Répertoire des fonctions (notice de codage)

**Dossier main**

**Fonction load\_config (main.py)**

* **Description :** Charge la configuration depuis le fichier "config.toml".
* **Modules utilisés :** tomllib (non standard, utilisé pour charger des fichiers TOML).

**Fonction signal\_handler (main.py)**

* **Description :** Attend une entrée utilisateur pour continuer l'exécution du programme après une interruption de signal.**Fonction calibrate\_from\_config (main.py)**
* **Description :** Effectue une calibration en utilisant les points spécifiés dans la configuration.
* **Fonctions appelées :** calibrate (définie dans le fichier arms.py).

**Fonction get\_coordinates\_converter (main.py)**

* **Description :** Obtient un convertisseur de coordonnées en fonction de la configuration et de la calibration du bras.
* **Fonctions appelées :** calibrate\_from\_config (définie dans le fichier main.py).

**Fonction edges\_from\_config (main.py)**

* **Description :** Applique la détection de bords à une image en utilisant les paramètres de configuration.
* **Fonctions appelées :** rgb2edges, sort\_edges (définies dans le fichier main.py).
* **Modules utilisés :** skimage, cv2 (utilisé dans save\_contours\_image).

**Fonction draw\_image (main.py)**

* **Description :** Dessine les bords d'une image sur le robot.
* **Fonctions appelées :** get\_coordinates\_converter, edges\_from\_config, save\_contours\_image, draw\_edges (définies dans le fichier main.py).

**Fonction save\_contours\_image (main.py)**

* **Description :** Sauvegarde une image avec des contours dessinés.
* **Modules utilisés :** cv2 (OpenCV).

**Fonction photomaton\_meta\_loop (main.py)**

* **Description :** Boucle principale pour le mode photomaton, capture des images, les dessine et les envoie au robot.
* **Fonctions appelées :** photomaton\_loop, draw\_image (définies dans le fichier main.py).
* **Modules utilisés :** cv2, CoordinatesConverter (non standard, doit être défini ailleurs).

**Fonction get\_coordinates\_from\_user (main.py)**

* **Description :** Obtient des coordonnées depuis l'utilisateur à l'aide de la boîte de dialogue Tkinter.
* **Modules utilisés :** tkinter (Tkinter).

**Dossier arms**

**Fonction get\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm à partir de l'adresse IP du robot.
* **Modules utilisés :** xarm.wrapper.XArmAPI (XArmAPI), utils (définie dans le fichier utils.py).

**Fonction get\_photomaton\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm spécifique au mode photomaton.
* **Fonctions appelées :** get\_arm, enable\_arm (définies dans le fichier arms.py).
* **Modules utilisés :** Aucun.

**Fonction get\_big\_drawing\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm spécifique au grand dessin.
* **Fonctions appelées :** get\_arm, enable\_arm (définies dans le fichier arms.py).
* **Modules utilisés :** Aucun.

**Fonction enable\_arm (arms.py)**

* **Description :** Active le mouvement du bras en mode position.

**Fonction calibrate (arms.py)**

* **Description :** Calibre le bras du robot à partir de trois points spécifiés.
* **Modules utilisés :** utils.find\_surface, numpy.

**Fonction calibrate\_from\_dimensions (arms.py)**

* **Description :** Calibre le bras du robot à partir de la dimension du dessin.
* **Fonctions appelées :** calibrate (définie dans le fichier arms.py).

**Fonction draw\_edge (arms.py)**

* **Description :** Dessine les bords en déplaçant le bras du robot.
* **Modules utilisés :** numpy.

**Fonction wait\_for\_input (arms.py)**

* **Description :** Attend une entrée utilisateur pour continuer.

**Fonction draw\_edges (arms.py)**

* **Description :** Dessine une série de bords en minimisant les mouvements inutiles.
* **Fonctions appelées :** draw\_edge (définie dans le fichier arms.py).

**Dossier robot\_control\_app**

**Classe RobotControlApp (robot\_control\_app.py)**

* **Description :** Interface graphique Tkinter pour le contrôle manuel du robot.
* **Fonctions appelées :** move\_axis (définie dans la classe RobotControlApp).
* **Modules utilisés :** tkinter.

**Fonction move\_axis (robot\_control\_app.py)**

* **Description :** Déplace l'axe spécifié du robot dans la direction donnée.

**Fonction run (robot\_control\_app.py)**

* **Description :** Lance la boucle principale Tkinter.
* **Modules utilisés :** tkinter.

**Dossier photomaton**

**Fonction process\_frame (photomaton.py)**

* **Description :** Applique un traitement à un cadre vidéo capturé.
* **Modules utilisés :** cv2 (OpenCV).

**Fonction get\_frame (photomaton.py)**

* **Description :** Capture et traite un cadre vidéo.
* **Fonctions appelées :** process\_frame (définie dans le fichier photomaton.py).
* **Modules utilisés :** cv2 (OpenCV).

**Fonction show (photomaton.py)**

* **Description :** Affiche une image dans une fenêtre avec une légende.
* **Modules utilisés :** cv2 (OpenCV).

**Fonction drawing\_in\_progress (photomaton.py)**

* **Description :** Affiche un message indiquant que le dessin est en cours.
* **Fonctions appelées :** show (définie dans le fichier photomaton.py).
* **Modules utilisés :** cv2 (OpenCV).

**Fonction optimize\_path (photomaton.py)**

* **Description :** Optimise le chemin en regroupant les segments adjacents.
* **Modules utilisés :** numpy.

**Fonction sort\_edges (photomaton.py)**

* **Description :** Trie les bords pour minimiser les mouvements inutiles du robot.

**Fonction image\_thresholding (photomaton.py)**

* **Description :** Applique des étapes de traitement d'image pour améliorer la détection des bords.
* **Modules utilisés :** cv2 (OpenCV).

**Dossier utils**

**Fonction get\_config (utils.py)**

* **Description :** Charge la configuration à partir du fichier "config.toml".
* **Fonctions appelées :** tomllib.load (non standard, utilisé pour charger des fichiers TOML).

**Fonction calibrate\_from\_user\_input (utils.py)**

* **Description :** Interface utilisateur pour calibrer le bras du robot.
* **Fonctions appelées :** RobotControlApp.run (définie dans le fichier robot\_control\_app.py).
* **Modules utilisés :** tkinter (Tkinter).

**Dossier main**

**Fonction load\_config (main.py)**

* **Description :** Charge la configuration depuis le fichier "config.toml" en utilisant la bibliothèque tomllib.

**Fonction signal\_handler (main.py)**

* **Description :** Attend une entrée utilisateur pour continuer l'exécution du programme après une interruption de signal.

**Fonction calibrate\_from\_config (main.py)**

* **Description :** Effectue une calibration du bras en utilisant les points spécifiés dans la configuration.
* **Fonctions appelées :** calibrate (définie dans le fichier arms.py).

**Fonction get\_coordinates\_converter (main.py)**

* **Description :** Obtient un convertisseur de coordonnées en fonction de la configuration et de la calibration du bras.
* **Fonctions appelées :** calibrate\_from\_config (définie dans le fichier main.py).

**Fonction edges\_from\_config (main.py)**

* **Description :** Applique la détection de bords à une image en utilisant les paramètres de configuration.
* **Fonctions appelées :** rgb2edges, sort\_edges (définies dans le fichier main.py).
* **Modules utilisés :** skimage, cv2 (utilisé dans save\_contours\_image).

**Fonction draw\_image (main.py)**

* **Description :** Dessine les bords d'une image sur le robot en utilisant les fonctions auxiliaires appropriées.
* **Fonctions appelées :** get\_coordinates\_converter, edges\_from\_config, save\_contours\_image, draw\_edges (définies dans le fichier main.py).

**Fonction save\_contours\_image (main.py)**

* **Description :** Sauvegarde une image avec des contours dessinés en utilisant la bibliothèque cv2 (OpenCV).

**Fonction photomaton\_meta\_loop (main.py)**

* **Description :** Boucle principale pour le mode photomaton, capture des images, les dessine et les envoie au robot.
* **Fonctions appelées :** photomaton\_loop, draw\_image (définies dans le fichier main.py).
* **Modules utilisés :** cv2, CoordinatesConverter (non standard, doit être défini ailleurs).

**Fonction get\_coordinates\_from\_user (main.py)**

* **Description :** Obtient des coordonnées depuis l'utilisateur à l'aide de la boîte de dialogue Tkinter.
* **Modules utilisés :** tkinter (Tkinter).

**Dossier arms**

**Fonction get\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm à partir de l'adresse IP du robot.
* **Modules utilisés :** xarm.wrapper.XArmAPI (XArmAPI), utils (définie dans le fichier utils.py).

**Fonction get\_photomaton\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm spécifique au mode photomaton.
* **Fonctions appelées :** get\_arm, enable\_arm (définies dans le fichier arms.py).

**Fonction get\_big\_drawing\_arm (arms.py)**

* **Description :** Obtient une instance de l'API XArm spécifique au grand dessin.
* **Fonctions appelées :** get\_arm, enable\_arm (définies dans le fichier arms.py).

**Fonction enable\_arm (arms.py)**

* **Description :** Active le mouvement du bras en mode position.

**Fonction calibrate (arms.py)**

* **Description :** Calibre le bras du robot à partir de trois points spécifiés.
* **Modules utilisés :** utils.find\_surface, numpy.

**Fonction calibrate\_from\_dimensions (arms.py)**

* **Description :** Calibre le bras du robot à partir de la dimension du dessin.
* **Fonctions appelées :** calibrate (définie dans le fichier arms.py).

**Fonction draw\_edge (arms.py)**

* **Description :** Dessine un bord en déplaçant le bras du robot.
* **Modules utilisés :** numpy.

**Fonction wait\_for\_input (arms.py)**

* **Description :** Attend une entrée utilisateur pour continuer.

**Fonction draw\_edges (arms.py)**

* **Description :** Dessine une série de bords en minimisant les mouvements inutiles.
* **Fonctions appelées :** draw\_edge (définie dans le fichier arms.py).

**Dossier robot\_control\_app**

**Classe RobotControlApp (robot\_control\_app.py)**

* **Description :** Interface graphique Tkinter pour le contrôle manuel du robot.
* **Fonctions appelées :** move\_axis (définie dans la classe RobotControlApp).

**Fonction move\_axis (robot\_control\_app.py)**

* **Description :** Déplace l'axe spécifié du robot dans la direction donnée.

**Fonction run (robot\_control\_app.py)**

* **Description :** Lance la boucle principale Tkinter.

**Dossier photomaton**

**Fonction process\_frame (photomaton.py)**

* **Description :** Applique un traitement à un cadre vidéo capturé.
* **Modules utilisés :** cv2 (OpenCV).

**Fonction get\_frame (photomaton.py)**

* **Description :** Capture et traite un cadre vidéo.
* **Fonctions appelées :** process\_frame (définie dans le fichier photomaton.py).

**Fonction show (photomaton.py)**

* **Description :** Affiche une image dans une fenêtre avec une légende.
* **Modules utilisés :** cv2 (OpenCV).

**Fonction drawing\_in\_progress (photomaton.py)**

* **Description :** Affiche un message indiquant que le dessin est en cours.
* **Fonctions appelées :** show (définie dans le fichier photomaton.py).

**Fonction optimize\_path (photomaton.py)**

* **Description :** Optimise le chemin en regroupant les segments adjacents.
* **Modules utilisés :** numpy.

**Fonction sort\_edges (photomaton.py)**

* **Description :** Trie les bords pour minimiser les mouvements inutiles du robot.

**Fonction image\_thresholding (photomaton.py)**

* **Description :** Applique des étapes de traitement d'image pour améliorer la détection des bords.

**Dossier utils**

**Fonction get\_config (utils.py)**

* **Description :** Charge la configuration à partir du fichier "config.toml" en utilisant la bibliothèque tomllib.
* **Fonctions appelées :** tomllib.load (non standard, utilisé pour charger des fichiers TOML).

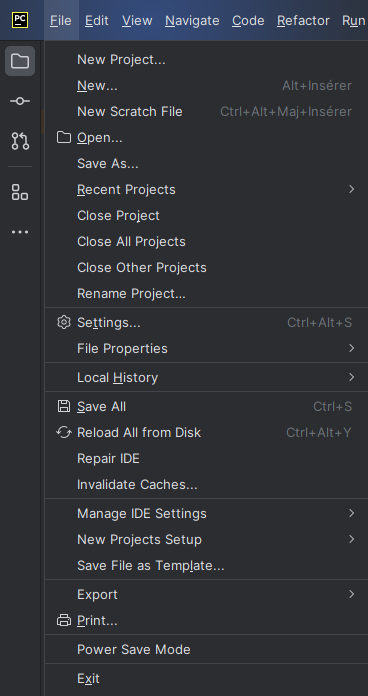
**Fonction calibrate\_from\_user\_input (utils.py)**

* **Description :** Interface utilisateur pour calibrer le bras du robot.
* **Fonctions appelées :** RobotControlApp.run (définie dans le fichier robot\_control\_app.py).
* **Modules utilisés :** tkinter (Tkinter).

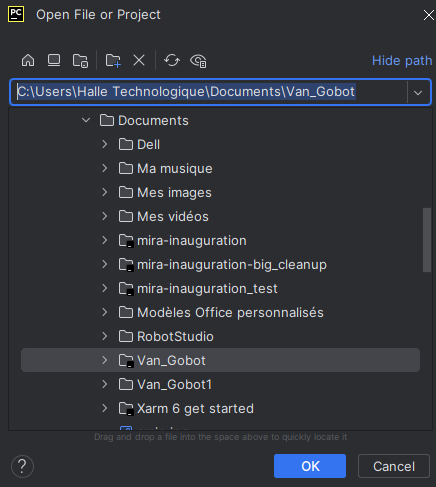
Notice de lancement

Ouvrir pycharm

File – Open

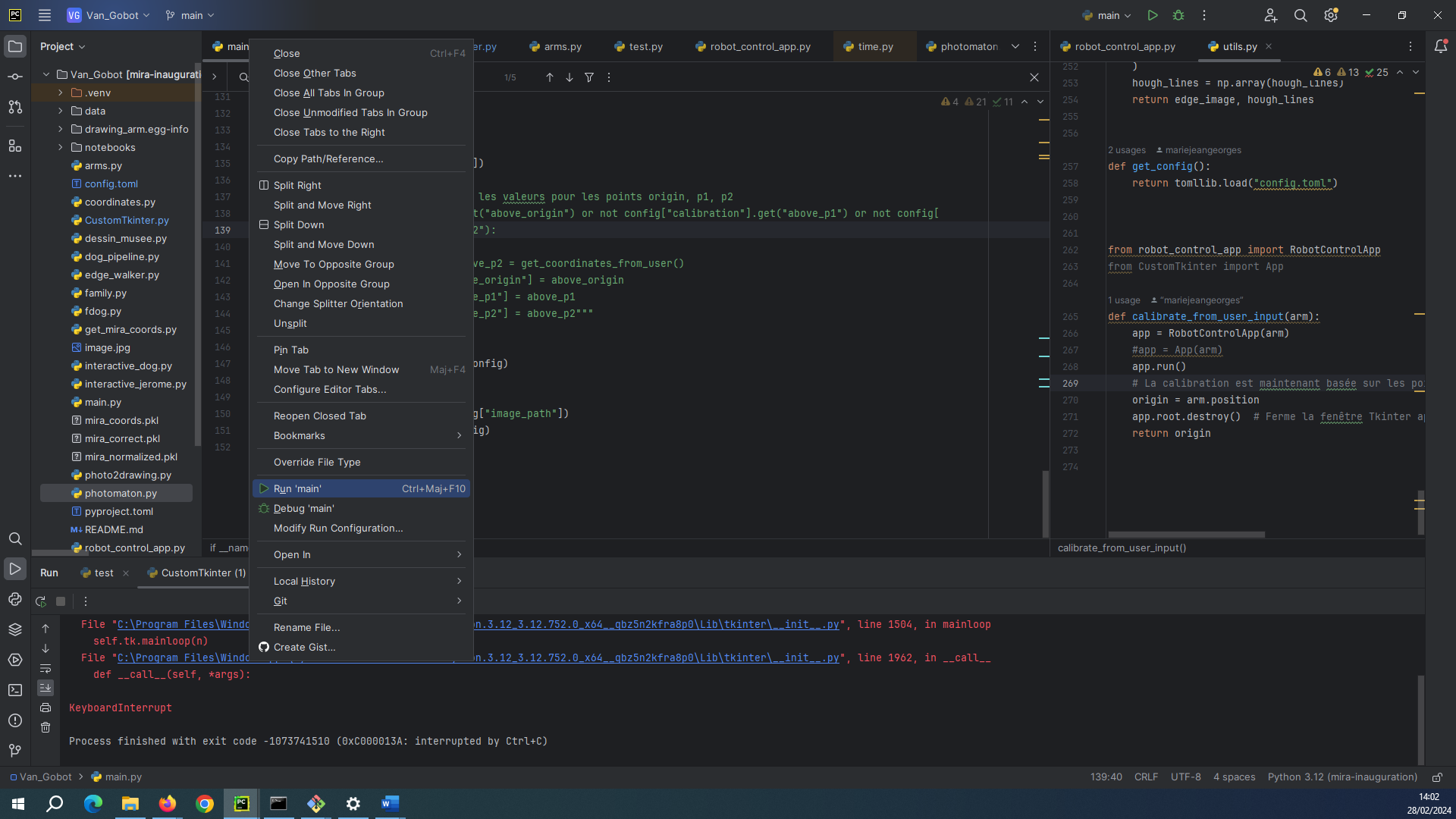


Ouvrir le dossier Van\_Gobot



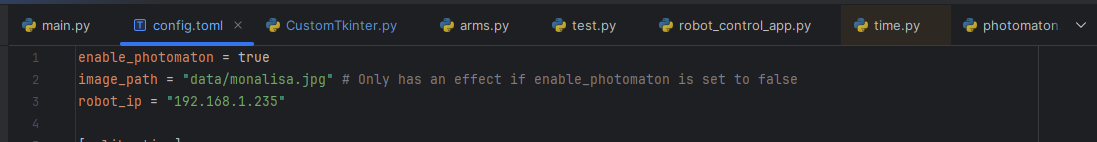
Pour lancer il faut aller dans le main puis run, ou clique droit puis run.

Mais avant de run il faire quelques vérifications



Vérifier que l’adresse ip robot\_ip du robot est la bonne

Il faut aussi déplacer le robot avec le stylo au dessus du point origine et rentre les bonne coordonnées dans above\_origin, puis faire above\_p1=above\_origin+[-130, 0, 0] et above\_p2=above\_origin+[0, 120, 0] (sauf si on souhaite travailler avec des dimensions différentes (dans le dossier config.toml



Puis pour prendre une photo avec la webcam, il faut enable\_photomaton=true



Cela ouvre cette fenêtre, quand on est satisfait de la photo, on peut appuyer sur c (sur le clavier), ce qui lance un compte à rebours de 10s puis prend la photo et lance le dessin

Sinon on peut en desactivant le photomaton (enable\_photomaton=false), on peut choisir de le faire sur un fichier photo (format .jpg) et en plaçant la photo dans le dossier data (qui se trouve lui-même dans le dossier van\_gobot)

Notice CAO