# AI vidéo bird-tracker

## Contexte du Projet

Ce projet a été conçu pour détecter, suivre et analyser le mouvement des oiseaux à partir d'un flux vidéo en temps réel. Il utilise des modèles d'apprentissage profond et des algorithmes de suivi avancés afin de localiser précisément les oiseaux et de suivre leur trajectoire jusqu'à ce qu'ils quittent le champ de vision.

Le système est conçu pour être utilisé dans des applications telles que la surveillance de la faune, les études comportementales ou encore la prévention des collisions avec des infrastructures.

---

## Architecture du Système

### 1. Environnement de Travail

Le projet s'appuie principalement sur les outils et bibliothèques suivants :

- **Python** : Langage principal du projet.
- **YOLOv5** : Modèle de détection d'objets pré-entraîné et personnalisé pour la reconnaissance des oiseaux.
- **OpenCV** : Utilisé pour le traitement d'images et l'affichage des résultats.

- **Sort** : Algorithme de suivi multi-objets pour maintenir l'identité des oiseaux détectés dans le flux vidéo.

## 2. Structure des Fichiers

- `track.py` : Contient le cœur du système de détection et de suivi en temps réel.
  - Ce script utilise **YOLOv5** pour détecter les objets dans chaque image du flux vidéo. Les détections incluent des coordonnées de boîtes englobantes, des scores de confiance et des étiquettes de classes.
  - Les détections sont passées à l'algorithme **Sort** pour associer les objets détectés entre les images successives et leur attribuer des identifiants uniques.
  - Les résultats sont affichés sur le flux vidéo avec des boîtes, des labels et des scores.
- `result.py` : Analyse et visualisation des résultats d'entraînement du modèle YOLOv5.
  - Charge un fichier CSV contenant les métriques et les pertes d'entraînement (époque par époque).
  - Produit deux graphiques principaux :
    1. L'évolution des erreurs (box, obj, cls) pour l'entraînement et la validation.
    2. Les métriques de performance (échelle de précision, rappel, mAP).
- `verif.py` : Script de vérification pour s'assurer que les trackers nécessaires sont disponibles dans OpenCV.
  - Teste plusieurs trackers (BOOSTING, KCF, CSRT) pour confirmer leur compatibilité.
- `conf.yaml` : Fichier de configuration contenant les informations sur les données d'entraînement, les classes d'objets et les chemins pertinents.
  - Spécifie les classes d'objets : `person`, `bird`, et une classe supplémentaire anonymisée.

- Configure les chemins pour les ensembles de données d'entraînement et de validation.

---

# Fonctionnalités

## Détection des Oiseaux

Le modèle YOLOv5 est entraîné pour reconnaître des oiseaux. Les coordonnées des boîtes englobantes sont ajustées pour correspondre à la résolution d'origine de l'image. Le score de confiance permet de filtrer les détections les moins fiables.

Exemple de code :

```
# Charger le modèle YOLOv5
model = torch.hub.load('ultralytics/yolov5', 'custom', path=m
odel_path)

# Faire une prédiction
results = model(image)

# Extraire les résultats
for *box, conf, cls in results.pred[0]:
    x1, y1, x2, y2 = map(int, box)
    label = model.names[int(cls)]
    print(f"Détection: {label}, Confiance: {conf:.2f}, Boîte:
{x1}, {y1}, {x2}, {y2}")
```

## Suivi des Oiseaux

Le module de suivi repose sur l'algorithme **Sort**, qui permet de :

- Associer les détections entre les images consécutives en utilisant une mesure d'IOU (Intersection over Union).
- Gérer les cas où les oiseaux quittent temporairement ou reviennent dans le champ de vision.

- Identifier les objets persistants grâce à des identifiants uniques.

Exemple de code :

```
# Initialiser le tracker
tracker = Sort()

# Mettre à jour le tracker avec les détections
trackers = tracker.update(detections)

# Afficher les résultats
for trk in trackers:
    x1, y1, x2, y2, obj_id, score, label = trk
    print(f"ID: {obj_id}, Label: {label}, Confiance: {score:.
2f}, Boîte: {x1}, {y1}, {x2}, {y2}")
```

## Analyse des Performances

Le script `result.py` produit des visualisations des erreurs (box, obj, cls) et des métriques (précision, rappel, mAP) issues de l'entraînement du modèle YOLOv5.

Exemple de code :

```
# Charger les résultats d'entraînement
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('results.csv')

# Tracer les métriques
plt.plot(df['epoch'], df['metrics/precision'], label='Précisi
on')
plt.plot(df['epoch'], df['metrics/recall'], label='Rappel')
plt.legend()
plt.xlabel('Époch')
plt.ylabel('Métriques')
```

```
plt.title('Évolution des métriques d'entraînement')
plt.show()
```

---

# Instructions d'Utilisation

## Prérequis

1. Installer les dépendances Python :

   ```
   pip install torch torchvision opencv-python pandas matplotlib
   ```

2. Vérifier que le modèle YOLOv5 entraîné est présent au chemin spécifié dans le script `track.py` :

   ```
   model_path = 'D:\Tracking\yolov5\runs\train\exp2\weights\best.pt'
   ```

## Exécution

1. Lancez le script `verif.py` pour confirmer la présence des trackers dans OpenCV :

   ```
   python verif.py
   ```

2. Démarrez le suivi en temps réel :

   ```
   python track.py
   ```

   Ce script ouvrira un flux vidéo et affichera les détections et le suivi des oiseaux en temps réel.

3. Visualisez les performances d'entraînement :

   ```
   python result.py
   ```

# Résultats Attendues

## En Temps Réel

- Les oiseaux détectés apparaîtront entourés de boîtes avec leurs scores de confiance.
- Chaque oiseau suivi aura un identifiant unique tant qu'il restera dans le champ de vision.

## Analyse Post-Entraînement

- Graphiques d'évolution des erreurs et des métriques pour évaluer la qualité de l'entraînement du modèle YOLOv5.

*Image d'illustration prise au cours de test*