université PARIS-SACLAY
FACULTÉ DES SCIENCES D'ORSAY

Acquisition de données multimodales sur Thymio II

Documentation Technique

Webots / Python / HDF5

Table des matières

1. Contexte et objectifs

Ce document détaille l'implémentation du **contrôleur Python sous Webots** dédié à la création d'un jeu de données (dataset). Ce contrôleur est conçu pour l'apprentissage par imitation (*Behavioral Cloning*), où le robot apprend à reproduire les actions d'un expert.

Le contrôleur remplit trois fonctions critiques :

1. **Pilotage hybride** : Il gère les moteurs en mode autonome (via une logique d'évitement) tout en permettant une reprise en main manuelle via le clavier.
2. **Synchronisation sensorielle** : Il garantit que chaque échantillon du dataset associe précisément les lectures du LiDAR, de la caméra et des proximètres au même instant t.
3. **Persistance des données** : Il gère l'accumulation des données en mémoire vive et leur sérialisation structurée dans un fichier **HDF5** en fin de simulation.

2. Chaîne de traitement des capteurs

Avant l'enregistrement, les données brutes subissent un prétraitement pour réduire le bruit et la dimensionnalité.

2.1. Capteurs de proximité (infrarouges)

Le robot dispose de 7 capteurs horizontaux. Le script applique une normalisation pour stabiliser les entrées du futur réseau de neurones. Pour chaque capteur p_i (où $i \in [0, 6]$), le pipeline applique les transformations suivantes :

1. **Écrêtage (clamping)** : Limitation de la dynamique pour supprimer les valeurs aberrantes. Les valeurs sont bornées entre 0 et 4500 (valeur max du capteur).

$$p'_i = \min(p_i, 4500) \tag{1}$$

2. **Mise à l'échelle non-linéaire** : Compression de la variance pour linéariser la perception de la distance. Pour cela nous appliquons la racine carrée pour linéariser la réponse en distance.

$$p''_i = \sqrt{\frac{p'_i}{4500}} \tag{2}$$

3. **Normalisation vectorielle (L_1)** : Division par la somme totale du vecteur (plus un epsilon $1e^{\{-6\}}$ pour éviter la division par zéro). Transformation du vecteur P en une distribution de probabilité relative, assurant l'invariance à l'intensité globale.

$$\hat{p}_i = \frac{p''_i}{\sum_{j=0}^{6} p''_j + \varepsilon} \tag{3}$$

Où $\varepsilon = 10^{-6}$ prévient la division par zéro.

2.2. LiDAR (télémétrie laser)

Le capteur LiDAR du Thymio est configuré pour effectuer un balayage circulaire avec une **résolution de 90 points par tour**. Cette précision définit un pas angulaire constant, essentiel pour la reconstruction spatiale de l'environnement :

$$\Delta\theta = \frac{2\pi}{90} \approx 0.0698 \ \ \text{rad} \tag{4}$$

Pour chaque échantillon $i \in [0, 89]$, le capteur mesure une distance d_i. Ces données polaires brutes sont transformées en un nuage de points dans le repère cartésien (x, y) relatif au robot. Cette transformation permet au modèle d'apprentissage de traiter des coordonnées euclidiennes cohérentes :

$$x_i = d_i \cdot \sin(i \cdot \Delta\theta) \tag{5}$$

$$y_i = d_i \cdot \cos(i \cdot \Delta\theta) \tag{6}$$

Cette opération convertit un simple tableau de distances en une structure géométrique exploitable. Elle permet notamment d'extraire des primitives spatiales.

2.3. Vision (caméra projective)

Afin d'optimiser le stockage et de faciliter l'apprentissage, le contrôleur n'enregistre pas l'image brute ($W \times H \times 3$). Il applique une opération d'extraction de primitives pour réduire l'entrée à un vecteur de dimension 9. Cette étape garantit que le modèle traite des données pré-traitées et normalisées.

1. **Conversion d'espace colorimétrique** : Passage de l'espace BGR (brut) vers l'espace **HSV** (Hue, Saturation, Value) pour découpler la chrominance de la luminance.
2. **Extraction du bandeau central** : Pour se concentrer sur les informations pertinentes à la navigation, le script extrait un bandeau horizontal de hauteur $h = 20$ pixels, centré sur la hauteur totale H de l'image. Les limites du bandeau sont définies par :

$$y_{\text{start}} = \frac{H}{2} - 10, \quad y_{\text{end}} = \frac{H}{2} + 10 \tag{7}$$

3. **Agrégation Spatiale (Pooling)** : Ce bandeau est segmenté en trois secteurs verticaux $s \in \{L, C, R\}$ (Left, Center, Right). Pour chaque secteur contenant M pixels, on calcule la moyenne par canal :

$$\mu_{s,c} = \frac{1}{M} \sum_{k=1}^{M} x_{k,c} \tag{8}$$

4. **Mise à l'échelle** : Les moyennes sont normalisées selon les bornes de l'espace HSV :
   - **Teinte (H)** :

$$\hat{H}_s = \frac{\mu_{s,H}}{179} \tag{9}$$

   - **Saturation (S) et Valeur (V)** :

$$\widehat{S|V}_s = \frac{\mu_{s,S|V}}{255} \tag{10}$$

Le vecteur résultant injecté dans le fichier HDF5 est :

$$V_{\text{cam}} = \left[\hat{H}_L, \hat{S}_L, \hat{V}_L, \hat{H}_C, \hat{S}_C, \hat{V}_C, \hat{H}_R, \hat{S}_R, \hat{V}_R\right] \tag{11}$$

Cette méthode de reduction de dimensionnalité permet de conserver l'information sémantique nécessaire (présence d'une couleur spécifique devant le robot) tout en ignorant les données superflues du ciel ou du sol.&

3. Stratégie de commande et supervision

Le robot opère selon une boucle de contrôle hybride enregistrée à chaque pas de temps.

3.1. Mode autonome (réseau de neurones)

Par défaut, le robot utilise un réseau de neurones artificiels (ANN) simple (type Braitenberg) pour l'évitement d'obstacles.

- **Entrées** : Biais (1) et capteurs de proximité normalisés (x_1, x_2, x_3).
- **Sortie** : Vitesses gauche et droite calculées par produit scalaire avec des poids fixes.

3.2. Mode manuel (clavier)

Un opérateur humain peut surcharger les commandes du réseau de neurones en temps réel via le clavier.

- **Flèches HAUT/BAS** : Avancer / Reculer.
- **Flèches GAUCHE/DROITE** : Pivoter (modifie le différentiel de vitesse).
- **Touche "S"** : Arrêt d'urgence.

Intérêt pédagogique : Cela permet de corriger le robot lorsqu'il se trompe, créant ainsi des données d'apprentissage plus robustes (Correction active).

4. Stockage des données (HDF5)

L'enregistrement doit être déclenché par la fin de l'essai (touche "X"). Les listes python sont converties en tableaux `numpy` puis écrites dans le fichier `dataset_webots.hdf5`.

4.1. Structure du fichier

Dataset (Clé)	Dimensions	Description
`thymio_commands`	$(N, 2)$	Vitesse moteurs (Gauche, Droite). Cible de l'apprentissage.
`thymio_prox`	$(N, 7)$	Valeurs des 7 capteurs IR normalisés.
`thymio_scans`	$(N, 90, 2)$	Nuage de points LiDAR (90 points, coord X et Y).
`thymio_cam`	$(N, 9)$	Vecteur de primitives visuelles (HSV moyenné par zones).

Note : N correspond au nombre de pas de temps enregistrés durant l'expérience.

4.2. Implémentation du contrôleur (écriture HDF5)

Le script de contrôle doit accumuler les données à chaque pas de temps (**t**). Une fois la session de collecte terminée, les données sont converties en tableaux **NumPy** pour être enregistrées de manière optimisée.

```python
import h5py
import numpy as np

# 1. Initialisation du stockage temporaire
storage = {
    'prox': [], 'scans': [], 'cam': [], 'cmds': []
```

```
}

# --- Boucle principale de simulation Webots ---
while robot.step(timestep) != -1:
    # ... calcul des features (MLOps) ...

    # Accumulation des données dans les listes
    storage['prox'].append(norm_prox_vector)
    storage['scans'].append(cartesian_points)
    storage['cam'].append(hsv_features)
    storage['cmds'].append([left_speed, right_speed])

    # Condition d'arrêt de la collecte (ex: durée ou touche clavier)
    if stop_condition:
        break

# --- Sauvegarde finale (Écriture HDF5) ---
with h5py.File('dataset_webots.hdf5', 'w') as f:
    # Conversion des listes accumulées en tableaux NumPy
    array1 = np.array(storage['cmds'])
    array2 = np.array(storage['scans'])
    array3 = np.array(storage['prox'])
    array4 = np.array(storage['cam'])

    # Création des datasets finaux
    f.create_dataset('thymio_commands', data=array1)
    f.create_dataset('thymio_scans',    data=array2)
    f.create_dataset('thymio_prox',     data=array3)
    f.create_dataset('thymio_cam',      data=array4)

print('Fichier dataset_webots.hdf5 sauvegardé avec succès.')
```

5. Utilisation des données

Voici comment charger et exploiter le fichier généré pour un entraînement futur en Python.

5.1. Lecture du dataset

```
import h5py
import numpy as np

# 1. Ouverture du fichier
with h5py.File('dataset_webots.hdf5', 'r') as f:
    # 2. Chargement des datasets
    prox = f['thymio_prox'][:]
    cmds = f['thymio_commands'][:]

    print(f'Nombre échantillons : {len(prox)}')
    print(f'Exemple de commande : {cmds[0]}')
```

5.2. Bonnes pratiques

- Validation : Vérifiez toujours que les dimensions (shapes) des tableaux sont cohérentes entre elles (même nombre **N** de lignes).
- Normalisation inverse : Si vous devez visualiser les données images, rappelez-vous que les canaux HSV ont été divisés par 179 (H) et 255 (S, V).