Guide Téléopération avec Caméra SO-ARM 101

Phase 6 : Visualisation temps réel

Service Écoles-Médias (SEM) - DIP Genève

✅ Prérequis

- Phase 1 complétée (LeRobot installé)
- Phase 2 complétée (Servos configurés avec IDs 1-6)
- Phase 3 complétée (Calibration effectuée)
- Phase 4 complétée (Tests de contrôle validés)
- Phase 5 complétée (Téléopération fonctionnelle)
- Scripts SEM installés depuis GitHub
- Au moins 1 caméra USB disponible
- Support de fixation pour la caméra (imprimé 3D ou bricolage)

🎯 Objectif de cette phase

Pourquoi ajouter une caméra ? La visualisation en temps réel permet de voir ce que "voit" le robot pendant la téléopération. C'est l'étape préparatoire avant l'enregistrement de trajectoires pour l'apprentissage par imitation, où la caméra capturera les démonstrations visuelles.

L'ajout de la caméra permet de :

- Visualiser l'espace de travail pendant la téléopération
- Préparer l'enregistrement visuel des démonstrations
- Valider le positionnement optimal de la caméra
- Tester la qualité d'image et la latence

📋 Étape 1 : Installation physique de la caméra

Matériel nécessaire

Élément	Description	Alternative
Caméra USB	Webcam standard (720p minimum)	Webcam intégrée PC portable
Support	Pièce imprimée 3D	Trépied, pince, bricolage
Fixation	Vis M3 ou M4	Colle forte, ruban adhésif double face
Position	Au-dessus ou sur le côté du robot	Sur la pince (caméra embarquée)

Montage de la caméra

> 💡 **Problème fréquent :** Les trous du support imprimé 3D ne correspondent souvent pas aux pas de vis des caméras USB standard.

Solutions de montage testées :

1. **Agrandissement des trous :** Percer avec une mèche de 3-3.5mm
2. **Collage :** Colle époxy ou cyanoacrylate (prévoir temps de séchage 30min-2h)
3. **Ruban double face épais :** Solution temporaire mais efficace
4. **Serre-câbles :** Fixation rapide pour tests
5. **Pâte à fixe :** Repositionnable, idéal pour trouver le bon angle

Positions recommandées :

- **Vue du dessus (TOP) :** 40-60cm au-dessus de l'espace de travail, angle 45-60°
- **Vue latérale (SIDE) :** À hauteur du robot, 30-40cm de distance
- **Vue poignet (WRIST) :** Montée sur le servo 5, orientée vers la pince

🔧 Étape 2 : Résolution du problème OpenCV

> ⚠️ **Erreur fréquente lors du premier lancement :**
> `cv2.error: The function is not implemented. Rebuild the library with Windows, GTK+ 2.x or Cocoa support`

Cette erreur apparaît car LeRobot a installé `opencv-python-headless` (version sans interface graphique) au lieu de `opencv-python` (version avec fenêtres).

Solution validée

```
# Activer l'environnement
conda activate lerobot

# Vérifier ce qui est installé
pip list | grep opencv

# Si vous voyez "opencv-python-headless", le remplacer :
pip uninstall opencv-python-headless -y
pip install opencv-python

# Vérifier la correction
python -c "import cv2; print('OpenCV version:', cv2.__version__)"
```

> ✅ **Après cette correction :** Les fenêtres OpenCV pourront s'afficher normalement pour visualiser le flux vidéo de la caméra.

📷 Étape 3 : Détection et test de la caméra

Vérification de la connexion USB

```
# Vérifier les périphériques vidéo disponibles
ls /dev/video*

# Résultat attendu :
# /dev/video0  /dev/video1

# Note : Certaines caméras créent 2 devices :
# video0 pour la vidéo, video1 pour les métadonnées

# Donner les permissions nécessaires
sudo chmod 666 /dev/video*
```

Test de capture avec LeRobot

Utilisez la commande officielle LeRobot pour détecter et tester la caméra :

```
# Activer l'environnement
conda activate lerobot

# Se placer dans le dossier LeRobot
cd ~/lerobot

# Lancer la détection et capture d'images test
python lerobot/common/robot_devices/cameras/opencv.py \
    --images-dir outputs/images_from_opencv_cameras
```

```
Linux detected.
Finding available camera indices through scanning '/dev/video*' ports
Camera found at index /dev/video0
Connecting cameras
OpenCVCamera(0, fps=30, width=640, height=480, color_mode=rgb)
Saving images to outputs/images_from_opencv_cameras
Frame: 0000     Latency (ms): 822.57
Frame: 0001     Latency (ms): 55.93
Frame: 0002     Latency (ms): 59.66
...
Frame: 0090     Latency (ms): 36.21
Frame: 0091     Latency (ms): 32.00
Images have been saved to outputs/images_from_opencv_cameras
```

Vérification des images capturées

⚠️ **Étape importante :** Vérifiez quelle caméra correspond à quel index !

```
# Lister les images capturées
ls outputs/images_from_opencv_cameras/

# Ouvrir une image pour vérifier visuellement
xdg-open outputs/images_from_opencv_cameras/camera_00_frame_000000.png

# Ou sur système sans interface graphique, copier vers un PC :
# scp outputs/images_from_opencv_cameras/*.png user@pc:/path/
```

Vérifications à effectuer :

1. Ouvrez l'image `camera_00_frame_000000.png`
2. Confirmez que c'est bien la vue souhaitée (pince, dessus, côté)
3. Notez l'index correspondant : camera_00 = index 0 = /dev/video0
4. Si ce n'est pas la bonne caméra, c'est peut-être votre webcam intégrée

💡 **Conseil :** Si /dev/video0 est votre webcam intégrée, votre caméra USB sera probablement sur /dev/video2.

🎮 Étape 4 : Lancement du script de téléopération avec caméra

Le script `SEM_so101_7_teleoperation_camera.py` est basé exactement sur le script 6 de la Phase 5, avec uniquement l'ajout de la visualisation caméra.

Structure du script

Modifications apportées au script 6 :

- Ligne 13 : Ajout de `import cv2`
- Fonction teleoperation : Ajout initialisation caméra (6 lignes)
- Boucle principale : Ajout affichage vidéo (6 lignes)
- Fin du script : Ajout fermeture caméra (4 lignes)

Total : 17 lignes ajoutées, tout le reste identique à la Phase 5

Lancement du script

```
# Activer l'environnement
conda activate lerobot

# Se placer dans le dossier des scripts
cd ~/lerobot/Scripts_SEM/scripts

# Lancer le script 7
python SEM_so101_7_teleoperation_camera.py
```

Déroulement de l'exécution

1. Identification des robots (identique Phase 5)

```
╔═══════════════════════════════════════════════════════╗
║     TÉLÉOPÉRATION SO-ARM 101 + CAMÉRA                 ║
╚═══════════════════════════════════════════════════════╝

📷 Test de la caméra...
✅ Caméra détectée : 640x480

⚠️ Débranchez tous les robots
   Entrée quand fait...

🔌 Branchez le LEADER
   Entrée quand branché...
✅ LEADER détecté sur /dev/ttyACM0
🔄 Test de connexion LEADER...
     → Centre...
     → Fermé (45°)...
     → Ouvert (90°)...
     → Centre...
  ✅ LEADER connecté et testé

Pince du LEADER bougée? [O/N]: O

🔌 Branchez le FOLLOWER (gardez Leader branché)
   Entrée quand branché...
✅ FOLLOWER détecté sur /dev/ttyACM1
🔄 Test de connexion FOLLOWER...
  ✅ FOLLOWER connecté et testé

Pince du FOLLOWER bougée? [O/N]: O

✅ Identification réussie!
```

2. Configuration et positionnement (identique Phase 5)

```
[C]ôte à côte ou [F]ace à face?
Choix [C]: C

✅ Mode : CÔTÉ À CÔTÉ

🎯 Positionnement automatique...

🎯 Centrage simultané des robots...
✅ Robots centrés

🏁 Position repos simultanée...
✅ Position repos atteinte (robot replié)

⚠️ Téléopération dans 3 secondes...
📷 Caméra activée
```

3. Interface de téléopération avec caméra

Une fenêtre `Camera SO-ARM 101` s'ouvre en plus de l'interface terminal habituelle :

> **Fenêtre vidéo :** Affiche le flux en temps réel de la caméra
>
> - Résolution : 640x480 pixels
> - FPS : 30 images/seconde
> - Latence : < 100ms

```
╔═══════════════════════════════════════════════════╗
║     TÉLÉOPÉRATION - CÔTÉ À CÔTÉ                   ║
║     Servos miroir: []                             ║
╚═══════════════════════════════════════════════════╝

🎮 Commandes:
  [Q] + Enter : Quitter
  [F] + Enter : Flip mode (côté ↔ face)
  [q] dans la fenêtre vidéo : Quitter aussi
--------------------------------------

✅ Téléopération active!
🤖 Bougez le LEADER, le FOLLOWER suit
```

📊 Étape 5 : Tests de validation

Test 1 : Qualité d'image

1. Vérifiez que l'image dans la fenêtre est nette
2. Ajustez la mise au point si nécessaire (molette sur la caméra)
3. Assurez-vous que toute la zone de travail est visible
4. Évitez les contre-jours et reflets

Test 2 : Synchronisation téléopération + vidéo

1. Bougez le Leader pour déplacer le Follower
2. Vérifiez que le mouvement est visible dans la fenêtre vidéo
3. La vidéo ne doit pas ralentir la téléopération
4. Latence téléopération doit rester < 50ms

Test 3 : Changement de mode avec vidéo active

1. Appuyez sur [F] + Enter pour changer de mode
2. La vidéo doit continuer sans interruption
3. Les mouvements miroir/copie restent synchronisés

Test 4 : Stabilité sur 5 minutes

1. Laissez le système tourner avec vidéo active
2. Effectuez des mouvements réguliers
3. Vérifiez qu'il n'y a pas de gel d'image
4. La fenêtre vidéo ne doit pas se fermer seule

✅ Tests réussis si :

- La fenêtre vidéo s'affiche correctement
- Image nette et stable
- Pas de ralentissement de la téléopération
- Système stable pendant 5 minutes
- Fermeture propre avec 'q' dans la fenêtre

🔧 Dépannage

Problème	Cause possible	Solution
Erreur "function not implemented"	opencv-python-headless installé	Voir Étape 2 : remplacer par opencv-python
Fenêtre vidéo ne s'ouvre pas	Caméra non détectée	Vérifier `ls /dev/video*`
Image noire	Mauvais index caméra	Modifier ligne cap = cv2.VideoCapture(0) → (2)
Téléopération ralentie	Charge CPU trop élevée	Réduire résolution ou FPS caméra
"Permission denied /dev/video0"	Droits insuffisants	`sudo chmod 666 /dev/video*`
Caméra se déconnecte	Câble USB instable	Changer de port USB, éviter les hubs
Fenêtre se ferme seule	Exception non gérée	Relancer le script, vérifier les logs
Image floue	Mise au point incorrecte	Ajuster la molette de focus sur la caméra
Robots ne bougent pas	Script 6 modifié incorrectement	Utiliser le script fourni sans modification
Module cv2 not found	Mauvais environnement	`conda activate lerobot`

💡 Conseils pour l'enregistrement futur

Cette phase 6 prépare l'enregistrement de trajectoires (Phase 7). Voici les points importants à valider maintenant :

Position de la caméra

- **Vue complète :** Toute la zone de manipulation doit être visible
- **Angle optimal :** 45-60° pour voir la profondeur
- **Distance fixe :** Ne pas changer après validation
- **Stabilité :** Aucune vibration pendant les mouvements

Éclairage

- **Constant :** Éviter les variations (fenêtres, néons)
- **Diffus :** Pas d'ombres dures
- **Pas de reflets :** Sur les surfaces brillantes
- **Reproductible :** Même éclairage pour entraînement et utilisation

Arrière-plan

- **Fixe :** Pas d'éléments mobiles
- **Contrasté :** Distinguer clairement les objets
- **Simple :** Éviter les motifs complexes
- **Marqueurs :** Repères visuels pour l'IA

> 💡 **Astuce :** Prenez une photo de votre configuration finale. Vous devrez la reproduire exactement pour que l'IA fonctionne correctement après l'entraînement.

✅ Notes finales

✅ Phase 6 terminée quand :

- La caméra est montée solidement (même avec du bricolage)
- OpenCV avec support GUI est installé
- La caméra est détectée sur /dev/video0 (ou video2)
- Les images test sont capturées avec succès
- La fenêtre vidéo s'affiche pendant la téléopération
- Pas de ralentissement de la téléopération
- Le système reste stable pendant 5 minutes
- Vous savez fermer proprement avec 'q'

🚀 **Objectif atteint :** Votre système de téléopération dispose maintenant d'une visualisation en temps réel ! La caméra est prête pour l'enregistrement de trajectoires et l'apprentissage par imitation qui seront couverts dans la **Phase 7**.

📝 Récapitulatif des fichiers

Fichier	Description	Créé par
~/lerobot/Scripts_SEM/scripts/SEM_so101_7_teleoperation_camera.py	Script de téléopération avec visualisation caméra	Phase 6
~/lerobot/outputs/images_from_opencv_cameras/	Dossier contenant les images test de la caméra	Test caméra
Fichiers Phase 5 inchangés	Tous les fichiers de calibration et configuration	Phases 3-5

Service Écoles-Médias - DIP Genève

Guide Phase 6 - Version 1.0

19.12.2024