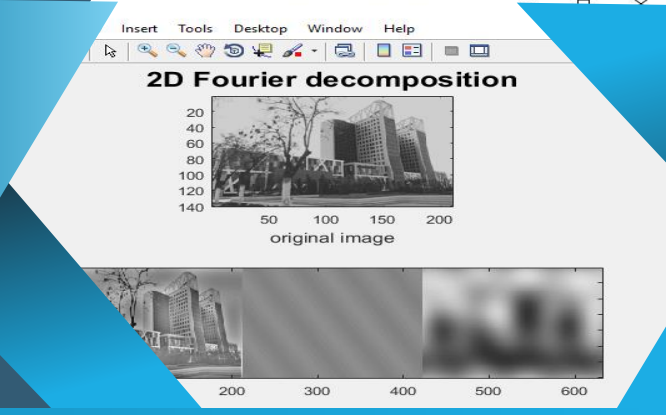
CHALLENGE

DECOMPOSITION & RECONSTRUCTION IMAGE AVEC FOURIER



Nom : Chihabeddine
Prénom : Merabet
Promotion : IGE45
Groupe : 03

Lire l'image :

Ce code Matlab commence par effacer toutes les variables de la mémoire à l'aide de la fonction `clear all` et nettoie la fenêtre de la console avec la fonction `clc`. Ensuite, il définit une variable `Vitesse` à une valeur de 20.

```
clear all;
clc;
Vitesse = 20;
```

La ligne suivante utilise la fonction `imread` pour charger une image nommée **"setif.jpg"** dans une variable appelée `image`.

```
image = imread('setif.jpg');
```

Enfin, une variable `scale` est définie avec une valeur de 0.5.

```
scale = 0.5;
```

Redimensionner l'image :

La première ligne de ce code utilise la fonction `imresize` pour redimensionner l'image `image` précédemment chargée en la mettant à l'échelle spécifiée par la variable `scale`. La fonction `mean` est d'abord appliquée à chaque canal de couleur de l'image à l'aide de `double(image),3)` pour calculer la moyenne des canaux de couleurs, puis l'image moyenne est redimensionnée. La nouvelle image est stockée dans la variable `image`.

```
image = imresize(mean(double(image),3), scale);
```

La deuxième ligne du code soustrait la moyenne de l'image `image` à l'aide de la fonction `mean(image(:))` pour normaliser l'image. Cela permet de centrer les valeurs de pixel de l'image autour de zéro.

```
image = image - mean(image(:));
```

La troisième ligne récupère la taille de l'image normalisée `image` à l'aide de la fonction `size(image)`, et stocke la taille dans la variable `N`.

```
N = size(image);
```

La quatrième ligne calcule la valeur minimale des pixels de l'image normalisée à l'aide de la fonction `min(image(:))`, et stocke cette valeur dans la variable `min_image`.

La cinquième ligne calcule la valeur maximale des pixels de l'image normalisée à l'aide de la fonction `max(image(:))`, et stocke cette valeur dans la variable `max_image`.

```
min_image = min(image(:));
max_image = max(image(:));
```

La dernière ligne du code calcule une plage dynamique pour l'image normalisée en utilisant les valeurs minimale et maximale des pixels. La plage est calculée de manière à étendre la plage des valeurs des pixels de l'image de 25 % à la fois au-dessus et en dessous de la plage d'origine. La plage étendue est stockée dans la variable `Range` sous la forme d'un tableau de deux éléments.

```
Range = [(1.25*min_image - 0.25*max_image), (1.25*max_image - 0.25*min_image)];
```

Code :

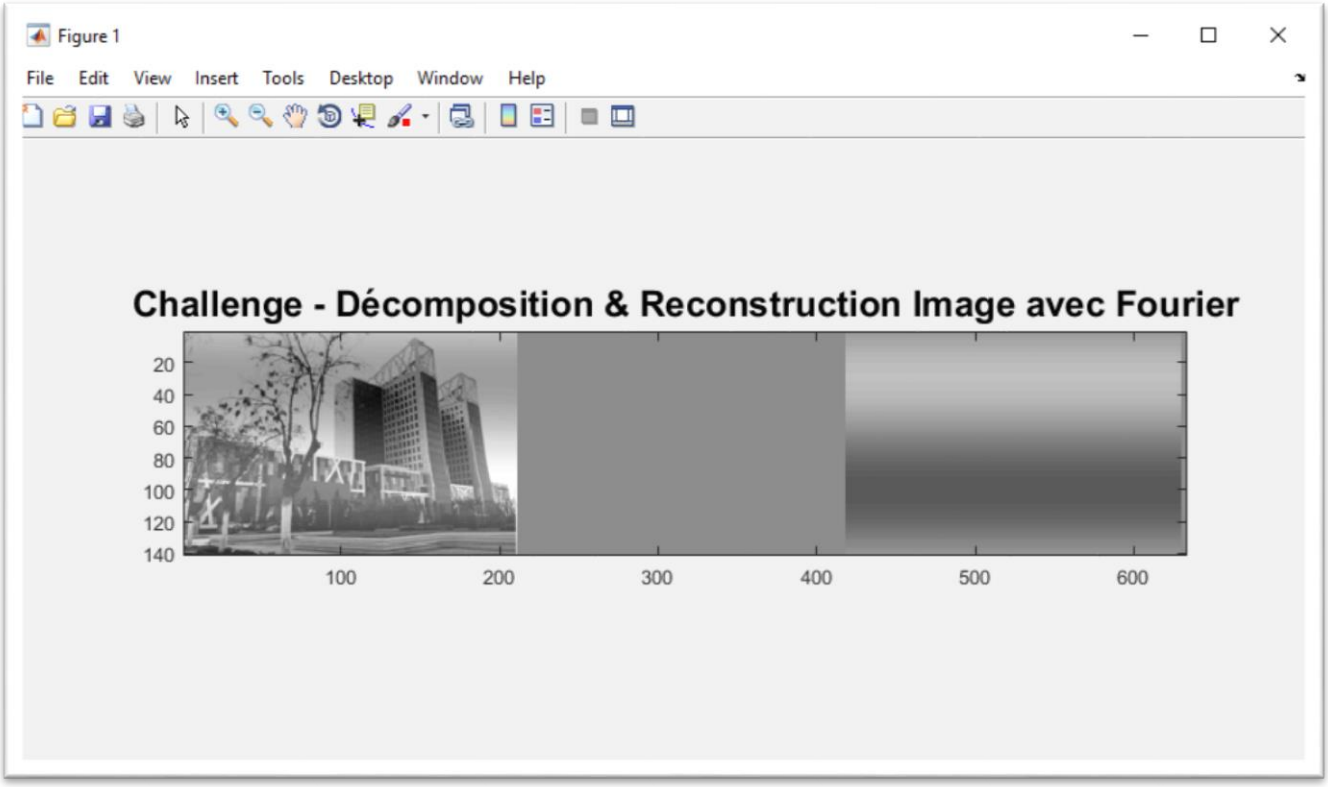
Ce code effectue une décomposition et une reconstruction d'image en utilisant la transformation de Fourier. La boucle `while` parcourt tous les éléments de l'image normalisée. À chaque itération, la boucle effectue les opérations suivantes :

- `FT_cur` est initialisé à 0. - Le nombre de vagues à utiliser pour la décomposition d'image est calculé en fonction de la position actuelle de l'itération dans l'image. Plus la position de l'itération est élevée, plus le nombre de vagues utilisées est grand. Une boucle `for` est utilisée pour sélectionner les `nombre_waves*2` coefficients de Fourier de la transformée de Fourier de l'image ayant les amplitudes les plus élevées. Ces coefficients sont stockés dans `FT_cur`. - La transformée de Fourier inverse de `FT_cur` est calculée à l'aide de la fonction `ifftn` et stockée dans `image_cur`.

- La variable `canvas` est utilisée pour stocker une représentation de l'image actuelle. La variable `canvas_view` est une copie de `canvas` qui sera modifiée pour afficher l'image mise à jour. La représentation de `canvas` est composée de trois parties. La variable `new_image` est mise à jour en ajoutant l'image mise à jour `image_cur`.

- L'itération est incrémentée par `nombre_waves`.

Le résultat de l'exécution de ce code sera une série d'images affichées progressivement, qui représentent l'image originale avec des détails de plus en plus fins ajoutés à chaque itération.



Puis :