Traitements d’image

Marcel Carbillet
Lab. Lagrange (UniCA, OCA, CNRS) & Dépt de Physique & Astrophysique
marcel.carbillet@unice.fr
https://lagrange.oca.eu/carbillet/enseignement/M2-GBM

I. Introduction

1- L’ANALYSE D’IMAGES / LES TRAITEMENTS D’IMAGE

- **« Analyse » et « Traitements » :**
  - **transformation des images (amélioration, codage, etc.)**
  - **extraction de l’information contenue en vue, p.ex., du contrôle automatique par la vision (chaînes de fabrication)**
  - **détection et reconnaissance de formes (p.ex. reconnaissance automatique d’écritures)**
  - **compression des images, etc.**

I. Introduction

- **Dans ce cours :**

- **détection de contours**
- **restauration/amélioration d'images**
- **via : l'analyse statistique, le filtrage, la morphologie mathématique, l'analyse de Fourier.**

- **Afin de :**

- **permettre de développer des solutions aux problèmes d'analyse d'images**
- **savoir et maîtriser ce qui se cache derrière une solution toute faite (limite de validité, évaluation du bien-fondé des résultats)**

- **Outil utilisé ici :**

  - Matlab, avec la toolbox *Image Processing*.
  - ou, de manière équivalente, Octave, avec la même toolbox (vérifier avec : *pkg list*, sinon utiliser : *pkg install -forge image* et *pkg load image* —> *voir* https://wiki.octave.org).

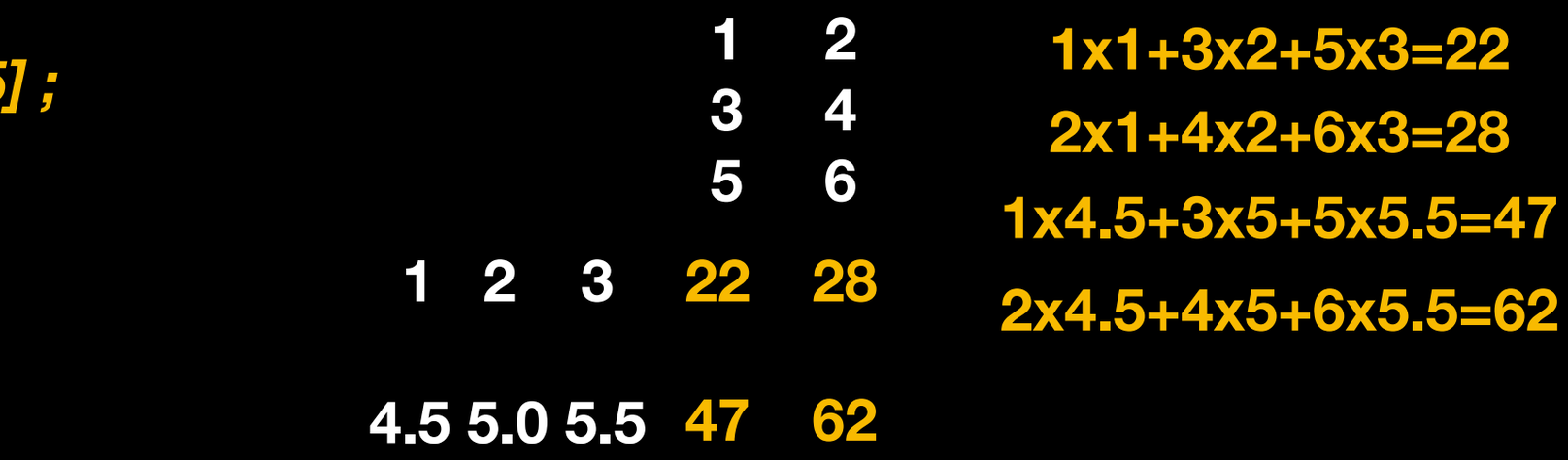
I. Introduction

- **Installation de Matlab :**
  - **Allez sur *https://fr.mathworks.com/mwaccount/register* .**
  - **Créez un compte avec votre adresse *@etu.univ-cotedazur.fr* .**
  - **Une fois connecté·e, cliquez sur la flèche bleue (proche de la licence).**
  - **Téléchargez Matlab pour votre OS (Windows/MacOS/linux), ainsi que toutes les *toolboxes* désirées (a minima *Image Processing*, par prudence *Signal Processing* et *Statistics*).**
  - **Une fois le téléchargement terminé, double-cliquez sur le fichier.**
  - **Entrez votre adresse *@etu.univ-cotedazur.fr* et acceptez le contrat de licence.**
  - **Sélectionnez (ou pas) l'option '*Add shortcut to Desktop*'.**
  - **Cliquez sur '*Begin install*' — ça va prendre un certain temps…**

I. Introduction

2- NOTIONS ÉLÉMENTAIRES DE MATLAB/OCTAVE

- **Langage de haut niveau (comme, p.ex., IDL, Maple, Mathematica, etc.)**
- **Très adapté à la manipulation de matrices (images=matrices 2D)**
- **Exemple :**
  >> M = [1 2 3 ; 4 5 6] ;
  >> M
- **Plus raffiné : utilisation de « inf:pas:sup » pour obtenir tous les entiers compris entre les 2 bornes « inf » et « sup » (par défaut : « pas »=1)**
- **Exemple :**
  >> M = [1:3 ; 4.5:0.5:5.5]
- **Produit de 2 matrices M et N (pour éviter l'utilisation de boucles) :**
  >> P = M*N ;
- **Exemple :**
  >> M = [1:3 ; 4.5:0.5:5.5] ;
  >> N = [1 2 ; 3 4 ; 5 6] ;
  >> P = M*N
  rend : *P = 22 28*
  47 62



I. Introduction

- **Pour M carrée : différence entre M*M (ou M^2) et M.*M (ou M.^2) !!**
  (Et c'est plus souvent M.*M qui nous intéresse ici - multiplication élément par élément)
- **Exemple :**

  >> M = [1 2 ; 3 4]

  >> M^2

  rend : ***7 10***
  15 22

  >> M.^2

  rend : ***1 4***
  9 16

1 2
3 4

1 2
3 4

- « ***>> help images*** » **ou** « ***>> help signal*** » **: rend la liste des fonctions disponibles de la toolbox en question**

- « ***>> help imagesc*** » **: rend le help de la fonction « imagesc »**
- « ***>> doc imagesc*** » **: pour avoir la doc complète de « imagesc »**
- « ***>> imagesc(*** » **: fait apparaître petit help contextuel… (MATLAB)**
  « ***>> imagesc*** » **: fait apparaître petit help contextuel… (OCTAVE)**
  (mais pas toutes les versions)

I. Introduction

- écriture d'un programme : placer les instructions dans un fichier avec l'extension « .m », et commencer par « close all » (fermer toutes les figures) et « clear » (effacer toutes les variables déjà existantes).

- « *>> whos* » : affiche toutes les variables existantes (+info sur celles-ci)

- trouver l'index correspondant à une valeur particulière dans un tableau : « find »
  >> idx = find(tab==0)
  rend, p.ex. : *2*
  3
  si tab = [1 0 1 ; 0 1 1]
  car Matlab considère l'origine du tableau en haut à gauche et parcours le tableau colonne par colonne… (et les indices commencent à 1 — pas à 0 !)

- mettre ces pixels à une valeur X donnée :
  >> tab(idx) = X ;
  (attention à ne pas négliger ces valeurs avec « >> tab(idx) = [] »
  => très mauvaise idée pour une matrice 2D/3D/4D/etc. !!)

I. Introduction

3- FORMATS D'IMAGE SOUS MATLAB/OCTAVE

- **Image = matrice bidimensionnelle (2D) = tableau rectangulaire**
  Un élément de l'image = un « picture element » = un pixel [px]

- **formats d'images sous Matlab/Octave :**
  - images binaires
  - images d'intensité
  - images couleurs RGB
  - images couleurs indexées

- **Pour visualiser une image :**
  >> imshow(image)
  >> imagesc(image)
  >> imtool **(pas sous Octave)**

- **Images binaires : 0 ou 1**
  Exemple :
  >> I = [0 1 0 ; 1 0 1 ; 0 1 0]
  >> imshow (I, 'InitialMagnification', 'fit') **[ou *imshow(I)* sous Octave]**

I. Introduction

- **Alternatives à imshow :**

  >> imagesc(I)
  >> colormap(gray)
  >> colorbar

  >> imtool
  puis : « File > Import from workspace » et « Fit to window »
  (pas encore implémenté sous Octave…)

- **Images d'intensités : NORMALEMENT réels compris entre 0.0 et 1.0**

  -> classe « double » (codage sur 64 bits => $2^{64} \approx 1.8447\ 10^{19}$ valeurs !)

  Ou, en alternative :

  -> classe « uint8 » (entiers codés sur 8 bits : [0 … 255], $2^8 = 256$)
  -> classe « uint16 » (entiers codés sur 16 bits : [0 … 65535], $2^{16} = 65536$)

I. Introduction

- **Exemple :**

  Créons une image en niveaux de gris constituée de colonnes passant progressivement du noir au blanc…

  >> I = ones(4,1) * [(0:6)/6]
  >> imagesc(I)
  rend une version en FAUSSES COULEURS (ou du moins dans la colormap actuelle)

  -> pour voir la table de couleurs utilisées (et les valeurs présentes dans l'image) :
  >> colorbar

  -> pour voir la version en niveaux de gris :
  >> colormap('gray') **(ou *colormap(gray)*…)**

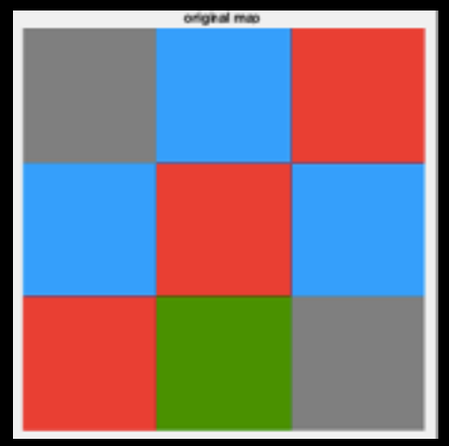
I. Introduction

- **Images couleurs RGB :**

  L'œil humain analyse les couleurs à l'aide de trois types de cellules photo-réceptrices : les « cônes » (sensibles aux basses, moyennes et hautes fréquences => rouge (Red), vert (Green), bleu (Blue) => R, G, B).

- **Exemple :**

  >> R = [0.5 0.0 1.0 ; 0.0 1.0 0.0 ; 1.0 0.0 0.5]
  >> G = [0.5 0.6 0.0 ; 0.6 0.0 0.6 ; 0.0 0.6 0.5]
  >> B = [0.5 1.0 0.0 ; 1.0 0.0 1.0 ; 0.0 0.0 0.5]
  >> I3C = zeros([size(R) 3]) ; **(ou *zeros([3 3 3])***
  >> I3C(:, :, 1) = R ;
  >> I3C(:, :, 2) = G ;
  >> I3C(:, :, 3) = B ;
  >> imagesc(I3C)



- **Modification du pixel [1,2] dans le codage « bleu » de l'image :**
  >> I3C(1, 2, 3) = 0
  >> imagesc(I3C)

I. Introduction

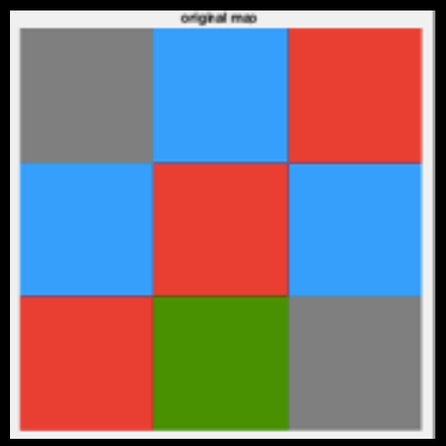
- **Images couleurs indexées :**

  Utiliser 3 matrices 2D (=1 matrice 3D) pour une image avec beaucoup de pixels est une occupation importante de la mémoire.
  => Alternative plus économique : la représentation indexée !

- **Les couleurs sont mémorisées dans une table de couleurs (« colormap ») de dimension *n*x3, avec *n*=nombre de couleurs définies.**

- **Dans l'exemple modifié précédent, on a 4 couleurs :**

	gris	turquoise	rouge (à 100%)		vert (à 60%)	
>> R = [	*0.5*	*0.0*	*1.0*	*…*	*0.0*	*…*
>> G = [	*0.5*	*0.6*	*0.0*	*…*	*0.6*	*…*
>> B = [	*0.5*	*1.0*	*0.0*	*…*	*0.0*	*…*



On peut donc écrire ça :

```
>> II = [1 2 3 ; 2 3 2 ; 3 4 1] ;
>> map = [0.5 0.5 0.5 ; 0.0 0.6 1.0 ; 1.0 0.0 0.0 ; 0.0 0.6 0.0] ;
>> imshow(II, map, 'InitialMagnification', 'fit')
```

puis utiliser *whos* pour comparer les tailles respectives *II+map* et *I3C*