Documentation du programme d'interface pour la manip DSPI

1. Architecture générale

Le programme est organisé en modules indépendants, chacun ayant une responsabilité spécifique. Il est basé sur la bibliothèque lensepy et les modules spécifiques à l'application en suivent la structure.

Les modules n'ont pas besoin d'être exécutés séparément. Ils sont appelés directement par le fichier "my_app.py". Le fichier "camera_dspi_widget" est utilisé pour définir l'affichage de l'interface et est appelé directement par "my_app.py"

2. Description des modules

Structure typique d'un module :

- **Fichier XML** : pour la configuration et l'initialisation du module
- **Fichier Controller** : gestion de l'application et communication entre les différents éléments du module
- **Fichier View** : affichage et interaction utilisateur
- **Fichier Models** : modèles physiques, algorithmes de calcul, pilotage des dispositifs externes (caméras par exemple)

Modules de l'application :

- **Module "basler_dspi" :** gère la connexion avec la caméra, le réglage de ses paramètres et l'affichage de l'image.
  Ce module possède un fichier "camera.ini" contenant les paramètres par défaut de la caméra. Ce fichier est nécessaire au bon fonctionnement du module et ne nécessite pas d'être modifier.
- **Module "my_module"** : s'occupe des traitements d'image spécifiques au TP DSPI. Ce module est incomplet car il manque certaines fonctionnalités. De plus, pour des raisons de simplicité les calculs des algorithmes de phase shift et les commandes à envoyer à la carte de commande du piézo ont été intégrés directement au fichier Controller. Pour s'aligner sur l'architecture des modules de lensepy et gagner en clarté il faudrait déplacer ces fonctions vers le fichier Models (actuellement inutilisé). Ce point n'empêche pas le fonctionnement du module en l'état actuel.

3. Fonctionnement général

L'interface fonctionne avec des widgets (bibliothèque PyQt6) interdépendants permettant d'afficher les différentes parties de la fenêtre. Ces widgets appartiennent à des classes définies soit dans la bibliothèque lensepy, soit dans les modules spécifiques de l'application. Dans les Controller de "basler_dspi" et "my_module", ces widgets sont nommés suivant leur position dans la fenêtre ('top_right', 'bottom_left', ...).

Le module "my_module" nécessite que le module "basler_dspi" ait été lancé d'abord car ce dernier initie la connexion avec la caméra et sa configuration.

Les modules de cette application utilisent un thread (PyQt6) pour l'affichage des images enregistrées par la caméra. Cela permet de faire tourner le code d'acquisition des images en parallèle du code principal, et donc de gérer les interactions entre l'utilisateur et l'interface même lorsque l'affichage se met à jour.

4. Prérequis et installation

- Vérifier que lensepy est installé (version 2.1.1)
- Il vaut mieux éviter de modifier l'arborescence des fichiers du code pour éviter des erreurs lors d'appels de fonctions ou de classes
- Si le nom des fichiers est modifié, il faudra probablement apporter des modifications dans les codes pour que les références au fichier modifié soient bonnes

5. Utilisation

Guide d'utilisation :

- Exécuter le fichier "my_app.py" pour lancer l'application
- Une fois l'interface ouverte, il faut cliquer sur le bouton `'basler_dspi'` pour lancer la connexion avec la caméra
- L'image de la caméra s'ouvre alors avec un histogramme, et des curseurs pour régler le temps d'exposition et le black level. On peut aussi définir une zone d'intérêt (ROI) en cliquant sur des points de l'image et en utilisant le bouton `'activate_roi_button'`
- Pour avoir accès aux traitements d'image, il faut cliquer sur le bouton `'my_module'`, ce qui affiche l'interface des traitements et une seconde fenêtre qui contient l'image en direct de la caméra. Le widget de traitement affiche initialement la même chose que la deuxième fenêtre.

- Les boutons à droite permettent de lancer les différents traitements disponibles (différence d'image, phase shift, ...). Le bouton `'Stop Processing'` permet de remettre les traitements à zéro
- On peut toujours repasser d'une interface à l'autre en appuyant sur les boutons en bas à gauche de la fenêtre

6. Améliorations possibles

- Ajout des fonctionnalités manquantes (mesure de vibration par phase shift par exemple)
- Régler le problème du traitement en continu. Pour le moment, il n'est pas possible de réaliser les opérations de traitement trop lourdes en temps réel (phase shift notamment) car le thread d'acquisition semble aller trop vite et rentrer en conflit avec le traitement.
- Passer les calculs de traitement d'image et de communication avec la carte de la cale piézo dans le fichier "my_module_models.py"