Chapitre 2

Notions de traitement d'images

Transformation ponctuelle

Table de conversion (des couleurs)

Introduction (1/2)

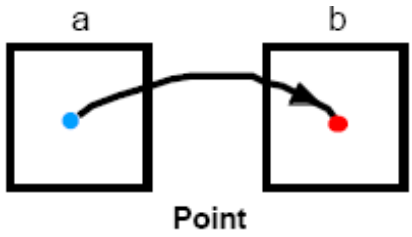
- **3 types d'opérations en traitement d'image:**

  - m : indice des lignes

  - n : indice des colonnes

  • Transformation point à point

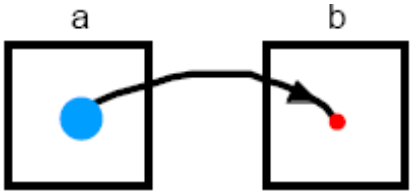
  ● = [m=m_0, n=n_0]



Point

  • Transformation locale vers un point

  ● = [m=m_0, n=n_0]



Local

En traitement d'image, on peut considérer 3 types de transformations. Les deux premiers types de transformation sont présentés sur la figure ci-dessus :

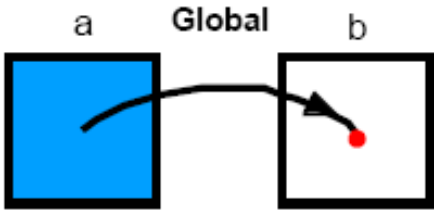
- ***de point à point***, telle que la valeur P(m_0, n_0) d'un pixel de l'image résultat « ***b*** » dépende uniquement de la valeur P(m_0', n_0') d'un pixel de l'image d'entrée « ***a*** ». Typiquement, les coordonnées (m_0', n_0') sont égales aux coordonnées (m_0, n_0) ;

- ***d'une zone locale vers un point***. Cette fois la valeur d'un pixel de l'image ***b*** ou d'un élément de ***b*** dépend d'un ensemble de pixels de ***a*** pris au sein d'une fenêtre R(m, n). Cette fenêtre est typiquement formée d'un nombre limité de pixels localisés autour du pixel (m_0', n_0') de l'image d'entrée ***a***. Par exemple, R(m, n) peut-être un bloc rectangulaire de taille (K × L) pixels, ou plus généralement un voisinage de pixels autour de (m_0', n_0') et qui conserve sa forme et sa taille quelles que soient les coordonnées (m_0', n_0').

Introduction (2/2)

- **Transformation globale vers un point**

● = [m=m_o, n=n_o]



Caractérisation de ces transformations

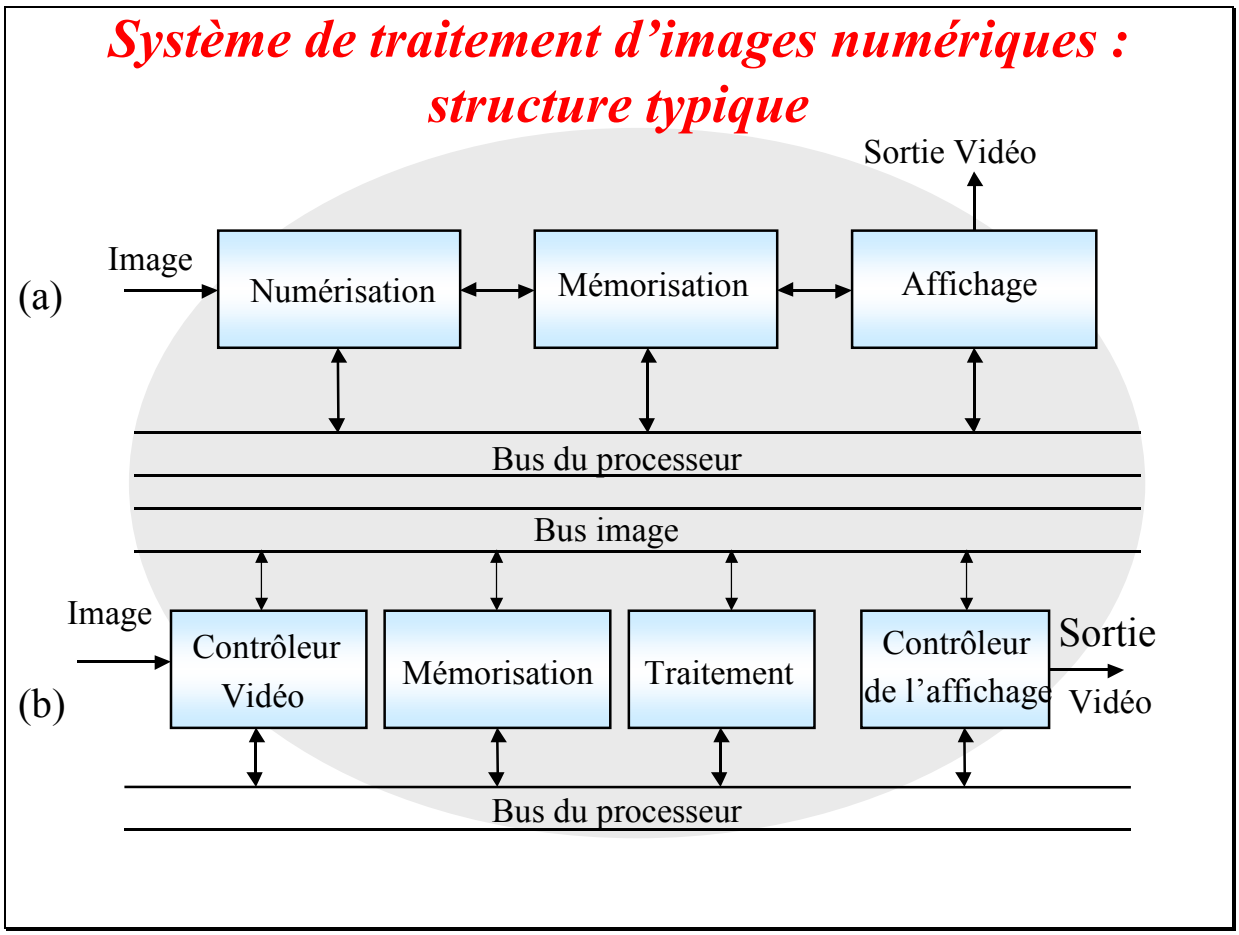
Opération	Caractérisation	Complexité par pixel
• Point	- la valeur de sortie, à une coordonnée spécifique, dépend uniquement de la valeur d'entrée à la même coordonnée.	constant
• Locale	- la valeur de sortie, à une coordonnée spécifique, dépend des valeurs d'entrée dans le voisinage de cette même coordonnée.	P^2
• Globale	- la valeur de sortie, à une coordonnée spécifique, dépend de toutes les valeurs de l'image d'entrée.	N^2

Taille Image : N x N ; taille du voisinage : p x p ; complexité: en nombre d'opérations par pixel

Le dernier type de transformation est présenté sur la figure ci-dessus.

- ***transformation globale vers un point***. La valeur calculée d'un élément (m_0, n_0) de la sortie dépend de la totalité des pixels de l'image d'entrée ***a***. C'est typiquement la cas lorsque l'on transforme globalement l'image du domaine spatial vers le domaine fréquentiel. La transformation de Fourier discrète (TFD) ou la transformation en cosinus discrète (TCD) sont des exemples bien connus (et qui seront développés dans un chapitre ultérieur) où chaque composante fréquentielle $X(\nu_X, \nu_Y)$ est fonction de l'ensemble des pixels de l'image à transformer.

Si nous considérons la complexité du traitement en terme de nombre d'opérations nécessaires pour obtenir une donnée de l'image de sortie, cette complexité est très liée au type de traitement effectué : de 1 à N^2 (pour une image carrée ($N \times N$)), ou de 1 à P^2 si on considère un opérateur mettant en œuvre une fenêtre de taille ($P \times P$) pixels, tel une convolution (cf. filtrage linéaire).
Un dernier aspect à considérer concerne les temps d'accès en mémoire image, plus ou moins réguliers, ce qui peut ralentir sérieusement le temps de traitement de l'image.



La structure typique d'un système de traitement d'image est illustré par la figure ci-dessus.

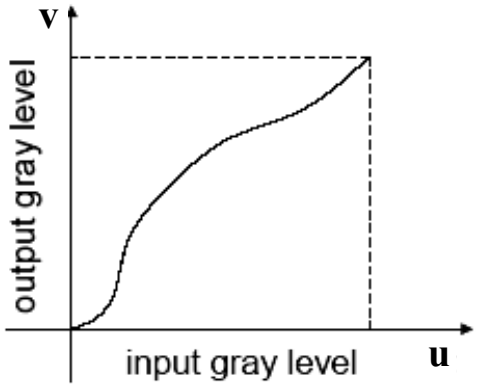
Nous découvrons en haut un exemple simple d'une chaîne de traitement constituée de la numérisation du signal vidéo analogique (échantillonnage puis quantification) produisant les données de l'image numérique. Ces données peuvent ensuite être stockées en mémoire image à la fréquence d'échantillonnage de la vidéo. Elles peuvent enfin être transmises à la seconde partie du système et/ou affichées à l'aide d'un moniteur.

Le traitement de l'image numérique est effectué à l'aide d'un processeur.
En bas, une unité dédiée au traitement d'image présente une architecture particulière. Elle est composée d'un processeur capable de réaliser des opérations (aussi bien linéaires que non-linéaires) en temps réel sur des fenêtres de taille limitée (par exemple (3×3) ou (5×5)), des transformations ponctuelles sur l'image avant et/ou après son traitement (utile à l'affichage).

Transformation ponctuelle

Transformations ponctuelles

• fait correspondre à un niveau donné de gris ou de couleur **u** du signal d'entrée **e**, un nouveau niveau de gris ou de couleur **v** du signal de sortie **s**, i.e. **v = f (u)**.



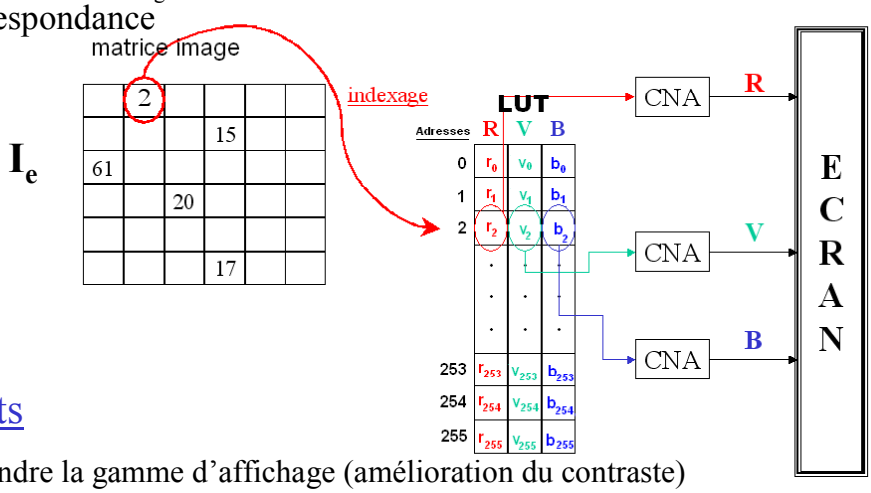
• les transformations ponctuelles sont utilisées, souvent en visualisation, pour mettre en évidence des pixels satisfaisant à une propriété donnée.

Une transformation ponctuelle peut facilement être réalisée à partir de données numériques. Elle correspond à la transformation d'une valeur (scalaire) en une autre (scalaire ou vectorielle pour une image couleur). Elle est entièrement déterminée par la fonction donnant les valeurs de sortie correspondant à chacune des valeurs d'entrée. L'usage d'une transformation ponctuelle permet, entre autres, de modifier l'apparence de l'image à l'affichage (exemple : affichage de fausses couleurs à partir d'une image en niveaux de gris), ou encore de modifier le niveau entrant de gris ou de couleur afin de compenser certaines non-linéarités dues au système d'affichage (moniteur). Une autre utilisation possible est la modification du contraste de l'image afin de le rehausser.

Transformation ponctuelle pour l'affichage

Principe (pour une image en niveau de gris)

Chaque pixel de l'image d'entrée I_e, ayant le niveau de gris N_g, a une niveau de gris transformé en $T(N_g)$ dans l'image de sortie I_s grâce à l'utilisation d'une table de mise en correspondance



Buts

- Étendre la gamme d'affichage (amélioration du contraste)
- Correction non-linéaire de l'image
- Binarisation de l'image
- Segmentation (découpage de l'image en régions composées de pixels ayant la même valeur)

Le principe de base pour la transformation ponctuelle d'une image numérique (monochrome) est de considérer la valeur d'entrée d'un pixel comme une adresse et de lire le contenu de la mémoire à cette adresse. Pour cela, les valeurs originales doivent avoir été converties préalablement en valeurs entières positives sur une échelle allant typiquement de 0 à $2^L - 1$ (par quantification et codage des niveaux de gris sur 2^L valeurs différentes). Pour un codage sur 8 bits par pixel, la taille de la mémoire sera de 256 octets.

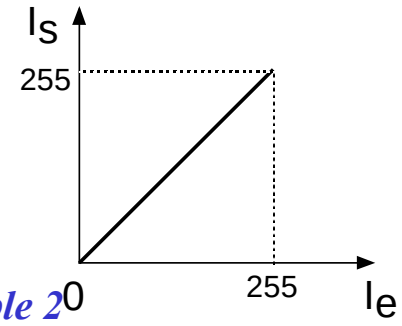
A chaque index « i » va correspondre une couleur créée par la combinaison r_i, v_i et b_i des couleurs primaires Rouge-Vert-Bleu. En effet, les valeurs r_i, v_i et b_i sont envoyées dans trois convertisseurs numérique-analogique (CNA) de façon à venir exciter respectivement les canons Rouge, Vert, et Bleu de l'écran. On obtient alors à l'affichage une couleur créée par la synthèse additive de trois couleurs primaires (technologie cathodique).

Ce type de mémoire est appelée « table de mise en correspondance » (ou table de conversion, appelée aussi LUT signifiant en anglais *Look-Up Table*). Elle est construite avant son utilisation : ni son contenu, ni les pixels de l'image ne sont modifiés durant la transformation.

La faible taille de la table permet éventuellement de la modifier pendant un temps de synchronisation vidéo (temps pendant lequel la vidéo n'est plus affichée à l'écran TV), c'est une façon de réaliser un fondu-enchaîné de transition entre 2 plans vidéo. Typiquement, pour des images monochromes, un système dispose d'une LUT à l'entrée et d'une seconde LUT en sortie du système de traitement d'images numériques. Dans le cas d'une image couleur, il faut donc considérer deux ensembles de trois LUTs : une LUT pour chacune des trois composantes couleur RVB.

Exemples de transformations ponctuelles « typiques »

Exemple 1



La transformation identité : $I_s = I_e$

Exemple 2

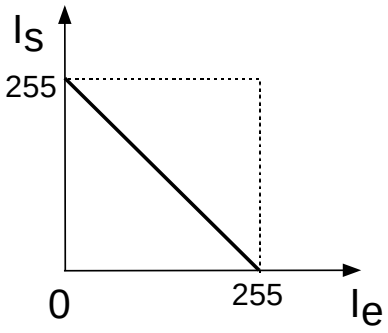


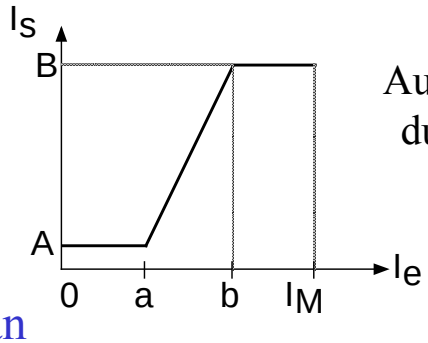
Image en “inverse vidéo”

inversion de l’échelle en niveau de gris

Pour ne pas modifier les valeurs d’une image, il faut utiliser une LUT « transparente » où le contenu à l’adresse du niveau de gris Ng est Ng. C’est la transformation Identité (exemple 1). Les niveaux de gris de Is correspondent donc aux niveaux de gris de Ie. Pour réaliser une inversion vidéo (exemple 2) où une petite valeur en niveau de gris est transformée en une valeur élevée et inversement, le contenu de la LUT à l’adresse Ng est (2^L - Ng –1).

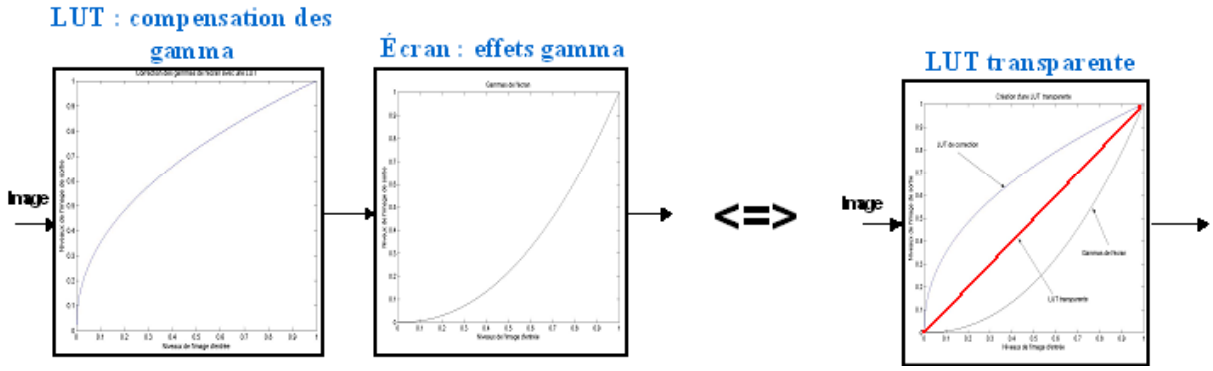
Exemples de transformations en niveaux de gris

Correction de
de la dynamique des
niveaux de gris
(*Re-scaling*)



Augmentation
du contraste

Correction du gamma écran



T2 o T1 = Opérateur Identité

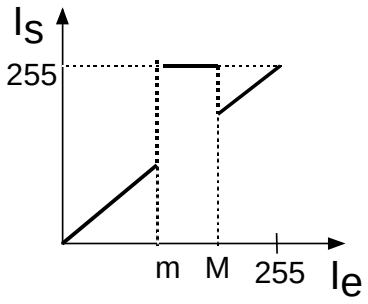
Deux exemples de transformations pour des images en niveaux de gris :

<u>figure du haut</u> : tous les pixels de l'image d'entrée Ie ayant une valeur inférieure à « a » (respectivement supérieure à « b ») auront la valeur « A » (respectivement « B ») dans l'image de sortie Is. Les niveaux de gris u_e dans l'image Ie ayant une valeur comprise entre a et b sont transformés en une nouvelle valeur v_s comprise entre A et B. $v_s = \left[\frac{u_e - a}{b - a}\ (B - A)\right] + A$.
Le but est de faire croître la dynamique des valeurs des pixels de l'intervalle de départ [a, b] et de rehausser le contraste. Pour l'augmentation du contraste, on a (B-A) > (b-a).

<u>figure du bas</u> : elle montre une correction typique d'affichage. Un tube cathodique est naturellement non-linéaire : l'intensité lumineuse reproduite à l'écran est une fonction non-linéaire de la tension d'entrée. La ***correction gamma*** peut être considérée comme un procédé permettant de compenser ce phénomène pour obtenir une reproduction fidèle de l'intensité lumineuse. Les effets ***gamma*** sont modélisés par des fonctions du type $f(x)=x^{\gamma}$, où γ est un réel qui varie entre 2 et 2,5 dans le cas de la télévision. Pour compenser ces effets, on modélise les gamma de l'écran par une LUT, et on crée la LUT de correction inverse afin que la composée « LUT correction o transformation non-linéaire écran » génère une transformation identité.

Transformations continues par morceaux

• Exemple

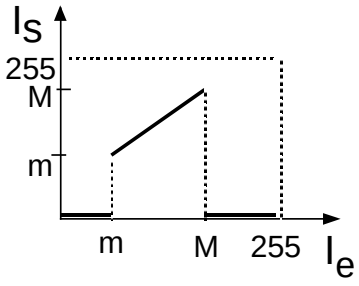


• Intérêt

Mise en œuvre de 2 seuils :

- les valeurs entre [0, ...,m[et]M,...,255] sont inchangées
- les pixels entre [m...M] sont mis à 255 (ces pixels sont « blancs »)

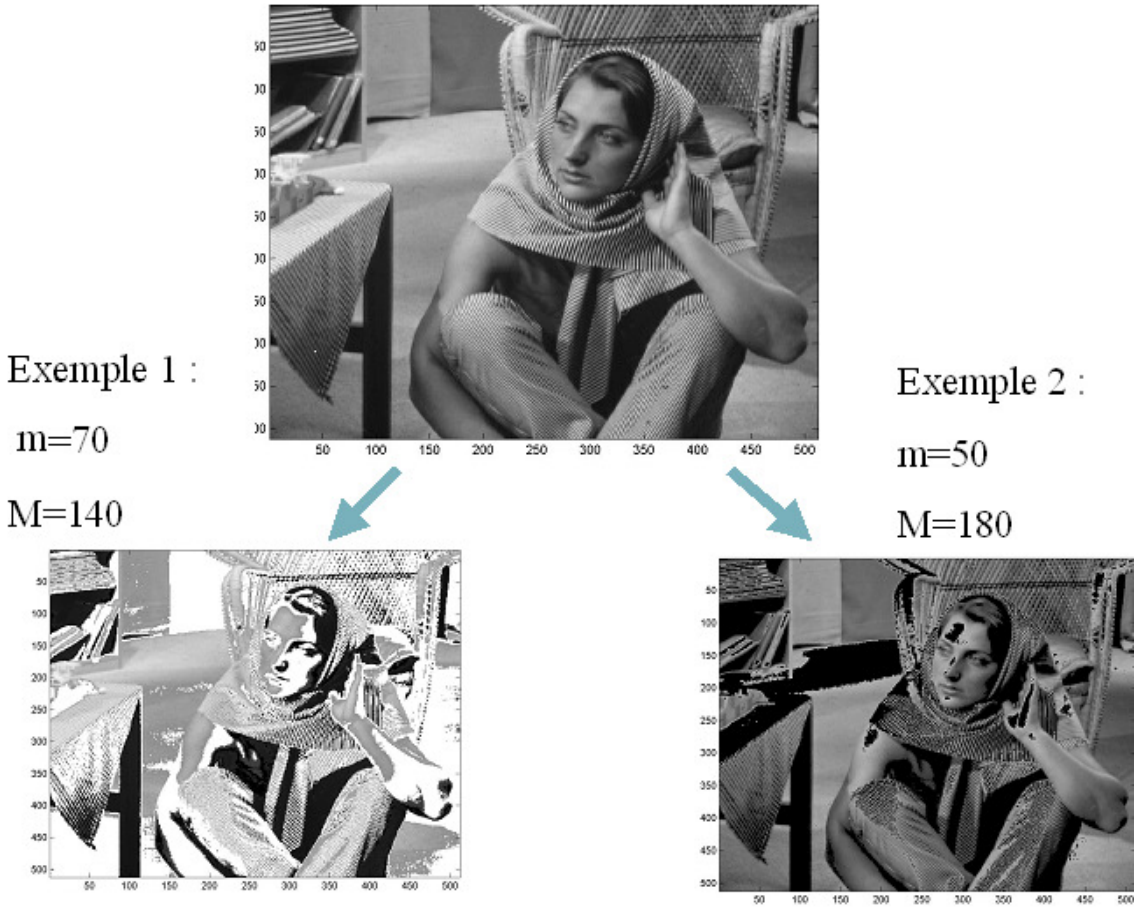
Autre transformation :



Deux autres exemples :

- figure du haut : tous les pixels de l'image d'entrée Ie dont les valeurs sont dans la gamme [m, M] sont mis à 255 (ils seront donc « blancs » à la visualisation). Pour les autres valeurs (supérieures à M ou inférieures à m) la valeur d'entrée demeure inchangée.
- figure du bas : la transformation « opposée » où seuls les pixels dont les valeurs sont appartiennent à l'intervalle [m, M] restent inchangés, les autres valeurs de pixels sont mises à 0 (« noirs »).

Ces deux cas sont des exemples de transformations continues par morceaux.



La figure ci-dessus présente, pour l'image *Barbara*, les résultats obtenus après application des LUTs continues par morceaux présentées précédemment.