, b) = (n, l, v \times 255)$.
- Si $t' = 5$, $(r, g, b) = (v \times 255, l, m)$.

3 Ajuster la saturation

En utilisant le système TSV, il est facile d'ajuster la saturation d'une image, en multipliant la valeur de la saturation de chaque pixel par un facteur $v \geq 0$. Pour obtenir la saturation d'un pixel après ajustement, si la valeur initiale de sa saturation est s, il suffit d'appliquer la fonction :

$$f(s) = v \times s$$

Si $v = 0$, la saturation de tous les pixels est mise à zéro et on obtient une image en niveaux de gris. C'est ce qu'on appelle une *désaturation*. Si $0 < v < 1$, les couleurs deviennent plus ternes. À l'inverse, si $v \geq 1$, les couleurs deviennent plus vives.

(a) Image en entrée ($v = 1$).

(b) Après désaturation ($v = 0$).

(c) Après avoir augmenté la saturation par un facteur $v = 1,5$.

FIGURE 8 – Exemple d'ajustement de la saturation.

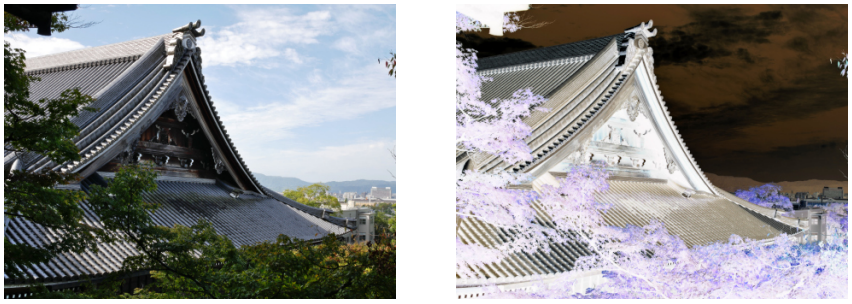
4 Négatif

Les photographies argentiques sont souvent réalisées sur des pellicules *négatives*. Lorsqu'on observe la pellicule à l'oeil nu après avoir pris une photo, les couleurs sont inversées, ainsi que les zones claires et les zones foncées. C'est lors du développement de la pellicule qu'on retrouve l'image avec ses vraies couleurs. Un traitement numérique peut réaliser la même chose, par exemple pour développer par ordinateur des pellicules qui ont été numérisées.

Pour réaliser ce traitement, il suffit d'inverser la valeur de chaque pixel. Par exemple, si la valeur v est comprise entre 0 et 255, il suffit d'appliquer la fonction :

$$f(v) = |255 - v|$$

Si l'image est en couleur, il faut inverser chaque valeur de chaque pixel. Par exemple dans une image RGB, pour chaque pixel, il faut appliquer f à la valeur du rouge, à la valeur du vert et à la valeur du bleu. La figure 9 montre le négatif d'une image en couleur.



(a) Image de base. (b) Négatif.

FIGURE 9 – Négatif d'une image.

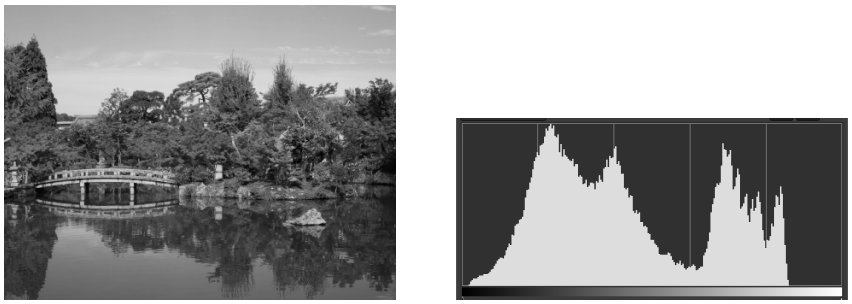
5 Histogramme

En traitement d'images, l'histogramme est une représentation de la fréquence d'apparition des différentes valeurs de pixels. Plus précisément, pour une image en niveau de gris codée en utilisant L niveaux de gris, l'*histogramme* correspond à la fonction H ci-dessous, où n_k est le nombre de pixels de niveau k et n est le nombre total de pixel de l'image :

$$H(k) = \frac{n_k}{n}, \quad 0 \leq k < L$$

Autrement dit, il s'agit de la probabilité $\mathbb{P}(x = k)$ d'occurrence d'un pixel de niveau k dans l'image. On parle aussi d'*intensité*.

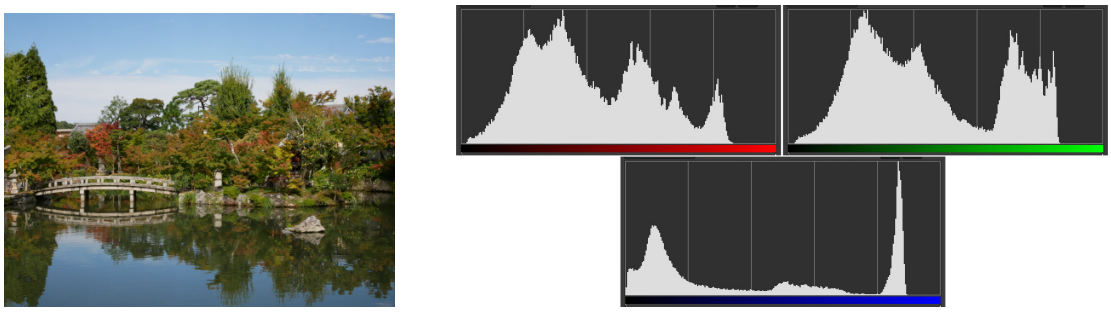
Par exemple, sur l'histogramme de la figure 10, on peut voir que la majorité des pixels sont gris foncés (pics sur la partie gauche de l'histogramme) ou gris clairs (pics sur la partie droite). Il n'y a presque aucun pixel parfaitement blanc ou parfaitement noir, ce qui peut être vérifié sur l'image.



(a) Image. (b) Histogramme.

FIGURE 10 – Histogramme d'une image en niveaux de gris.

Pour une image en couleurs, un histogramme peut être calculé pour chaque composante de couleurs, par exemple un pour le rouge, un pour le bleu et un pour le vert, voir figure 11.



(a) Image. (b) Histogramme.

FIGURE 11 – Histogramme d'une image en couleurs.

6 Ajuster le contraste

Le *contraste* est la différence de luminosité entre différentes parties d'une image. L'œil humain a tendance à mieux percevoir une image contrastée. Pour augmenter le contraste d'une image, on peut manipuler les valeurs des pixels pour changer la forme de son histogramme. Une méthode d'ajustement automatique du contraste est l'*égalisation d'histogramme*. Cela consiste à changer les valeurs des pixels afin de mieux répartir l'intensité sur l'ensemble des valeurs possibles.

À partir des valeurs de l'histogramme, on calcule la *fonction de répartition* :

$$R(k) = \mathbb{P}(x \leq k) = \sum_{i=0}^{k} \frac{H(i)}{n}, \quad 0 \leq k < L$$

où n est le nombre total de pixels dans l'image et L le nombre de niveaux de couleurs. Cela permet de définir la fonction associant les nouvelles valeurs de pixel :

$$E(k) = \frac{R(k) - R_{\min}}{n - 1} \times (L - 1), \quad 0 \leq k < L$$

où $R_{\min} = \min\{R(k) : 0 \leq k < L\}$. Ainsi, si I est l'image en entrée et $I_{x,y}$ est la valeur du pixel de coordonnées (x, y) sur cette image, la nouvelle valeur du pixel (x, y) est $I'_{x,y} = E(I_{x,y})$.

(a) Image de base.

(b) Image après égalisation.

(c) Histogrammes avant et après égalisation.

FIGURE 12 – Ajustement du contraste par égalisation d'histogramme.

Une autre méthode est la normalisation de la plage de valeurs des pixels en entrée vers une autre plage de valeurs. Cette normalisation peut par exemple être linéaire :

$$I'_{x,y} = (I_{x,y} - min) \times \frac{max' - min'}{max - min} + min'$$

où les valeurs des pixels de l'image d'entrée sont dans la plage $[min, max]$ et la plage des valeurs en sortie est $[min', max']$.

Par exemple, les pixels de la photo de la figure 13.a ont des valeurs comprises dans la plage $[5, 221]$. La figure 13.b montre le résultat de la normalisation linéaire vers la plage $[0, 255]$.

Ces deux méthodes ne donnent pas le même résultat. L'égalisation répartit équitablement les valeurs des pixels sur toute la plage de valeurs possibles. Après égalisation, la densité de pixels sombres est donc (approximativement) égale à la densité de pixels clairs. La normalisation consiste simplement à « étirer » l'histogramme (si la plage des valeurs en sortie est plus grande que la plage des valeurs en entrée) ou à « condenser » l'histogramme (dans le cas inverse). Une image avec beaucoup de pixels sombres garde beaucoup de pixels sombres, mais ceux-ci seront peut-être un peu plus clairs ou un peu plus sombres.

Notez que les exemples présentés ici sont réalisés sur des images en niveaux de gris. Pour une image en couleur, la même technique peut être utilisée sur les trois composantes rouge, vert et bleu. Par contre, cela peut grandement changer la balance des blancs (voir section 1). Il est donc généralement préférable d'utiliser le système TSV et d'appliquer la technique sur la valeur des pixels.

(a) Image de base. (b) Image après normalisation linéaire. (c) Histogrammes avant et après normalisation.

FIGURE 13 – Ajustement du contraste par normalisation d'histogramme.

7 Filtrage par convolution

Beaucoup d'opérations de traitement d'images, comme par exemple le floutage ou l'augmentation de la netteté, sont des *filtrages par convolution*. Ils utilisent un *noyau*, aussi appelé *matrice de convolution*. Il s'agit d'une matrice contenant des poids utilisée pour calculer la valeur d'un pixel de l'image après traitement, en fonction des valeurs des pixels proches. L'image en sortie (obtenue après le traitement) est ainsi calculée en faisant une convolution entre le noyau et l'image d'entrée.

Dans la plupart des cas, l'origine du noyau est la valeur centrale de la matrice. Cette origine correspond au pixel courant que l'on est en train de traiter. Prenons un exemple. À gauche, ω est un noyau. Au centre, I est l'image en entrée. À droite, le calcul de $I'_{1,1}$, la valeur en sortie du pixel de coordonnées $(1,1)$ de valeur 50 dans l'image en entrée.

$$\omega = \begin{bmatrix} 0 & 0,25 & 0 \\ 0,25 & 0 & 0,25 \\ 0 & 0,25 & 0 \end{bmatrix} \quad I = \begin{bmatrix} 100 & 80 & 45 & 20 \\ 80 & 50 & 20 & 5 \\ 60 & 40 & 10 & 0 \\ 30 & 25 & 5 & 0 \end{bmatrix}$$

$$\begin{aligned} I'_{1,1} &= 0 \times 100 + 0,25 \times 80 + 0 \times 45 \\ &\quad + 0,25 \times 80 + 0 \times 50 + 0,25 \times 20 \\ &\quad + 0 \times 60 + 0,25 \times 40 + 0 \times 10 \\ &= 55 \end{aligned}$$

Dans cet exemple, un poids de 0,25 est attribué aux pixels directement à gauche, à droite, au-dessus et en dessous du pixel à traiter. La valeur du pixel en sortie est donc la somme des valeurs des pixels voisins pondérées par leur poids. Ce calcul est répété pour tous les pixels de l'image.

Plus généralement, d'un point de vue mathématique, une convolution entre un noyau ω et une image en entrée I se note de la façon suivante : $I' = \omega \otimes I$

Plus précisément, si le noyau ω est de taille $(2a+1) \times (2b+1)$,

$$I'_{x,y} = \omega \otimes I_{x,y} = \sum_{i=-a}^{a} \sum_{j=-b}^{b} \omega(i,j) I_{(x-i),(y-j)}$$

où $I_{x,y}$ (respectivement, $I'_{x,y}$) est la valeur du pixel de coordonnées (x,y) dans l'image en entrée (respectivement, en sortie).

Normalisation du noyau. Si les poids de la matrice de convolution sont choisis de façon arbitraire, l'image en sortie peut être beaucoup plus sombre ou beaucoup plus claire que l'image en entrée. Les valeurs calculées peuvent même sortir de l'ensemble des valeurs autorisées pour un pixel.

Pour éviter cela, la somme des valeurs du noyau doit être égale à 1. Lorsque ce n'est pas le cas, on peut *normaliser* le noyau en divisant chaque valeur par la somme de toutes les valeurs du noyau.

Par exemple, soit le noyau ω suivant avant normalisation :

$$\omega = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Après normalisation :

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \text{ ou } \begin{bmatrix}1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\\ 1/9 & 1/9 & 1/9\end{bmatrix}$$

Gestion des bords. La convolution est réalisée pixel par pixel pour tous les pixels de l'image, en « faisant glisser » le noyau « au-dessus » de l'image :

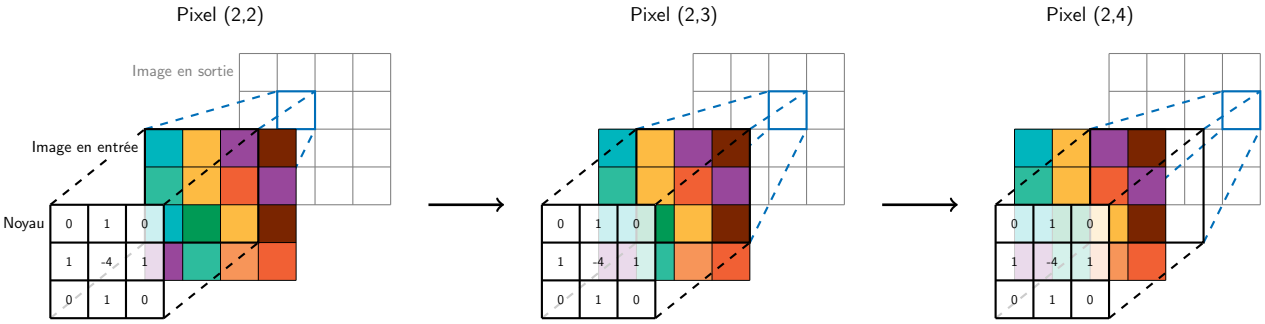
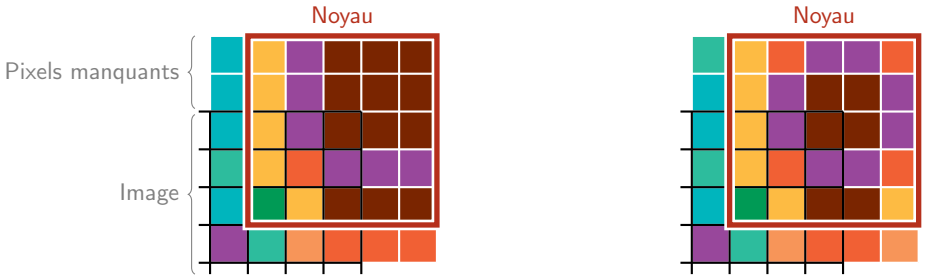


FIGURE 14 – Convolution.

Calculer les valeurs des pixels au bord de l'image, comme par exemple le pixel (2,4) dans la figure 14, pose problème. En effet, le calcul de la convolution nécessite d'utiliser des valeurs de pixels qui n'existent pas, puisqu'ils sont en dehors de l'image. Pour résoudre ce problème, plusieurs solutions sont utilisées :

- **Simuler l'existence des pixels manquants.** La première solution consiste à fixer une valeur pour les pixels manquants. Cette valeur peut être une constante (noir ou gris généralement). La valeur peut également dépendre de l'image, par exemple en étendant les bords (autrement dit, en prenant la valeur du pixel existant le plus proche) ou en utilisant l'image en miroir.



(a) Extension des bords. (b) Miroir.

FIGURE 15 – Exemples de méthodes pour simuler l'existence des pixels manquants lors du calcul de la convolution avec un noyau de taille 5×5.

- **Ne pas appliquer la transformation sur les bords.** Une autre solution est tout simplement de ne pas calculer la convolution pour les pixels au bord de l'image. Il y a alors deux possibilités : soit les pixels concernés gardent leur valeur en entrée (ce qui forme généralement un effet de cadre autour de l'image), soit ces pixels sont enlevés. Dans le deuxième cas, l'image en sortie est donc un peu plus petite que l'image en entrée.
- **Ajuster le noyau.** La dernière solution consiste à ignorer les poids du noyau qui concerne des pixels en dehors de l'image. Les poids restants sont normalisés pour compenser.

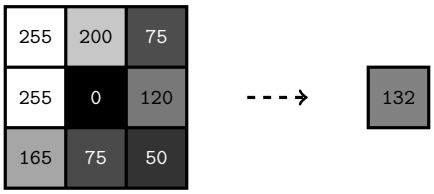
8 Flouter une image

L'application d'un floutage sur une image est une opération courante. Elle permet par exemple d'atténuer le bruit produit par les capteurs d'appareils photos ou la numérisation.

De nombreux types de flous existent.

8.1 Flou par moyenne

Le *flou par moyenne* (généralement appelé *box blur* en anglais) produit une image en sortie floutée en calculant la moyenne des valeurs des pixels voisins dans l'image en entrée. Par exemple, sur une image en niveaux de gris, la valeur de chaque pixel en sortie est calculée ainsi :



$$\frac{1}{9}(255 + 200 + 75 + 255 + 0 + 120 + 165 + 75 + 50) = \mathbf{132}$$

Dans cet exemple, les pixels considérés pour le calcul de la moyenne sont tous les pixels directement voisins du pixel à transformer, autrement dit, le rayon est 1. Ce rayon peut être paramétré pour considérer plus de pixels et obtenir un flou plus important (voir la figure 16).

(a) Image en entrée. (b) Flou par moyenne de rayon 1. (c) Flou par moyenne de rayon 5.

FIGURE 16 – Exemple de flou par moyenne.

Calculer un flou par moyenne de rayon r revient à calculer une convolution avec un noyau de taille $(2r+1) \times (2r+1)$ où tous les poids sont égaux. Par exemple, le noyau ω pour un flou par moyenne de rayon 1 est le suivant :

$$\omega = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

8.2 Flou Gaussien

Le flou par moyenne est facile à calculer mais offre un résultat peu réaliste. Le *flou gaussien* qui consiste à flouter l'image en lui appliquant une fonction Gaussienne permet d'obtenir un résultat similaire à l'observation d'une image à travers une vitre translucide.

Plus précisément, au lieu d'appliquer le même poids aux valeurs des pixels voisins comme dans le flou par moyenne, le poids utilisé ici est le résultat d'une fonction Gaussienne dépendant de la distance du pixel voisin (de coordonnées (x, y)) au pixel considéré (de coordonnées (x_0, y_0)) :

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x - x_0)^2 + (y - y_0)^2}{2\sigma^2}\right)$$

où σ est l'écart-type de la fonction Gaussienne que l'on a choisi.

En théorie, la valeur de $G(x, y)$ est non-nulle quelque soit le pixel (x, y) de l'image. Il faudrait donc prendre en compte la valeur sur l'image d'entrée de chaque pixel de l'image pour calculer la valeur d'un pixel en sortie. En pratique, au-delà de la distance 3σ, les poids associés aux pixels sont tellement faibles qu'ils peuvent être considérés comme nuls, et donc ces pixels peuvent être ignorés dans le calcul.

On peut ainsi définir un noyau de taille $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$ pour calculer le flou Gaussien par convolution. Par exemple, le

noyau ω ci-dessous permet de calculer le flou pour un écart-type $\sigma = 0,65$:

$$\omega = \begin{bmatrix} 0,000019 & 0,000659 & 0,002153 & 0,000659 & 0,000019 \\ 0,000659 & 0,022961 & 0,074981 & 0,022961 & 0,000659 \\ 0,002153 & 0,074981 & 0,244854 & 0,074981 & 0,002153 \\ 0,000659 & 0,022961 & 0,074981 & 0,022961 & 0,000659 \\ 0,000019 & 0,000659 & 0,002153 & 0,000659 & 0,000019 \end{bmatrix}$$

On peut remarquer que le poids associé aux pixels dans les coins est déjà 2000 fois inférieur au poids associé au pixel central. C'est donc une approximation suffisamment satisfaisante et beaucoup plus rapide à calculer.

Dans les logiciels de traitement d'images, le paramètre du filtre de flou Gaussien est généralement le rayon, autrement dit la taille du noyau. Chaque logiciel calcule alors, à partir de la taille du noyau désirée, l'écart-type approprié.

(a) Image en entrée.

(b) Flou Gaussien de rayon 1.

(c) Flou Gaussien de rayon 5.

FIGURE 17 – Exemple de flou Gaussien.

9 Améliorer la netteté

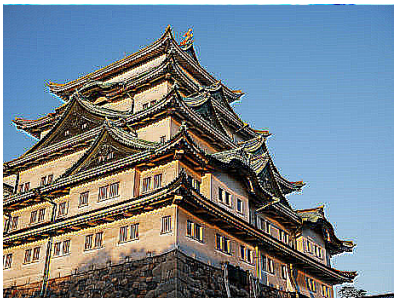
Pour améliorer la netteté d'une image, il est possible d'adopter une approche inverse à celle du floutage. Lorsqu'on floute une image, on prend en compte la valeur des pixels voisins pour la « mélanger » à la couleur du pixel courant pour atténuer la différence entre ces pixels. À l'inverse, lorsqu'on veut améliorer la netteté d'une image, on veut augmenter la différence entre un pixel et ses voisins. Il est donc possible d'utiliser une matrice de convolution comme celles-ci :

$$\omega = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \omega' = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

En mettant un poids négatif sur les pixels voisins, la différence de valeur entre ces pixels et le pixel courant est amplifiée. Plus le nombre de pixels voisins considérés augmente, plus l'effet du filtre sera important. Cependant, cette méthode provoque très rapidement du bruit sur l'image, voir figure 18.

(a) Image en entrée.

(b) Amélioration de la netteté avec la matrice ω.



(c) Amélioration de la netteté avec la matrice ω'.

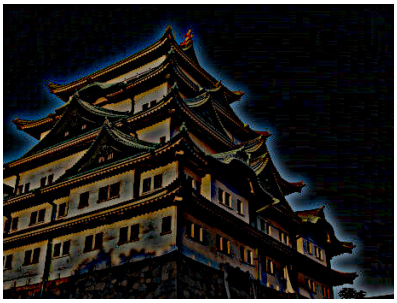
FIGURE 18 – Exemple d'amélioration de la netteté.

Pour éviter cela, les logiciels de traitement d'images utilisent plutôt une approche qui était déjà utilisée en photographie argentique mais qui est contre-intuitive : utiliser un masque de floutage pour améliorer la netteté de l'image. Les étapes sont illustrées dans la figure 19 :

1. L'image est copiée puis floutée (par exemple, avec un flou Gaussien).
2. L'image floutée est combinée à l'image de base par soustraction : pixel par pixel, on soustrait la valeur du pixel de l'image floutée à celle de l'image de base. Les zones avec le plus fort contraste sont celles qui sont le plus modifiées par le flou. On obtient donc une sorte de négatif mettant en évidence les zones de fort contraste.
3. L'image obtenue par soustraction est ensuite combinée à l'image de base par addition : pixel par pixel, on ajoute les valeurs du pixel dans les deux images. Les zones de fort contraste sont donc renforcées.

(a) Image en entrée.

(b) Étape 1



(c) Étape 2

(d) Étape 3

FIGURE 19 – Étapes pour améliorer la netteté d'une image sans introduire de bruit en utilisant un masque de floutage.

10 Détecter les contours

En traitement d'images, un *contour* est une zone de l'image dans laquelle la luminosité change brusquement. Cela peut correspondre dans la réalité à un changement de profondeur de champ, de matériaux, ou d'orientation de la surface photographiée par exemple.

La *détection des contours* est donc souvent utilisée pour la vision par ordinateur afin de reconnaître automatiquement des images ou des formes. En effet, la détection des contours permet de simplifier l'image tout en gardant les éléments importants, permettant par exemple de détecter les visages.

Une façon naïve de calculer les contours est de fixer un seuil tel que tout pixel ayant une différence de valeur supérieure à ce seuil avec au moins un pixel voisin est considéré comme faisant partie d'un contour. De nombreuses façons de détecter les contours ont été proposées. Quelques exemples sont décrits ici.

10.1 Filtre de Prewitt

Le *filtre de Prewitt* a été proposé par Judith M.S. Prewitt [Pre70]. Il utilise deux matrices de taille 3×3 permettant, par convolution avec l'image en entrée, d'approximer en tout point le *gradient (discret)* de l'intensité lumineuse de l'image.

$$\omega_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \omega_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Le premier noyau ω_x permet de calculer la « dérivée » horizontale et le second noyau ω_y permet de calculer la « dérivée » verticale. Si I est l'image en entrée, on obtient ainsi deux images en sortie : I'_x et I'_y représentant respectivement la « dérivée » horizontale et la « dérivée » verticale.

$$I'_x = \omega_x \otimes I \qquad I'_y = \omega_y \otimes I$$

L'amplitude du gradient peut alors être calculée de la façon suivante :

$$I' = \sqrt{{I'_x}^2 + {I'_y}^2}$$

Lorsqu'on utilise le filtre de Prewitt dans un logiciel de traitement des image, le résultat obtenu est généralement l'amplitude du gradient I'. Les couleurs claires représentent alors les pixels de plus grande amplitude, alors que les couleurs sombres représentent les pixels de faible amplitude.

(a) Image de base.

(b) Filtre de Prewitt.

FIGURE 20 – Exemple de détection des contours par filtre de Prewitt.

Dans certains cas d'utilisation de la détection des contours, il peut être utile de calculer également la direction du gradient :

$$\Theta = \arctan 2(I'_x, I'_y)$$

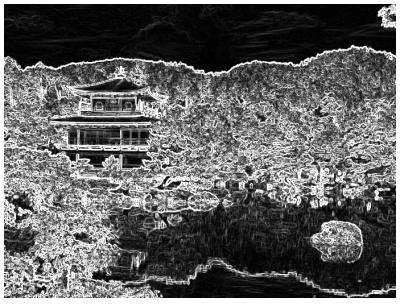
Par exemple, si $\Theta = 0$ pour un pixel de l'image, cela signifie qu'il est sur une zone de démarcation verticale où le côté droit est plus sombre que le côté gauche.

10.2 Filtre de Sobel

Le *filtre de Sobel* ou *filtre de Sobel-Feldman* a été proposé par Irwin Sobel et Gary M. Feldman [SF68]. L'idée est similaire à celle du filtre de Prewitt, mais utilise des matrices différentes :

$$\omega_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \omega_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(a) Image de base.



(b) Filtre de Sobel.

FIGURE 21 – Exemple de détection des contours par filtre de Sobel.

11 Conclusion

Les techniques de traitement d'images présentées ici sont aujourd'hui largement utilisées. Elles peuvent cependant être coûteuses en calcul et donc nécessiter un ordinateur performant avec une grande quantité de RAM (mémoire vive), un bon processeur et une bonne carte graphique (particulièrement adaptée aux calculs matriciels nécessaires en traitement des images). Elles ne sont pas seulement utilisées par les logiciels de développement photographique et de retouche comme Photoshop, Gimp ou Lightroom, mais elles sont aussi directement réalisées par les appareils photo, par exemple celui de notre smartphone, sans même qu'on s'en rende compte.

Des techniques encore plus complexes sont même utilisées pour améliorer la qualité des photos prises par votre smartphone. Par exemple, l'*HDR* (*High Dynamic Range* ou grande plage dynamique en français) est une technique utilisant une plus grand plage de valeurs possibles pour un pixel. Cela permet d'avoir un meilleur rendu des zones sombres et des zones claires. En pratique, la plupart des capteurs (photo ou vidéo) ne permettent pas de capturer directement une image HDR. En capturant plusieurs fois la même image avec différents paramètres d'*exposition*[2], et en combinant les images obtenues, on obtient une image HDR. Certains smartphones utilisent cette technique, notamment si vous utilisez un mode « photo de nuit ».

Cette fiche s'est concentrée sur le traitement d'images (fixes) mais ces techniques s'appliquent aussi à la vidéo. Par exemple, pour préserver votre vie privée, de nombreux logiciels de visioconférence proposent de flouter l'arrière-plan de votre webcam. Pour réaliser cela, de l'intelligence artificielle combinée à de la détection de contours permet de vous détecter sur l'image. Un flou, par exemple un flou Gaussien, est appliqué au reste de l'image. Ce traitement est réalisé pour chaque image du flux vidéo généré par votre webcam, et ce très rapidement puisque le résultat est affiché en temps réel.

Un autre domaine qui n'est pas du tout abordé par cette fiche est la *synthèse d'image* (ou *CGI* pour *Computer Generated Image* en anglais). Au lieu d'utiliser un ordinateur pour modifier des images du monde réel capturées par un capteur, la synthèse d'image est l'utilisation d'un ordinateur pour générer des images. Cela consiste à utiliser des modèles mathématiques pour représenter les formes présentes sur l'image, la lumière, les textures et couleur des objets *etc.* La synthèse d'image a de nombreuses applications dont le jeu vidéo, l'animation 3D, les effets spéciaux ou l'imagerie médicale. Les IA génératives comme DALL-E ou Midjourney réalisent également de la synthèse d'image.

Références

[Pre70] Judith M. S. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1) :15–19, 1970.

[SF68] Irwin Sobel and Gary M. Feldman. A 3x3 isotropic gradient operator for image processing. Présentation à Stanford Artificial Intelligence Laboratory (SAIL), 1968.

[XW21] Miao Xu and Yangzhe Wei. Detection of lane line based on robert operator. *Journal of Measurements in Engineering*, 9(3) :156–166, 2021.

2. L'exposition est la quantité de lumière qui atteint le capteur.