Documentation du Code Python

Boyer Apolline

August 1, 2024

Contents

1 Classes et Fonctions

1.1 Classe `StereoCalibration`

- **Description**: Cette classe gère les paramètres et les processus de calibration stéréoscopique pour deux caméras.
- **Attributs**:
  - `cam_mats`: Dictionnaire des matrices de caméra (paramètres intrinsèques) pour les caméras gauche et droite.
  - `dist_coefs`: Dictionnaire des coefficients de distorsion pour les caméras gauche et droite.
  - `rot_mat`: Matrice de rotation.
  - `trans_vec`: Vecteur de translation.
  - `e_mat`: Matrice essentielle.
  - `f_mat`: Matrice fondamentale.
  - `rect_trans`: Dictionnaire des transformations de rectification pour les caméras gauche et droite.
  - `proj_mats`: Dictionnaire des matrices de projection pour les caméras gauche et droite.
  - `disp_to_depth_mat`: Matrice de conversion de la disparité à la profondeur.
  - `valid_boxes`: Dictionnaire des boîtes englobantes des pixels valides pour les caméras gauche et droite.
  - `undistortion_map`: Dictionnaire des cartes de dédistorsion pour les caméras gauche et droite.
  - `rectification_map`: Dictionnaire des cartes de rectification pour les caméras gauche et droite.

1.1.1 Méthode `__str__`

- **Description**: Retourne une représentation en chaîne de caractères des attributs de la classe.
- **Arguments**: Aucun.
- **Retourne**: **str** - Représentation en chaîne des attributs de l'objet.
- **Exemple d'utilisation**:

```
1 print(calibration)
```

1.1.2 Méthode `__init__`

- **Description**: Initialise les paramètres de calibration pour les caméras stéréo.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibration = StereoCalibration()
```

1.1.3 Méthode `save_data`

- **Description**: Enregistre les paramètres de calibration dans des fichiers .npy et .csv.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibration.save_data()
```

1.1.4 Méthode `rectify`

- **Description**: Rectifie les images stéréoscopiques en utilisant les cartes de dédistorsion et de rectification.
- **Arguments**:
  - `frames`: **list** - Liste des images des caméras gauche et droite.
- **Retourne**: **list** - Liste des images rectifiées.
- **Exemple d'utilisation**:

```
rectified_frames = calibration.rectify(frames)
```

1.1.5 Méthode `load_data`

- **Description**: Charge les paramètres de calibration à partir de fichiers .npy dans un répertoire spécifié.
- **Arguments**:
  - `directory`: **str** - Répertoire contenant les fichiers de paramètres.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibration.load_data('calibration_data')
```

1.2 Classe `Calibrator`

- **Description**: Gère le processus de calibration pour une paire de caméras stéréoscopiques.
- **Attributs**:
  - `image_count`: Nombre d'images de calibration utilisées.
  - `row`: Nombre de coins internes dans les rangées du tableau de calibration.
  - `column`: Nombre de coins internes dans les colonnes du tableau de calibration.
  - `square_size`: Taille des carrés du tableau de calibration en cm.
  - `image_size`: Taille des images de calibration en pixels.
  - `corner_coordinates`: Coordonnées 3D des coins du tableau de calibration.
  - `object_points`: Liste des coordonnées des coins réels trouvés dans chaque image.
  - `image_points`: Dictionnaire des coordonnées des coins trouvés dans les images pour les caméras gauche et droite.

1.2.1 Méthode `__init__`

- **Description**: Initialise les paramètres pour le calibrage des caméras.
- **Arguments**:
  - `row`: **int** - Nombre de coins internes dans les rangées du tableau.
  - `column`: **int** - Nombre de coins internes dans les colonnes du tableau.
  - `square_size`: **float** - Taille des carrés du tableau en cm.
  - `image_size`: **tuple** - Taille des images de calibration en pixels.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibrator = Calibrator(row=6, column=9, square_size=2.5,
    image_size=(1920, 1080))
```

1.2.2 Méthode `corner_detect`

- **Description**: Détecte les coins du tableau de calibration dans une paire d'images.
- **Arguments**:
  - `image_pair`: **tuple** - Tuple contenant les images gauche et droite.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibrator.corner_detect((left_image, right_image))
```

1.2.3 Méthode `calibrate_camera`

- **Description**: Calibre les deux caméras et détermine leurs matrices liées.
- **Arguments**: Aucun.
- **Retourne**: **StereoCalibration** - Instance de la classe `StereoCalibration` avec les paramètres calibrés.
- **Exemple d'utilisation**:

```
calibration = calibrator.calibrate_camera()
```

1.2.4 Méthode `calibration_process`

- **Description**: Effectue le processus de calibration en lisant les images de calibration et en appelant la méthode de calibration.
- **Arguments**:
  - `nbr_photo`: **int** - Nombre d'images de calibration à traiter.
  - `image_folder`: **str** - Répertoire contenant les images de calibration.
- **Retourne**: **StereoCalibration** - Instance de la classe `StereoCalibration` avec les paramètres calibrés.
- **Exemple d'utilisation**:

```
calibration = calibrator.calibration_process(nbr_photo=20,
    image_folder='calibration_images')
```

1.3 Classe `DualCameraCapture`

- **Description**: Capture des images stéréoscopiques en utilisant deux caméras et fournit des outils pour afficher et valider ces images.
- **Attributs**:
  - `left_cam_id`: ID de la caméra gauche.
  - `right_cam_id`: ID de la caméra droite.
  - `preview_size`: Taille de l'aperçu des images capturées.
  - `preview_type`: Type d'aperçu.
  - `capture_delay`: Délai avant la capture d'image.
  - `interval`: Intervalle entre les captures d'images.

1.3.1 Méthode `__init__`

- **Description**: Initialise les paramètres pour la capture d'images avec deux caméras.
- **Arguments**:
  - `left_cam_id`: **int** - ID de la caméra gauche.
  - `right_cam_id`: **int** - ID de la caméra droite.
  - `preview_size`: **tuple** - Taille de l'aperçu.
  - `preview_type`: **Preview** - Type d'aperçu.
  - `capture_delay`: **float** - Délai avant la capture d'image.
  - `interval`: **float** - Intervalle entre les captures d'images.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
capture = DualCameraCapture(left_cam_id=0, right_cam_id=1,
    preview_size=(800, 600), capture_delay=2, interval=5)
```

1.3.2 Méthode `capture_and_save_image`

- **Description**: Capture et sauvegarde une image depuis la caméra spécifiée.
- **Arguments**:
  - `picam_id`: **int** - ID de la caméra à utiliser.
  - `filename`: **str** - Nom du fichier pour sauvegarder l'image.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
capture.capture_and_save_image(picam_id=0, filename='left_image.png
    ')
```

1.3.3 Méthode `display_images`

- **Description**: Affiche les images capturées à partir des fichiers spécifiés.
- **Arguments**:
  - `left_filename`: **str** - Nom du fichier de l'image gauche.
  - `right_filename`: **str** - Nom du fichier de l'image droite.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
capture.display_images(left_filename='left_image.png',
    right_filename='right_image.png')
```

1.3.4 Méthode `validate_images`

- **Description**: Valide si les images capturées sont acceptables.
- **Arguments**: Aucun.
- **Retourne**: **bool** - Retourne True si les images sont acceptables, sinon False.
- **Exemple d'utilisation**:

```
is_valid = capture.validate_images()
```

1.3.5 Méthode `capture_images`

- **Description**: Capture un nombre spécifié de paires d'images et les sauvegarde dans le dossier spécifié.
- **Arguments**:
  - `nbr_photos`: **int** - Nombre de paires d'images à capturer.
  - `image_folder`: **str** - Dossier où sauvegarder les images.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
capture.capture_images(nbr_photos=10, image_folder='captured_images')
```

1.4 Classe `StereoVision`

Cette classe est utilisée pour capturer des images stéréo, calculer des cartes de disparité et de profondeur, et afficher les résultats.

1.4.1 Méthode `__init__`

- **Description**: Initialise les paramètres pour la vision stéréo.
- **Arguments**:
  - `cam_capture`: Instance de `DualCameraCapture` pour capturer les images.
  - `baseline`: Distance entre les caméras (en mètres).
  - `focale`: Focale de la caméra.
  - `block_size`: Taille du bloc pour la correspondance stéréo.
  - `P1`: Poids pour la régularisation des coûts d'assignation.
  - `P2`: Poids pour la régularisation des coûts d'assignation.
  - `min_disp`: Disparité minimale à considérer.
  - `max_disp`: Disparité maximale à considérer.

- `uniqueRatio`: Ratio d'unicité pour la correspondance stéréo.
- `speckleWindowSize`: Taille de la fenêtre pour filtrer les speckles.
- `speckleRange`: Plage de valeurs pour filtrer les speckles.
- `disp12MaxDiff`: Différence maximale entre les disparités gauche et droite.

- **Retourne**: Aucun.

1.4.2 Méthode `stereo_taking`

- **Description**: Capture et rectifie les images stéréo.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.stereo_taking()
```

1.4.3 Méthode `save_images`

- **Description**: Sauvegarde les images et la carte de disparité normalisée.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.save_images()
```

1.4.4 Méthode `depth_map_calcul`

- **Description**: Calcule la carte de disparité à partir des images rectifiées.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.depth_map_calcul()
```

1.4.5 Méthode `depth_calcul`

- **Description**: Calcule la profondeur pour chaque pixel à partir de la carte de disparité.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.depth_calcul()
```

1.4.6 Méthode `process_stereo`

- **Description**: Traite la carte de profondeur en utilisant `DepthMapProcessor`.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.process_stereo()
```

1.4.7 Méthode `capture_and_compute`

- **Description**: Capture les images, calcule la carte de disparité et la profondeur, puis place les résultats dans une file d'attente.
- **Arguments**:
  - `queue`: File d'attente pour transmettre les résultats entre les processus.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.capture_and_compute(queue)
```

1.4.8 Méthode `depth_map_display`

- **Description**: Affiche la carte de disparité et la profondeur à partir des résultats de la file d'attente.
- **Arguments**:
  - `queue`: File d'attente pour obtenir les résultats calculés.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.depth_map_display(queue)
```

1.4.9 Méthode `process_and_display`

- **Description**: Crée des processus pour la capture et le calcul des images, ainsi que pour l'affichage des résultats.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
stereo_vision.process_and_display()
```

1.5 Classe `TofCamera`

- **Description**: Cette classe gère la capture d'images à partir d'une caméra ToF, le traitement des données de profondeur et d'amplitude, ainsi que l'affichage des résultats.
- **Attributs**:
  - `cam`: Instance de `ArducamCamera` pour la caméra ToF.
  - `max_distance`: Distance maximale mesurable par la caméra (en mètres).
  - `frame`: Cadre actuel capturé par la caméra.
  - `amplitude_buf`: Tampon pour les données d'amplitude.
  - `depth_buf`: Tampon pour les données de profondeur.
  - `depth_normalized`: Carte de profondeur normalisée pour affichage.
  - `result_image`: Image résultante après traitement.
  - `n`: Compteur pour le nom des images sauvegardées.

1.5.1 Méthode `__init__`

- **Description**: Initialise la caméra ToF avec les paramètres de distance maximale.
- **Arguments**:
  - `max_distance`: **float** - Distance maximale mesurable par la caméra (en mètres).
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
tof_camera = TofCamera(max_distance=4)
```

1.5.2 Méthode `process_frame`

- **Description**: Traite le cadre capturé pour produire une image résultante en combinant les données de profondeur et d'amplitude.
- **Arguments**: Aucun.
- **Retourne**: **np.ndarray** - Image résultante après traitement.
- **Exemple d'utilisation**:

```
result_image = tof_camera.process_frame()
```

1.5.3 Méthode `capture_image`

- **Description**: Sauvegarde l'image résultante sous le nom `tof{n}.png`.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
tof_camera.capture_image()
```

1.5.4 Méthode `process_tof`

- **Description**: Traite la carte de profondeur en utilisant `DepthMapProcessor` pour analyser et extraire les contours.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
tof_camera.process_tof()
```

1.5.5 Méthode `continuous_display`

- **Description**: Capture et affiche les images en continu à partir de la caméra ToF, avec des options pour sauvegarder et traiter les images.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
tof_camera.continuous_display()
```

1.5.6 Méthode `cleanup`

- **Description**: Arrête et ferme la caméra, et détruit toutes les fenêtres OpenCV.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
tof_camera.cleanup()
```

1.5.7 Méthode `get_depth_buf`

- **Description**: Retourne le tampon de profondeur actuel.
- **Arguments**: Aucun.
- **Retourne**: **np.ndarray** - Tampon de profondeur.
- **Exemple d'utilisation**:

```
depth_buf = tof_camera.get_depth_buf()
```

1.5.8 Méthode `get_depth_normalized`

- **Description**: Retourne la carte de profondeur normalisée.
- **Arguments**: Aucun.
- **Retourne**: **np.ndarray** - Carte de profondeur normalisée.
- **Exemple d'utilisation**:

```
depth_normalized = tof_camera.get_depth_normalized()
```

1.6 Classe `DepthMapProcessor`

- **Description**: Cette classe gère le traitement des cartes de profondeur et de disparité, y compris la segmentation, le calcul des amplitudes moyennes, et le dessin des contours.
- **Attributs**:
  - `depth_map_original`: Carte de profondeur originale.
  - `depth_map_normalized`: Carte de disparité normalisée.
  - `pixel_min`: Nombre minimum de pixels non nuls pour considérer un segment (par défaut 15000).
  - `min_contour_area`: Aire minimale pour les contours à considérer (par défaut 10).

- `thresholds`: Liste des seuils pour la segmentation de la disparité.
- `kernel_size`: Taille du noyau pour les opérations morphologiques (par défaut 5).
- `dilate_iterations`: Nombre d'itérations pour la dilatation (par défaut 1).
- `erode_iterations`: Nombre d'itérations pour l'érosion (par défaut 2).
- `segmented_image`: Image segmentée après application des seuils.
- `contours`: Liste des contours trouvés.
- `mean_amplitudes`: Dictionnaire des amplitudes moyennes pour chaque contour.

1.6.1 Méthode `__init__`

- **Description**: Initialise la classe `DepthMapProcessor` avec les paramètres fournis.
- **Arguments**:
  - `depth_map`: **np.ndarray** - Carte de profondeur originale.
  - `disparity`: **np.ndarray** - Carte de disparité normalisée.
  - `pixel_min`: **int** - Nombre minimum de pixels non nuls pour considérer un segment (par défaut 15000).
  - `min_contour_area`: **int** - Aire minimale pour les contours à considérer (par défaut 10).
  - `thresholds`: **list** - Liste des seuils pour la segmentation de la disparité.
  - `kernel_size`: **int** - Taille du noyau pour les opérations morphologiques (par défaut 5).
  - `dilate_iterations`: **int** - Nombre d'itérations pour la dilatation (par défaut 1).
  - `erode_iterations`: **int** - Nombre d'itérations pour l'érosion (par défaut 2).
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
depth_map = np.zeros((480, 640))  # Exemple de carte de profondeur
disparity = np.zeros((480, 640))  # Exemple de carte de disparite
processor = DepthMapProcessor(depth_map, disparity)
```

1.6.2 Méthode `apply_morphological_operations`

- **Description**: Applique des opérations morphologiques (dilatation et érosion) à l'image spécifiée.
- **Arguments**:
  - `image`: **np.ndarray** - Image à traiter.

- **Retourne**: **np.ndarray** - Image après application des opérations morphologiques.
- **Exemple d'utilisation**:

```
processed_image = processor.apply_morphological_operations(image)
```

1.6.3 Méthode `calculate_mean_amplitude`

- **Description**: Calcule l'amplitude moyenne pour chaque contour spécifié.
- **Arguments**:
  - `contours`: **list** - Liste des contours trouvés dans l'image.
- **Retourne**: **dict** - Dictionnaire des amplitudes moyennes pour chaque contour.
- **Exemple d'utilisation**:

```
mean_amplitudes = processor.calculate_mean_amplitude(contours)
```

1.6.4 Méthode `find_and_draw_contours`

- **Description**: Trouve et dessine les contours dans l'image traitée.
- **Arguments**:
  - `processed_image`: **np.ndarray** - Image après les opérations morphologiques.
- **Retourne**: **np.ndarray** - Image avec les contours dessinés.
- **Exemple d'utilisation**:

```
image_with_contours = processor.find_and_draw_contours(
    processed_image)
```

1.6.5 Méthode `process_contour`

- **Description**: Traite les contours en appliquant des opérations morphologiques, en trouvant et en dessinant les contours, et en calculant les amplitudes moyennes pour les contours trouvés.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
processor.process_contour()
```

1.6.6 Méthode `process_disparity_image`

- **Description**: Traite l'image de disparité en la segmentant selon les seuils définis, puis en appliquant le traitement de contours sur chaque segment.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
processor.process_disparity_image()
```

1.7 Fonctions Utilitaires

1.7.1 Fonction `calculate_histogram`

- **Description**: Calcule l'histogramme des valeurs de pixels de l'image.
- **Arguments**:
  - `image`: **np.ndarray** - Image à analyser.
- **Retourne**: **np.ndarray** - Histogramme des valeurs de pixels.
- **Exemple d'utilisation**:

```
hist = calculate_histogram(image)
```

1.7.2 Fonction `count_non_zero_pixels_from_histogram`

- **Description**: Compte le nombre de pixels non nuls à partir de l'histogramme des valeurs de pixels.
- **Arguments**:
  - `hist`: **np.ndarray** - Histogramme des valeurs de pixels.
- **Retourne**: **int** - Nombre total de pixels non nuls.
- **Exemple d'utilisation**:

```
non_zero_count = count_non_zero_pixels_from_histogram(hist)
```

1.7.3 Fonction `plot_histogram`

- **Description**: Trace et affiche l'histogramme des valeurs de pixels.
- **Arguments**:
  - `title`: **str** - Titre du graphique.
  - `hist`: **np.ndarray** - Histogramme des valeurs de pixels.

- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
plot_histogram("Histogramme des Pixels", hist)
```

2 Fonctions

2.1 Fonction `folder_create`

- **Description**: Vérifie si un dossier existe, sinon il le crée.
- **Arguments**:
  - `folder`: **str** - Le chemin du dossier à vérifier/créer.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
folder_create('path/to/folder')
```

2.2 Fonction `file_create`

- **Description**: Crée un fichier du type spécifié dans un dossier donné (facultatif).
- **Arguments**:
  - `data`: **np.ndarray**, **list**, ou autre - Données à enregistrer dans le fichier.
  - `file_name`: **str** - Nom du fichier à créer (sans extension).
  - `file_type`: **str** - Type de fichier à créer ('csv', 'image', 'npy', etc.).
  - `folder_name`: **str** (optionnel) - Dossier dans lequel créer le fichier.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
file_create(data, 'example_image', 'png', 'path/to/folder')
file_create(data, 'example_array', 'npy')
file_create(data, [['header1', 'header2'], [1, 2]], 'csv', 'path/to
    /folder')
```

2.3 Fonction `show_image`

- **Description**: Affiche une image avec une colormap spécifiée.
- **Arguments**:
  - `title`: **str** - Titre de la fenêtre d'affichage.

- `image`: **np.ndarray** - Image à afficher.
- `cmap`: **str** (optionnel) - Colormap OpenCV à appliquer ('gray', 'jet', 'rainbow', etc.).

- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
show_image('Image Affichee', image, 'jet')
```

2.3.1 Liste des Colormaps

- **Colormaps disponibles**:
  - `autumn`
  - `bone`
  - `jet`
  - `winter`
  - `rainbow`
  - `ocean`
  - `summer`
  - `spring`
  - `cool`
  - `hsv`
  - `pink`
  - `hot`
  - `parula`
  - `magma`
  - `inferno`
  - `plasma`
  - `viridis`
  - `cividis`
  - `twilight`
  - `twilight_shifted`
  - `turbo`
  - `deepgreen`

3 Gestion des Caméras et des Processus

3.1 Fonction `calibrate_cameras`

- **Description**: Calibre les caméras en prenant des photos d'un échiquier et en utilisant un processus de calibration.
- **Arguments**:
  - `cam_capture`: Instance de `DualCameraCapture` pour capturer les images.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
calibrate_cameras(cam_capture)
```

3.2 Fonction `run_tof_camera`

- **Description**: Fonction pour exécuter la caméra ToF en continu et mettre à jour une queue avec les données de profondeur.
- **Arguments**:
  - `camera_queue`: File d'attente pour stocker les données de la caméra.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
run_tof_camera(camera_queue)
```

3.3 Fonction `run_stereo_vision`

- **Description**: Fonction pour exécuter la vision stéréo et retourner les résultats de la disparité et de la profondeur.
- **Arguments**: Aucun.
- **Retourne**: Tuple contenant la disparité normalisée et la profondeur.
- **Exemple d'utilisation**:

```
disparity_normalized, depth = run_stereo_vision()
```

3.4 Fonction `terminate_processes`

- **Description**: Termine les processus donnés.
- **Arguments**:
  - `processes`: Liste de processus à terminer.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
terminate_processes(processes)
```

3.5 Fonction `kill_zombie_processes`

- **Description**: Termine les processus zombies détectés sur le système.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
kill_zombie_processes()
```

3.6 Fonction `clean_temp_dirs`

- **Description**: Nettoie les répertoires temporaires spécifiés.
- **Arguments**:
  - `directories`: Liste de répertoires à nettoyer.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
clean_temp_dirs(['/tmp', '/var/tmp'])
```

3.7 Fonction `cleanup`

- **Description**: Effectue les opérations de nettoyage du système et libère la mémoire.
- **Arguments**: Aucun.
- **Retourne**: Aucun.
- **Exemple d'utilisation**:

```
cleanup()
```

4 Exécution du Script Principal

```
if __name__ == "__main__":
    cleanup()
    folder_create('data')
    folder_create('image')
    folder_create('corner')

    calib_choice = input("Voulez-vous calibrer les cameras (y/n) ? ").
       strip().lower()

    if calib_choice == "y":
        cam_capture = DualCameraCapture(left_cam_id=2, right_cam_id=1,
           preview_size=(840, 820))
        calibrate_cameras(cam_capture)
    elif calib_choice == "n":
        print("Les cameras ne seront pas calibrees.")
    else:
        print("Choix invalide. Veuillez entrer 'y' ou 'n'.")
        exit(1)

    # Initialisation des queues pour la communication entre processus
    camera_queue = multiprocessing.Queue()

    # Creation des processus
    tof_process = multiprocessing.Process(target=run_tof_camera, args=(
       camera_queue,))
    stereo_process = multiprocessing.Process(target=run_stereo_vision)

    # Demarrage des processus
    tof_process.start()
    stereo_process.start()

    processes = [tof_process, stereo_process]

    # Attente de la fin des processus
    try:
        # Maintenir les processus en vie
        while True:
            sleep(1)
    except KeyboardInterrupt:
        print("Interruption detectee. Arret des processus...")
    finally:
        # Arret des processus
        terminate_processes(processes)
        print("Tous les processus ont ete arretes.")
        cleanup()
        sys.exit(0)
```