Bibliothèque orientée objet sous Matlab pour le traitement d'images

D. Legland

18 mai 2010

Résumé

Réflexions sur la possibilité de créer une bibliothèque Matlab pour améliorer le traitement des images. Le but est de tenir compte des images de dimension multiples, de types multiples, de la calibration spatiale, et des meta-informations (résolution, orientation, nom de fichier...).

Table des matières

1 Introduction **2**
- 1.1 Motivation . . . 2
- 1.2 Wishlist . . . 2
- 1.3 Terminologie . . . 2

2 Nature des images **2**
- 2.1 Dimension . . . 2
- 2.2 Types de pixels . . . 3
- 2.3 Calibration spatiale . . . 3
- 2.4 Information de visualisation . . . 3
- 2.5 Meta-données . . . 3

3 Opérations souhaitées **4**
- 3.1 Taille de l'image . . . 4
- 3.2 Accès et modification des valeurs de l'image . . . 4
- 3.3 Calibration spatiale . . . 5
- 3.4 Filtrage . . . 6
- 3.5 Affichage . . . 6

4 Réflexions sur la conception **7**
- 4.1 Multiplicité des images . . . 7
- 4.2 Identification des pixels . . . 7

5 Ébauche d'implémentation **7**
- 5.1 Classe Image . . . 7
- 5.2 Classe Image « concrète » . . . 7
- 5.3 Interpolation . . . 8
- 5.4 Classe utilitaire Images . . . 9

6 Conclusions **9**

1 Introduction

1.1 Motivation

La plateforme Matlab est très performante pour traiter les images, mais manque parfois de souplesse pour manipuler les données. En particulier :

- les dimensions 1, 2 et 3 correspondent respectivement aux coordonnées y, x, et z, ce qui est peu intuitif et pénible pour les conversions
- pour les images couleur, la couleur est gérée comme une dimension, ce qui rend la manipulation des pixels laborieuse
- pas de gestion de la résolution ou des meta-informations des images
- l'indexation des pixels à partir de 1 n'est pas très heureuse, notamment pour les conversions de coordonnées
- les transformations géométriques d'images ne sont pas intuitives, et semblent difficile à étendre au 3D ou à d'autres types de transformations

L'idée de ce document est d'étudier si la création d'une (ou plusieurs ?) classes images permettraient d'améliorer la manipulation des images sous Matlab.

1.2 Wishlist

Autres éléments de la « wish-list » :

- gestion intégrée des ROI
  - setROI(...), avec comme arg soit une image binaire, soit un polygone/polyèdre. Par défaut, toute l'image.
- fonction sum, mean, hist... opérant directement sur la totalité des pixels (ou des pixels de la ROI).

1.3 Terminologie

index vecteur ligne contenant les coordonnées entières d'un pixel selon chaque dimension

position vecteur ligne contenant les coordonnées (en virgule flottante) d'un point dans l'espace physique

2 Nature des images

On se restreint pour le moment aux images matricielles.

Les images autres que matricielles qui pourraient être traitées dans un deuxième temps :

- vectorielles (ensemble de primitives géométriques)
- piles d'images, avec rotations différentes pour chaque pile

2.1 Dimension

Les images les plus typiques sont des images 2D. On manipule régulièrement des images 3D, plus rarement des images 4D. On peut avoir à manipuler des images en fonction du temps, ce qui rajoute une dimension. Idéalement il faudrait aussi inclure les images 1D afin d'être complet.

2.2 Types de pixels

On veut pouvoir manipuler des images de dimension 2 (classiques), 3 (souvent), 4 (plus rare), 1 (pour des profils). Le type de pixel peut varier. On a d'une part des données scalaires :

- binaire
- niveaux de gris, codés sur 8, 12, 16... bits
- intensités, stockées sur des flottants en simple ou double précision
- labels, stockés en général en 16 bits
- autre : une carte d'orientation, une valeur de périmètre... -> peut être codé avec un type double.

On manipule aussi couramment des pixels représentés non plus par une, mais par un ensemble de valeurs :

- complexe, par exemple pour le résultat d'une transformation de Fourier
- couleur (RGB, HSV...)
- spectral

La solution la plus simple sous Matlab pour gérer les types est de passer par la surcharge des opérateurs. Par exemple, on peut utiliser une classe Spectrum pour gérer les spectres.

2.3 Calibration spatiale

L'approche dans le logiciel ITK est d'intégrer les informations de calibration spatiale à la structure de données image. Idée à garder a priori.

Les deux infos à garder absolument :

- la taille du pixel dans chaque dimension
- la position du premier pixel

Ces deux informations sont en unité métrique, et correspondent à un tableau 1D avec autant d'éléments que la dimension de l'image.

Pour l'orientation, des outils comme ITK passent par la matrice des cosinus. A voir pour une version ultérieure. On peut aussi imaginer un offset : l'origine ne correspond pas au pixel d'indice (0,0), mais à un autre (le pixel central, un autre coin...).

2.4 Information de visualisation

En gros, comment représenter les pixels de l'image. Il s'agit d'une aplication qui à une valeur de l'image (NDG, double, complexe...) associe une valeur d'affichage (NDG ou RGB).Cela comprend :

- pour une image en NDG, la palette de couleur utilisée.
- pour une image d'intensité affichée en niveaux de gris, la dynamique d'affichage (min et max)
- pour une image multivariée (spectre, complexe, couleur...), une fonction qui détermine la couleur d'affichage en fonction d'un vecteur.

2.5 Meta-données

Quelques infos qu'il est parfois bon d'avoir sous la main :

- nom du fichier
- nom de l'image

- min et max des valeurs des pixels (pour images scalaires)
- résolution, pour avoir une correspondance entre coord pixels et coord physiques.
- nom des axes
- unité de la valeur des pixels

Matlab utilise une fonction spéciale pour lire les infos d'un fichier et stocke le résultat dans une structure.

3 Opérations souhaitées

On a besoin d'opération de base pour accéder aux valeurs des pixels, éventuellement les modifier, et connaître les informations générales de l'image (taille, nature des éléments...). On aimerait aussi avoir des fonctions qui se comportent de la même manière quelle que soit la dimension (moyenne, min, max... de l'image).

3.1 Taille de l'image

Fonctions de base pour connaître l'image.

size

Renvoie la taille de l'image, dans un vecteur ligne dont la longueur est égale à la dimension de l'image.

dim

Renvoie le nombre de dimensions -> reprendre ndims qui est casse-pieds pour les tableaux 1D. partir sur outerDimension ?

innerDimension (innerDim)

Renvoie le nombre de coordonnées pour lesquelles on a plus de 1 coupes.

outerDimension (outerDim)

Renvoie la dimension dans laquelle est inclue l'image. Est supérieur ou égal à la innerDimension.

3.2 Accès et modification des valeurs de l'image

pixel

Renvoie un pixel, ou un ensemble de pixels, sous la forme d'un scalaire, ou d'un tableau à plusieurs dimensions.

subImage

Extrait une sous-image, soit de même dimension, soit de dimension réduite. Le résultat est encore un objet image.

3.2.1 Utilisation de subsref ?

Permet :
- renvoyer la valeur d'un pixel à une position donnée
- crop image en spécifiant une plage.

3.2.2 Itérateurs sur pixels ?

Il serait pratique d'avoir une fonction pour itérer sur les pixels de l'image. Voir si gérable en pratique avec Matlab.

Principe :

– une classe Iterator

En fait, si le gain est intéressant en C++ via la gestion de pointeur, le gain semble négligeable sous Matlab.

3.2.3 Statistiques de base

Ces fonctions renvoient des valeurs calculées sur l'ensemble des pixels de l'image.

mean renvoie la valeur moyenne

min renvoie la valeur minimum

max renvoie la valeur maximum

median renvoie la valeur médiane

var renvoie la variance des valeurs

std renvoie l'écart-type des valeurs de l'image

3.2.4 Opération arithmétiques sur l'ensemble des pixels

plus ajoute une constante à chaque pixel

minus enlève une constante à chaque pixel

times multiplie chaque pixel par une constante

divide divise chaque pixel par une constante

3.2.5 Opérations entre deux images

plus additionne les valeurs des pixels de deux images, qui doivent avoir des pixels de même type

minus soustrait les valeurs les pixels de deux images, qui doivent avoir des pixels de même type

3.3 Calibration spatiale

Il nous faut quelques méthodes de conversion entre les deux systèmes de coordonnées utilisés :

– le système des indices, entre 0 et le nombre d'éléments dans la direction correspondante

– le système physique, en unité utilisateur

On peut imaginer les fonctions suivantes :

getSpacing/setSpacing pour changer l'espacement entre les pixels ou voxels

getOrigin/setOrigin pour changer la position spatiale du premier élément du tableau

getUnitName

pointToIndex conversion entre les coordonnées physiques et les coordonnées index

indexToPoint conversion entre les coordonnées index et les coordonnées physiques

3.4 Filtrage

3.4.1 Transformations géométriques basiques

Modification de la géométrie par rotation, symétries...

flip

Renverse une des dimensions

rotate90

Rotation de 90 degrés autour d'un des axes

3.5 Affichage

On veut pouvoir afficher une image avec différents niveaux de zoom. Choix entre 2 possibilités.

3.5.1 Pixels entre les bornes

Pour chaque pixel, on affiche un rectangle de la couleur du pixel entre les coordonnées (i, j) et $(i+1, j+1)$. Méthode classique de représentation des images sur écran matriciel.

L'avantage est que le coin supérieur de l'image est situé aux coordonnées $(0, 0)$, et que l'image occupe à l'écran un rectangle dont la position se calcule facilement : $[0; M_i] \cdot zoom$, où M_i est le nombre de pixels dans la dimension i.

Pour convertir les coordonnées écran en coordonnées pixel, il suffit de tronquer, après avoir divisé par le facteur de zoom.

3.5.2 Pixel centré sur les bornes

L'idée est de positionner le centre de chaque pixel aux coordonnées physiques du pixel. C'est ce qui est utilisé dans ITK.

L'avantage est que l'on a une meilleure correspondance entre les coordonnées physiques et le centre des pixels, ce qui est préférable si on veut mixer différentes images ou alors des images avec des objets vectoriels (résultats de segmentation par exemple).

Un pixel sera affiché dans un rectangle de coordonnées :

$$\prod_{i=0,\ldots,d} \left[x_i + o_i - \frac{w_i}{2}; x_i + o_i + \frac{w_i}{2} \right]$$

où les x_i sont les coordonnées du pixel, o_i l'origine de la grille, et w_i la taille de la grille selon chaque dimension.

Pour convertir les coordonnées écran en coordonnées pixel, l'opération semble un peu plus compliqué que dans le cas d'un pixel dans les mailles de la grille, mais reste abordable. De plus, sous Matlab on peut déléguer à la gestion des axes.

3.5.3 Conclusion

Il vaut mieux choisir des pixels centrés sur la grille.

4 Réflexions sur la conception

4.1 Multiplicité des images

On a besoin de jongler entre des images qui diffèrent principalement selon :
- le type de données utilisées
- la dimension
- le type de stockage retenu (un tampon, un ensemble de fichier, une vue d'une autre image...)

Une possibilité est d'avoir une classe unique « Image », qui gère tous les cas.

Une autre possibilité est de jouer le polymorphisme, et d'avoir une interface « Image », puis des classes concrètes qui implémentent les méthodes.

Pour le type de données, le plus simple est peut-être tout de même de partir sur un nombre restreint de types de bases (cf DIP lib).

4.2 Identification des pixels

On utilise trois types de coordonnées pour identifier les pixels :
- les coordonnées spatiales calibrées (position)
- les indices dans chaque axe (index)
- les indices non arrondis, si on veut interpoler et que l'on cherche les voisins (continuous index)

Dans ITK, ces trois types d'accès sont définis dans la classes ImageFunction.

5 Ébauche d'implémentation

5.1 Classe Image

Classe principale, qui sera peut-être complétée par des classes utilitaires (LUT, élément structurant...). Représente une image matricielle de dimension arbitraire.

5.1.1 Champs de classe

data Contient le tableau de données de l'image. Pour éviter les changements d'index de matlab, l'idée est de stocker l'image telle que les dimensions correspondent dans l'ordre à x, y, et éventuellement z. Pour des images multivariées (tensorielles), on rajoute une dimension, à la fin.

dataSize La dimension totale de l'image. Nécessaire de le stocker pour éviter les problèmes de dimensionnement.

dataType Le type des éléments stockés. A voir si on en a vraiment besoin, car peut être obtenu via une méthode du type : classname(this.data).

5.2 Classe Image « concrète »

Supposons qu'on ait une seule classe image. Pour incorporer toutes les informations, il faut inclure un certain nombre de champs.

5.2.1 Champs de classe

On regroupe en fonction du type d'information fournie. On a d'abord des données qui concernent le tableau de pixels/voxels :

data Un pointeur vers un tableau de données, ou vers une structure qui permet de récupérer un pixel pour un index (en coordonnées entières) donné.

dataSize La taille de l'image en nombre de pixels dans chaque dimension. Nécessaire de le stocker pour éviter les problèmes de dimensionnement dus à Matlab. Stocké dans un vecteur ligne. La longueur du vecteur donne le nombre de dimensions de l'image.

dataType le type des données stockées. Peut être déduit directement du tableau de données.

On peut stocker les infos nécessaire pour l'interprétation et l'affichage des valeurs

lut/dynamic Pour l'affichage. Stocké sous forme de tableau $2^G \times 3$.

minValue, maxValue peuvent être utiles pour certains calculs. Nécessitent d'être actualisés si on change les valeurs de certains pixels. à réfléchir.

On a ensuite des informations de calibration spatiale, un ensemble de données permettant le passage entre les coordonnées physique et l'index

origin les coordonnées physique du premier pixel de l'image. Stocké dans un vecteur ligne de d éléments.

spacing l'espacement, en unités physique, entre deux éléments successifs

unitName le nom de l'unité physique (microns, millimètres, pixels...)

Enfin, on aime garder quelques meta-informations

infos une structure de données contenant le nom de l'image, son fichier d'origine, l'auteur, la date de prise de vue...

5.2.2 Méthodes

getInfos renvoie les meta-données associées à l'image.

getPixel renvoie la valeur du pixel à partir de son index (tableau de coordonnées entières indexées à 0)

show affiche l'image dans la fenêtre courante

5.3 Interpolation

Un aspect particulièrement développé sous ITK est la gestion des transformations et des interpolations d'image. Une abstraction comparables consisterait à considérer plusieurs classes.

5.3.1 EvaluationFunction

Classe qui permet d'accéder à une valeur en fonction d'une position spatiale.

5.3.2 InterpolatedImage

Classe qui permet d'évaluer la valeur d'une image en fonction d'une position spatiale. Il s'agit d'une classe abstraite (ou d'une interface) qui sera spécialisée en fonction des méthodes d'interpolation.

5.3.3 BackTransformedImage

Classe qui permet d'évaluer la valeur d'une image après avoir transformé la position spatiale. Contient un pointeur vers une image interpolée, et un pointeur vers la transformation.

5.4 Classe utilitaire Images

Une classe pour fournir des méthodes statiques telles que le chargement d'une image, ou quelques filtres standards.

read pour charger une image depuis un fichier

write pour sauver les données image dans un ou plusieurs fichiers.

6 Conclusions

Pour le moment à l'état de réflexions.