TER : Vision par Ordinateur Temps Réel pour Voitures Miniatures, Détection et Suivi de Trajectoire

Manda ANDRIAMAROMANANANA
Encadré par Steven Martin

Université Paris-Saclay

07 mai 2025

Plan de la Présentation

1. Introduction & Objectifs du projet
2. Architecture du système
3. Fonctionnalités implémentées
4. Résultats & Performances
5. Difficultés rencontrées & solutions
6. Conclusion & perspectives

Introduction et Objectifs

Contexte et Objectifs

Projet Carrera GO : voiturette autonome vue du dessus

Objectif : suivre la position de la voiture et vérifier si elle reste sur la piste

Contraintes : traitement en temps réel, déploiement embarqué, robustesse du suivi

Architecture du Système

Entrées et sorties du système

Entrée : flux vidéo (caméra ou vidéo test)

Sorties :

- Vidéo annotée (OpenCV)
- Statut `on_track` (par objet)
- Données envoyées via HTTP (coordonnées, identifiants)

Pipeline Global

1. Acquisition vidéo (OpenCV)
2. Détection d'objets (YOLOv5 ONNX)
3. Suivi avec identifiants persistants (CentroidTracker)
4. Segmentation de la piste sombre (HSV)
5. Calcul du statut `on_track`
6. Export des infos via HTTP (optionnel)

Configuration via YAML I

Tous les paramètres sont définis dans `config.yaml`

Structure modulaire : modèle, détection, suivi, visualisation...

Extraits représentatifs :

```yaml
# === Modèle et classes ===
model_path: "model/best.onnx"
data_yaml_path: "carrera_go/data.yaml"
target_class: "mario"
resize_size: 640

# === Détection ===
conf_threshold: 0.4
iou_threshold: 0.5
```

Configuration via YAML II

```yaml

# === Piste sombre (HSV) ===
upper_black: [179, 100, 100]
dark_mask_history_length: 10
min_area: 35000
dark_area_threshold: -1
contour_update_interval: 1
mask_alpha: 0.4

# === Suivi / Analyse mouvement ===
tracker_max_distance: 50
speed_threshold: 1.0
weight_dark: 0.101
weight_speed: 0.7
on_track_score_threshold: 0.599
```

Configuration via YAML III

```
25
26 # === Affichage / communication ===
27 max_fps_window: 5
28 send_interval: 0
```

Fonctionnalités Implémentées

Extrait : Vérification de la présence sur la piste

```
1 def is_on_track(object_bbox, final_mask, threshold=-5)
    :
2     x1, y1, x2, y2 = object_bbox
3     object_center = (int((x1 + x2) / 2), int((y1 + y2)
          / 2))
4     contours, _ = cv2.findContours(final_mask, cv2.
          RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
5     return any(cv2.pointPolygonTest(contour,
          object_center, False) > threshold for contour
          in contours)
```

Tracking : CentroidTracker

```python
class CentroidTracker:
    def __init__(self, max_distance=50):
        self.next_object_id = 0
        self.objects = {}
        self.max_distance = max_distance

    def update(self, detections):
        # Attribution ou cr ation d'identifiants
            persistants
        ...
```

Segmentation de la piste sombre

```
1 def get_dark_mask(frame, min_area=35000, upper_black=(
    360,75,95)):
2     hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
3     mask = cv2.inRange(hsv, (0,0,0), upper_black)
4     mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, np.
        ones((3,3),np.uint8))
5     contours, _ = cv2.findContours(mask, cv2.
        RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
6     valid = [c for c in contours if cv2.contourArea(c)
         > min_area]
7     ...
```

Envoi HTTP des positions

```
1 def send_position_http(url, object_center, on_track,
      frame_id=None):
2     payload = {"x": object_center[0], "y":
          object_center[1],
3                "on_track": on_track, "frame": frame_id
                    }
4     try:
5         response = requests.post(url, json=payload,
              timeout=1.0)
6         ...
```

Résultats et Performances

.pt vs .onnx : Comparaison

.pt (PyTorch) :

Format natif pour le développement et l'entraînement
Performant sur GPU, mais lourd pour CPU/embarqué

.onnx :

Format d'inférence optimisé, compatible ONNX Runtime / NCNN
Plus rapide et léger sur CPU (1.5–3× plus rapide)
Recommandé pour déploiement embarqué

Performances

Machine avec GPU (format *.pt) :

- ˜30 FPS avec affichage (OpenCV)
- ˜45+ FPS sans affichage

Affichage de la position, du statut *on-track*, des identifiants

Interface visuelle fiable et réactive

Non fluide en .pt sur CPU, d'où le choix du format .onnx

Sortie Terminal : Profilage Temps Réel

```
[INFO] Average FPS over session: 18.76

[PROFILE] Total time spent per block (seconds):
  frame_acquisition              : 3.662 sec total (0.0043 sec/frame)
  dark_area_processing           : 8.497 sec total (0.0100 sec/frame)
  yolo_inference                 : 30.113 sec total (0.0356 sec/frame)
  tracking_update                : 0.457 sec total (0.0005 sec/frame)
  fps_calc                       : 0.004 sec total (0.0000 sec/frame)
```

FPS moyen : environ 20 (avec affichage, sinon 25-30 fps)

YOLOv5 reste le bloc le plus coûteux : ~36ms/frame

Profilage utile pour cibler les optimisations

Démonstration

Vidéo montrant :

- Suivi par boîte englobante et ID persistants
- Masque vert représentant la piste sombre détectée
- Indicateur de statut : vert = sur la piste, rouge = hors-piste

Difficultés Rencontrées

Difficultés techniques

Raspberry Pi : performances NCNN insuffisantes

Modules externes (Pedro/Javier) : absence de code → généralisation des requêtes HTTP

Stabilité de la détection : ajustement des seuils, robustesse au bruit

Stratégies Envisagées pour le Suivi sur la Piste I

QR Codes / Fiduciaux :

Balises visuelles type ArUco pour localiser la voiture dans un repère global

Limites : occlusion fréquente, perte de fiabilité en cas de flou ou vitesse élevée, calibration contraignante

Détection de la géométrie de la piste :

Reconstruction ou modélisation de la forme du circuit (contours, spline, graphes)

Limites : dépend d'un modèle explicite, difficilement généralisable, sensible à la perspective

Mémoire des positions valides (KD-Tree) :

Phase d'apprentissage : enregistrement des positions détectées comme « sur la piste »

Calcul d'une zone valide à partir d'un KD-Tree de ces points

Stratégies Envisagées pour le Suivi sur la Piste II

Limites : spécifique à un circuit, nécessite un échantillonnage préalable fiable

Approche par zones sombres (HSV) :

Masquage via HSV pour détecter la piste (basse saturation, faible valeur)

Moyenne temporelle pour lisser les variations, suivi via `is_on_track`

Limites : sensible à la lumière, bruit visuel, pas de contexte spatial global

Fusion heuristique : score pondéré (vitesse + distance) :

Score de distance à la zone sombre + **score de vitesse instantanée**

Pondération ajustable : score global $= \alpha \cdot$ vitesse $+ \beta \cdot$ proximité

Avantages :

Stratégies Envisagées pour le Suivi sur la Piste III

Meilleure robustesse aux erreurs ponctuelles (ex. ombres)
Permet d'anticiper les sorties de piste (ralentissement, dérive)

Conclusion et Perspectives

Bilan du projet

Suivi et détection fonctionnels

Pipeline modulaire et extensible

Déploiement sur GPU réussi, partiellement sur RPi

Perspectives

Optimisation pour exécution sur Raspberry Pi

Intégration avec le contrôle moteur

Apprentissage par renforcement pour conduite autonome

Références I

Onnx runtime.

`https://onnxruntime.ai/`.
Accessed : 2025-05-06.

Opencv.

`https://opencv.org/`.
Accessed : 2025-05-06.

Roboflow.

`https://roboflow.com/`.
Accessed : 2025-05-06.

Tencent ncnn.

`https://github.com/Tencent/ncnn`.
Accessed : 2025-05-06.

Aho, A. V. and Ullman, J. D. (1972).

The Theory of Parsing, Translation and Compiling, Vol. 1.
Prentice-Hall.

Bojarski, M. et al. (2016).

End to end learning for self-driving cars.
arXiv preprint arXiv :1604.07316.

Gusfield, D. (1997).

Algorithms on Strings, Trees, and Sequences.
Cambridge University Press.

Références II

Howard, A. et al. (2017).

Mobilenets : Efficient convolutional neural networks for mobile vision applications.
arXiv preprint arXiv :1704.04861.

Jocher, G. et al. (2020).

YOLOv5 by ultralytics.
`https://github.com/ultralytics/yolov5`.

Redmon, J. and Farhadi, A. (2018).

Yolov3 : An incremental improvement.
arXiv preprint arXiv :1804.02767.

Wojke, N., Bewley, A., and Paulus, D. (2017).

Simple online and realtime tracking with a deep association metric.
In *IEEE International Conference on Image Processing (ICIP)*.