Projet Problèmes Inverses CC Tomographie 2021

(kerry.gallagher@univ-rennes1.fr)

Introduction

Pour un modèle composé de blocs discrets, le problème de la tomographie peut être écrit comme une somme linéaire du problème en avant pour les cellules de rayon j à k

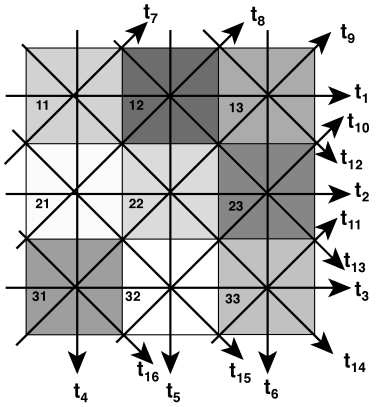
$$t_j = \sum_{i=1}^{K} l_{i,j} s_i$$

avec $l_{i,j}$ la longueur du trajet du rayon j dans le bloc i, qui a le paramètre du modèle s_i i.

Nous écrivons ensuite ceci comme une équation vecteur-matrice, **d** = **Gm**. Ici, **d** est le vecteur de données Nx1 des temps de parcours observés (t), **m** est le vecteur modèle Kx1 des valeurs de lenteur (s), et **G** est une matrice NxK qui transforme **m** vers **d**.

La solution est donnée comme $\mathbf{m} = \mathbf{G}^{-g}\mathbf{d}$ où $\mathbf{G}^{-g}$ est l'inverse généralisé, calculé en utilisant la décomposition de la valeur singulière (SVD).

Le but de ce petit projet est d'écrire un code général pour un problème de tomographie linéaire 2D en utilisant la décomposition des valeurs singulières (c'est-à-dire la matrice inverse généralisée). Nous voulons considérer le problème tomographique général, pour lequel nous voulons pouvoir changer les dimensions du modèle 2D (p.ex. **Nx x Ny**) et effectuer différents tests de résolution. La matrice **G** dépend du nombre de rayons et de les géométries de leurs chemins. Par exemple, nous pouvons supposer une distribution générale du trajet des rayons comme ci-dessous (pour **Nx** = **Ny** = 3).



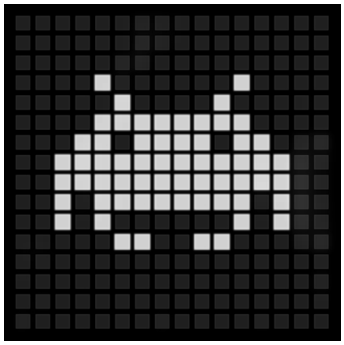
(1) Ecrivez un code général pour calculer la matrice **G** avec la géometrie de rayons comme ddécrit dessus, pour un modèle de taille **Nx * Ny** (où N**x** est le nombre de lignes, **Ny** est le nombre de colonnes, et le code devrait permettre le cas où **Nx** est différent de **Ny**). Utilisez une taille de bloc de largeur = hauteur = 1. Écrivez une explication de votre algorithme (si vous le souhaitez, vous pouvez numéroter les rayons différemment de celle montrée ci-dessus).

(2) En utilisant **Nx** = **Ny** = 7, testez la reconstruction d'un modèle de pic (un dans lequel le bloc central est 1 et tous les autres ont une valeur de 0). Comparez les reconstructions en utilisant des données parfaites, et des données avec différents niveaux de bruit (sigma = 0.1, 0.5, 1.0). Que remarquez-vous (par rapport aux solutions, et le 'data fit')?

(Le modèle Spike peut être comme ... myspike = zéros (Nx, Ny); Image (sol (Nx / 2 + 1), sol (Ny / 2) +1) = 1;)

(3) Répétez (2) et (3) avec **Nx** = **Ny** = 11, 15, 21, et aussi essayez un cas pour lequel **Nx** > **Ny**, et **Nx** < **Ny**. Y a-t-il des différences dans la résolution du modèle de pic au fur et à mesure que le nombre de blocs augmente et comment expliquer les différences?

(4) Afin de tester le code général, on veut générer une image synthétique et essayer de le récupérer. Générez une simple image en pixels noir et blanc (vous pouvez prendre la première lettre de votre prénom ou utilisez l'image de l'envahisseur spatial ci-dessous) dans un modèle de cellule carrée (vous choisissez la taille du modèle mais faites au moins 15x15). Dans le cas d'une image binaire (noir et blanc), le paramètre du modèle aura la valeur 1 pour un carré noir et 0 pour un carré blanc. On veut également avoir la possibilité de choisir une structure de modèle plus complexe pour capturer des images telles que le SuperMario ci-dessous. Des images plus complexes peuvent être construites en utilisant un entier pour chaque couleur différente. Avec Matlab, vous pouvez charger une image avec imread (myimage) et la convertir en niveaux de gris avec greymyimage = rgb2gray (myimage). Ensuite, choisissez les chemins de rayons comme décrit précédemment et générez des données synthétiques en utilisant **d** = **Gm**. On veut faire l'inversion avec les données avec ou sans bruit.



Nous voulons avoir la possibilité d'enregistrer à la fois le modèle et les données dans un fichier et les recharger, alors assurez-vous que votre code peut le faire. Vous pouvez choisir le format pour la sortie que vous le souhaitez.
Inversez les données parfaites, puis réessayez avec des données bruitées (sigma = 0.1) et tracez les modèles reconstruits.

(5) On veut répéter (4) mais en utilisant une distribution aléatoire des orientations du trajet des rayons. Utilisez le même nombre de rayons que dans le cas précédent, c'est-à-dire quatre fois le nombre de cellules du modèle mais avec les chemins aléatoire. Comparez les résultats entre (4) et (5).

(6) Écrivez un court guide de l'utilisateur pour votre code. Vous devrez expliquer comment formater et saisir à la fois un modèle et des données selon les besoins, quelle que soit la manière dont vous avez défini le problème).

Envoyer les codes (y compris vos données de test / image), un court rapport avec les réponses aux questions et le guide d'utilisation court à kerry.gallagher@univ-rennes1.fr en pdf au plus tard 17h00, vendredi, 15 janvier 2021. Veuillez utiliser le sujet «TOMOGRAPHIE». Je vais (essayer d')exécuter vos codes avec mes propres images pour vérifier le fonctionnement des codes.
Vous pouvez travailler en binôme, sachant que la note sera la même pour les deux, sauf si vous me dites au contraire.

HOW TO INPUT IMAGES....either binary but in an obvious way, so we can read in (or create in the code) a matrix (Nx x Ny with integer or real values)... or load an bitmap image...
Note that sometimes Matlab/Scilab can be very slow with big images, so you may reduce the size, with an appropriate function, or by resampling the image (e.g. take every 10th pixel in X and Y).

For Matlab

```
myimage = imread('/Users/kerry/mario.png');
imagesc(myimage)
greymyimage = rgb2gray(myimage);
myimage = imresize(myimage,Nx,Ny) % Nx, Ny = desired size
```

For Scilab we need to install toolboxes for **IPD** (Image Processing Design), or **SIVP**, **SIP** (just for PC). They require **opencv**. For matlab like plot commands we can use the plotlib toolbox.

To install **opencv**, you can follow the videos on youtube…
For Mac
https://www.youtube.com/watch?v=U49CVY8yOxw
For Windows
https://www.youtube.com/watch?v=EcFtefHEEII
or
https://www.youtube.com/watch?v=ScAPinibluA

(SCI gives us the pathname for the working Scilab directory)

To install the toolboxes we can use **Atoms**
(details are given here http://wiki.scilab.org/ATOMS)
Type `"Scilab help Atoms"`
To launch the user interface (you need to be connected to the internet)

```
atomsGui();
```

To find a toolbox called module

```
atomsSearch("module name")
```

To install a module

```
atomsInstall("plotlib") // Matlab like commands
atomsInstall("IPD");
```

or

```
atomsInstall("IPD","user") …just for single user (do not need admin)
```

We need to quit Scilab to update the toolboxes. Once this is done, the toolboxes will be loaded automatically. They will also be present in the Scilab help menu.
To check what has been installed, we can type

```
disp( atomsGetInstalled() );
```

To load a toolbox manually, call:

```
atomsLoad("IPD");
```

Example code with the IPD toolbox.

```
myimage = ReadImage("/Users/kerry/Mario.png");
bwimage = RGB2Gray(myimage);
% Note these may be integer so need to convert to double for plotting
SSG = double(SSG);
figure();
ShowColorImage(myimage,'Color Image')   % this did not work on Mac 10.9
figure();
ShowImage(bwimage,'BW Image')  or Matplot(SSG)
```