Rapport :

Traitement de l'image sous MATLAB

Réalisé par :

- **Chaima Lgdour**
- **Youssef Ibourk**

L’objectif :

- L’objectif de cette introduction au traitement d’images sous Matlab est de présenter la notion d’image et d’effectuer des opérations simples d’analyse d’images telles que la rotation, l’effet noir et blanc...
- Le traitement d'images est un thème de recherche situé entre l'informatique et le traitement du signal.

MATLAB :

- MATLAB (« matrix laboratory ») est un langage de programmation de quatrième génération émulé par un environnement de développement du même nom ; il est utilisé à des fins de calcul numérique.
- Le logiciel MATLAB propose un environnement interactif de développement qui dispose d'un langage propre. L'application intègre un langage de haut niveau qui autorise l'exécution de tâches qui nécessitent une grande puissance de calcul.

Le sommaire :

Image mosaique:

Quand je dis mosaïque, je veux dire produire une image à partir de deux images. Il suffit de superposer l'une sur l'autre, plutôt que de fusionner une image par-dessus l'autre.

Image en niveaux de gris :

Une image de niveaux de gris autorise un dégradé de gris entre le noir et le blanc. En général, on code le niveau de gris sur un octet (8 bits) soit 256 nuances de dégradé. L'expression de la valeur du niveau de gris avec Ng = 256 devient: p(i,j) ∈[0, 255].

Image binaire :

Une image binaire est une image MxN où chaque point peut prendre uniquement la valeur 0 ou 1. Les pixels sont noirs (0) ou blancs (1). Le niveau de gris est codé sur un bit (Binary digIT). Dans ce cas, on revient au cas donné en I.1. avec Ng = 2 et la relation sur les niveaux de gris devient: p(i,j) = 0

La rotation de l'image:

Transformer une image par rotation c'est la faire tourner autour d'un point .En générale, la Rotation est définie par :

- Un centre
- Une angle de rotation
- Un sens de rotation (horaire ou antihoraire)

Version binaire

Afin d'appliquer l'effet binaire à l'image, nous utiliserons la fonction bien connue : *im2bw(I,level),* qui convertira n'importe quelle image en noir et blanc , voici la totalité de code qu'on a appliqué :

```
EE=imread("togg.jpg")
BW=im2bw(EE,90) ;
Figure
imshow(BW);
```

- ✓ **imread :** Lire le fichier image 'togg.jpg' (MATLAB Toolbox) et de placer son contenu dans la variable EE de type matrice
- ✓ **im2bw :**
  Convertit l'image EE en image binaire BW, en remplaçant tous les pixels de l'image d'entrée avec une luminance supérieure au niveau par la valeur 1 (blanc) et en remplaçant tous les autres pixels par la valeur 0 (noir).
  Cette plage est relative aux niveaux de signal possibles pour la classe de l'image. Ainsi, une valeur de niveau de 0,5 correspond à une valeur d'intensité à mi-chemin entre la valeur minimale et maximale de la classe.
- ✓ **Figure :** pour créer une nouvelle fenêtre graphique tout en conservant intacte ma fenêtre précédente.
- ✓ **Imshow :** Affiche l'image BW .
- ✓ **Title :** pour donner un titre à l'image.

version gris

Une fonction utile à connaître est la fonction de passage d'une image couleur à une image en niveaux de gris. Cette fonction rgb2gray permet d'obtenir une image en niveaux de gris en partant d'une image couleur. Dans certains cas, cette transformation est très utile

```
GS= rgb2gray(EE);
figure
imshow(GS);
title("GrayScale")
```

- ✓ La même chose ici on a utilisé la fonction rg2grap pour appliquer l'effet gris sur l'image EE. Puis, on a sélectionné une figure par la commande figure .Enfin, pour afficher l'image GS , on a tapé la commande imshow(GS) .

Rotation

imrotate(I,angle) fait pivoter l'image I de degrés d'angle dans le sens inverse des aiguilles d'une montre autour de son point central.

```
rot1=imrotate(EE,90);
figure
imshow(rot1);
title("90 degrees  rotation")
```

Pour faire pivoter l'image dans le sens des aiguilles d'une montre, spécifiez une valeur négative pour l'angle. imrotate rend l'image de sortie J suffisamment grande pour contenir l'intégralité de l'image pivotée. Par défaut, imrotate utilise l'interpolation du voisin le plus proche, définissant les valeurs des pixels dans J qui sont en dehors de l'image pivotée à 0.

version mosaique

Afin d'appliquer l'effet mosaique à l'image, on va utiliser 2 images EE et II , liserons les deux images en utilisant imread. Ensuite, on va vérifier les tailles des deux images. On va Convertir les images en deux de même taille si elles ne sont pas de la même taille en utilisant imresize (image, size) , ici **imresize** renvoie l'image EE et II qui ont le nombre de lignes et de colonnes spécifié par le vecteur à deux éléments [numrows numcols] et 1024 NaN désigne 1024 unité dans les colonnes et les lignes par défaut. Puis on va afficher l'image E qui a le même size que I maintenant .

```
II = imread("mosa5.jpg");
E = imresize(EE, [1024 NaN]);
I = imresize(II, [1024 NaN]);
imshow( E );
hold on;
h = imagesc( I ); % show the edge image
set( h, 'AlphaData', .3 ); % .5 transparency
title("Mosaic")
```

- ✓ **Hold on** :pour maintenir les images dans les axes actuels afin que la nouvelle image I ajoutés aux axes ne suppriment pas l'image existants E.
- ✓ **imagesc(I)** affiche les données du tableau I sous la forme d'une image qui utilise toute la gamme de couleurs de la palette de couleurs
- ✓ **set(H,Name,Value)** pour spécifie une valeur pour la propriété Name sur l'objet identifié par H. Utilisez des guillemets simples autour du nom de la propriété, par exemple, set(H, 'La couleur rouge'). Si H est un vecteur d'objets, alors set définit la propriété pour tous les objets. Si H est vide (c'est-à-dire []), set ne fait rien, mais ne renvoie pas d'erreur ou d'avertissement.

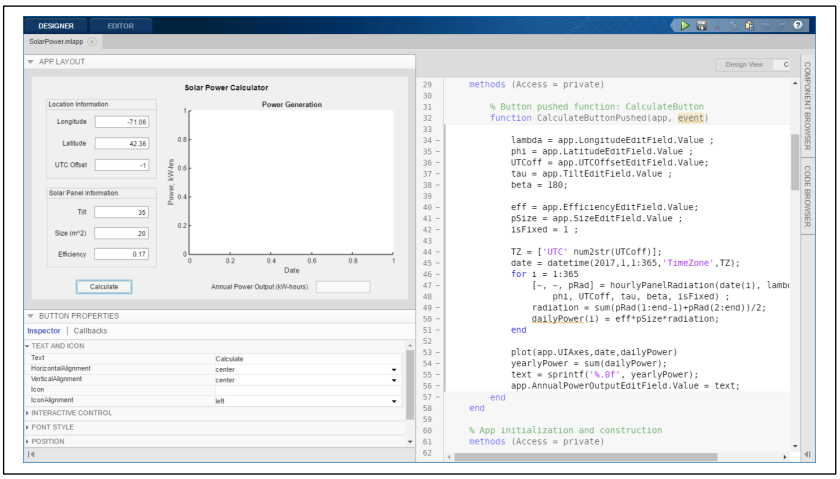
L’interface :

Il y a deux façons de réaliser une interface graphique Matlab :

- Utiliser l'outil graphique **Guide** qui facilite beaucoup la vie mais qui ne permet pas de comprendre vraiment ce que l'on fait. Il génère deux fichiers dont l'un ne peut pas être édité et l'autre pas très facile à déchiffrer. Il suffit d'avoir un petit problème dans l'un des deux fichiers et on n'arrive plus à ouvrir l'interface graphique.
- La deuxième méthode consiste à tout écrire soit même. C'est un peu plus compliqué, un peu plus long, mais on maîtrise tout. Le programme tient dans un seul script que l'on peut retoucher comme bon il nous semble. C'est cette méthode que j'ai décidé de décrire ici. Il ne s'agit pas d'un cours très détaillé mais plutôt d'un exemple permettant de monter une interface graphique rapidement.
- Dans le présent Project on va travailler par **App designer** car il vous permet également de créer des applications professionnelles sans avoir à être un développeur de logiciels professionnel. Faites glisser et déposez des composants visuels pour présenter la conception de votre interface utilisateur graphique (GUI) et utilisez l'éditeur intégré pour programmer rapidement son comportement.

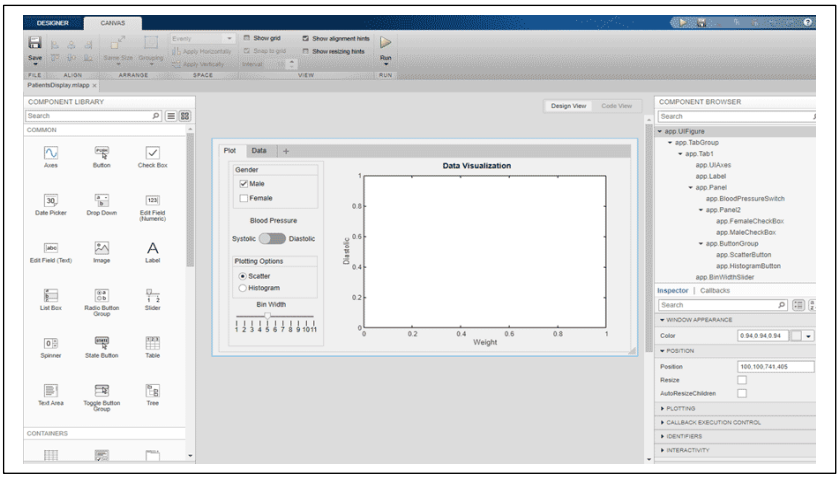
Building Your App :

App Designer intègre les deux tâches principales de la création d'applications : la présentation des composants visuels d'une interface utilisateur graphique (GUI) et la programmation du comportement de l'application. C'est l'environnement recommandé pour créer des applications dans MATLAB.



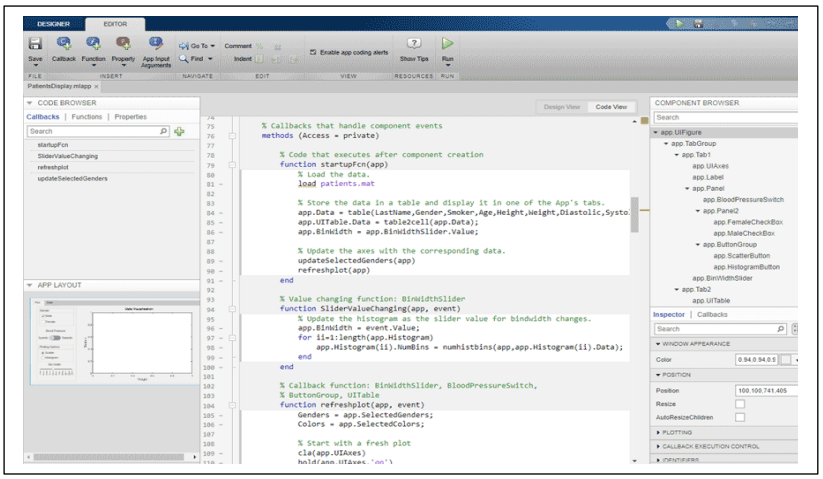
Design User Interface :

Vous pouvez faire glisser et déposez les composants visuels sur le canevas de conception et utilisez des conseils d'alignement pour obtenir une mise en page précise. App Designer génère automatiquement le code orienté objet qui spécifie la mise en page et la conception de l'application.



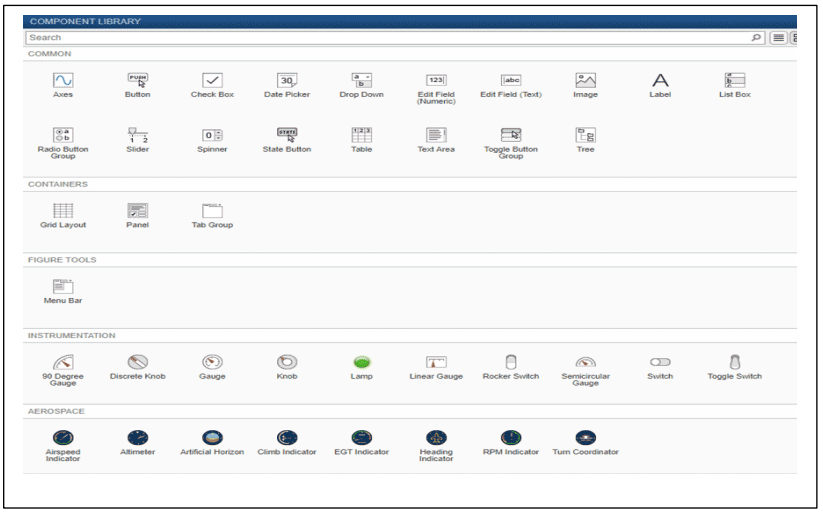
App Behavior

Vous pouvez utiliser la version intégrée de l'éditeur MATLAB pour définir le comportement de votre application. App Designer peut rechercher automatiquement les problèmes de codage à l'aide de l'analyseur de code. Vous pouvez afficher les messages d'avertissement et d'erreur concernant votre code pendant que vous l'écrivez et modifier votre application en fonction des messages.



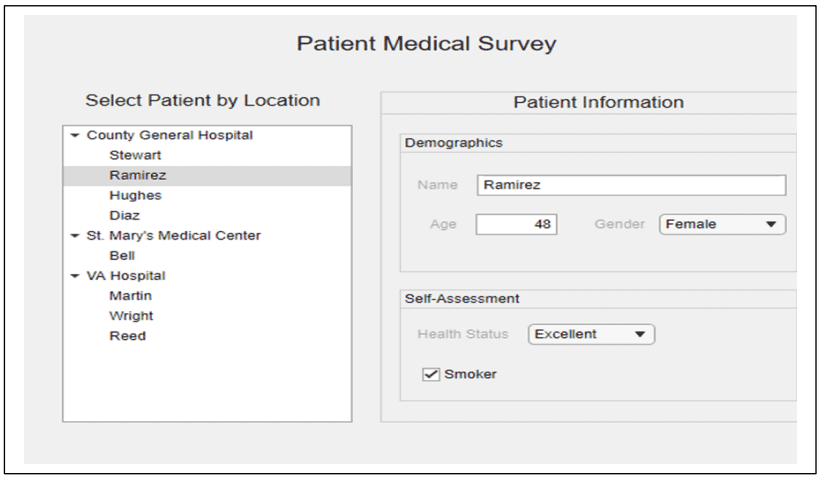
Component Library

Vous pouvez également Créer des applications avec des composants standard tels que des boutons, des cases à cocher, des arborescences et des listes déroulantes. App Designer fournit également des commandes telles que des jauges, des lampes, des boutons et des commutateurs qui vous permettent de reproduire l'apparence et les actions des panneaux d'instrumentation. Vous pouvez également utiliser des composants de conteneur, tels que des onglets, des panneaux et des dispositions de grille pour organiser votre interface utilisateur.

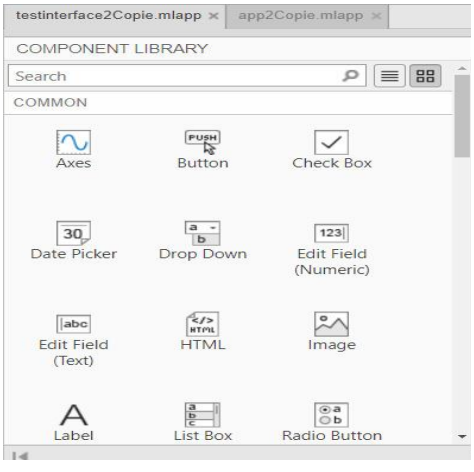


Component Interactions

Pour ajouteer des rappels de composants et des interactions personnalisées avec la souris et le clavier qui s'exécutent lorsqu'un utilisateur interagit avec votre application. Utilisez des tracés 2D et 3D, ainsi que des tableaux, dans votre application pour permettre aux utilisateurs d'explorer les données de manière interactive. Votre interface utilisateur.

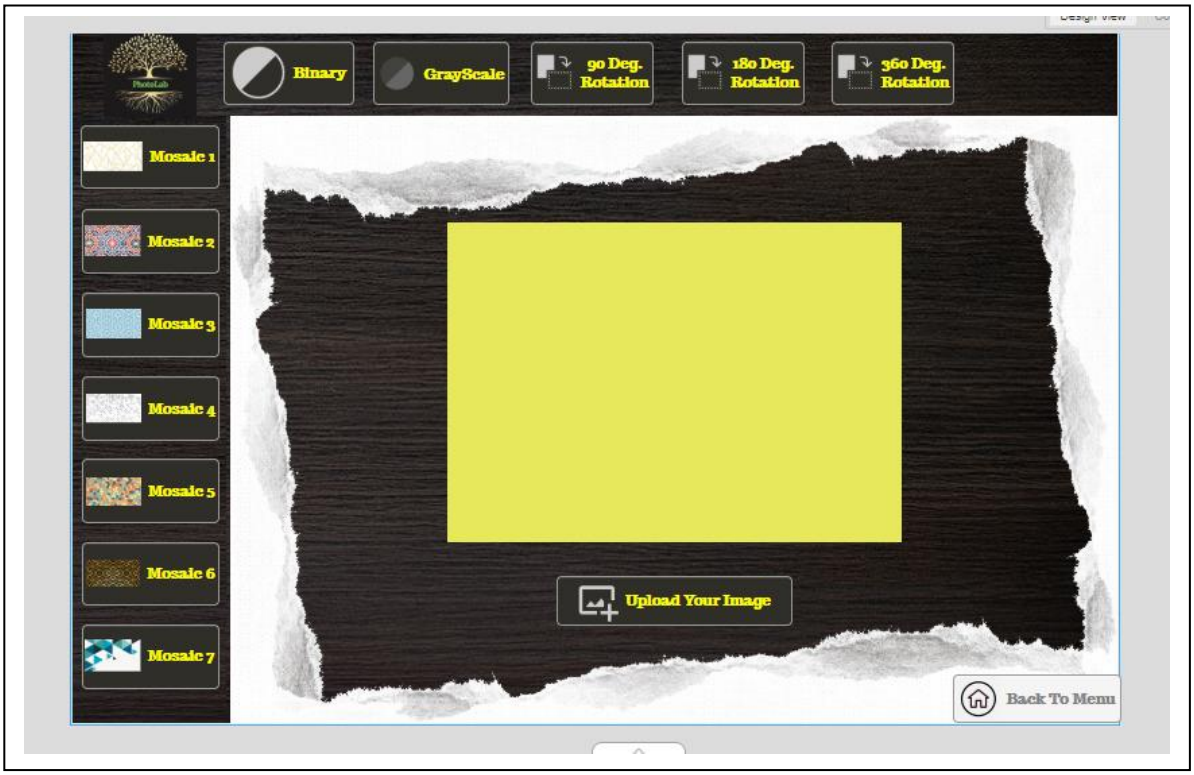


- En exploitant toutes ces composantes, on a créé deux interfaces, la première est une interface d'accueil qui contient une image de background avec un logo à l'intérieure et une zone de texte '' the best image editor''. En bas il y a un button qui nous ramène à la deuxième interface.
- La deuxième interface contient cinq buttons en haut et sept button de mosaique à gauche, un axis à l'intérieure dans lequel on va afficher notre 'image insérée à travers le button en bas ''insert your image ''.sans oublié également le logo et le button en bas à droite pour retourner à la page précédente.



La première interface :

La deuxième interface :



Ici, On a utilisé une fonction qui sera appelée si le contrôle est cliqué , c'est dire on a utilisé le Callbacks en cliquant sur le button qu'on veut puis le Botton droit de la souri ->callbacks , qui nous ramène directement vers la fonction associée à ce button , et ici on a entré le code qui va effectuer notre tâche spécifique , par exemple si on veut créer un button qui changer l'image insérée en une image noire et blanc .on doit suivi la même démarche de callbacks , ensuite on va insérer le code qui rend l'image en noir blanc (on l'a déjà expliquer dans les pages précédentes) à l'intérieur de la fonction. En fait, c'est la même chose pour les autres buttons. Voici nos callbacks associés à nos buttons.

L'image binaire :

```
function BinaryButton_2Pushed(app, event)
    global a;
    BW = im2bw(a,0.5);
    %figure
    imshow(BW);
end
```

L'image grise :

```
function GrayScaleButton_2Pushed(app, event)
    global a;
    GS= rgb2gray(a);
    %figure
    imshow(GS);
end
```

L'image mosaique :

```
function Mosaic1ButtonPushed(app, event)
    global a;
    EE = a;
    II = imread("mosa1.jpg");
    E = imresize(EE, [1024 NaN]);
    I = imresize(II, [1024 NaN]);
    imshow( E );
    hold on;
    h = imagesc( I ); % show the edge image
    set( h, 'AlphaData', .3 ); % .5 transparency
    % title("Mosaic")
end
```

La rotation :

```
function DegRotationButton_4Pushed(app, event)
    global a;
    rot1=imrotate(a,90);
    %figure
    imshow(rot1);
    title("90 degrees rotation")
end
```

- Pour qu'on puisse sélectionner une image lorsqu'on cliquera sur le button ''upload your Image'' on va utiliser le code à droite qui contient la fonction principale uigetfile() . Cette fonction nous permet de choisir l'image qu'on veut insérer à travers notre ordinateur.
- La variable globale 'a' c'est la variable dans laquelle on va stocker notre image pour l'afficher et la modifier par la suite.
- La deuxième image désigne le code d'affichage de l'image qu'on a déjà insérer dans l'axis. Cette image ca va être affiché directement et non pas dans un axis parce que cette manière d'affichage nous donne plus d'avantage par exemple enregistrer , copier.. …

Sélection de l'image

```
function UploadYourImageButtonPushed2(app, event)
    global a;
    [filename, pathname] = uigetfile('*.*', 'Pick an Image');
    filename=strcat(pathname,filename);
    a=imread(filename);
    imresize(a,[512 NaN]);
    imshow(a,'Parent',app.UIAxes);
end
```

L'affichage de l'image:

```
% Create UIAxes
app.UIAxes = uiaxes(app.UIFigure);
title(app.UIAxes, '')
xlabel(app.UIAxes, '')
ylabel(app.UIAxes, '')
app.UIAxes.AmbientLightColor = 'none';
app.UIAxes.PlotBoxAspectRatio = [1.8375 1 1];
app.UIAxes.GridColor = 'none';
app.UIAxes.MinorGridColor = 'none';
app.UIAxes.XColor = 'none';
app.UIAxes.XTick = [];
app.UIAxes.YColor = 'none';
app.UIAxes.YTick = [];
app.UIAxes.ZColor = 'none';
app.UIAxes.Color = 'none';
app.UIAxes.BackgroundColor = [0.9098 0.9098 0.3647];
app.UIAxes.Position = [350 169 424 298];
```

Pour la liaison entre les interfaces, on va taper une fonction qui porte le nom de l'interface qu'on veut, dans le callbacks du button qui nous amènera à la deuxième interface lorsque cliquons dessus. par exemple, pour lier l'interface ''app2copie'' (la première interface) avec la 2 eme interface à travers le button ''start editing '' on a entré une fonction nommé 'testinterface2copie() dans la fonction du button . et la même chose pour retourner de l'interface 2 vers 1 juste on tapé la fonction 'app2copie()'' dans le callback de button ''back to menu ''.

```
function BackToMenuButtonPushed2(app, event)
    app2Copie();
end
```


```
function BackToMenuButtonPushed2(app, event)
    app2Copie();
end
```


En ce qui concerne la police on a déjà installé la police 'trocchi' dans le pc, et pour changer la police d'un bouton ou d'un texte on a utilisé la composante à droite