Travail d'initative personnelle encadrée

Louis Dewaele

15 avril 2024

Plan de la présentation

Réflexion sur le thème global

Jeux et sport

Sujet et enjeux

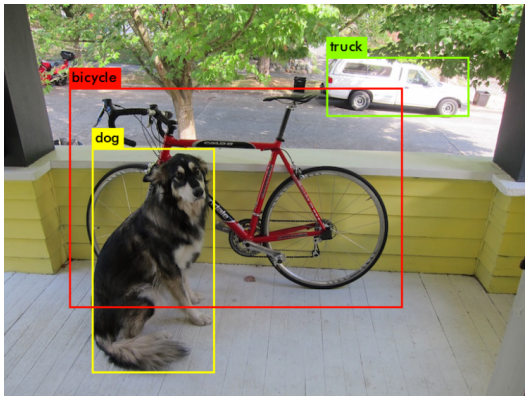
Sujet : Un golfeur peut-il s'entraîner avec l'aide d'un suivi vidéo ?

Enjeux : Appréhender l'arrivée de nouvelles technologies dans le domaine sportif afin d'étudier l'impact que ces dernières pourraient avoir sur la manière dont le sport est pratiqué à haut niveau.

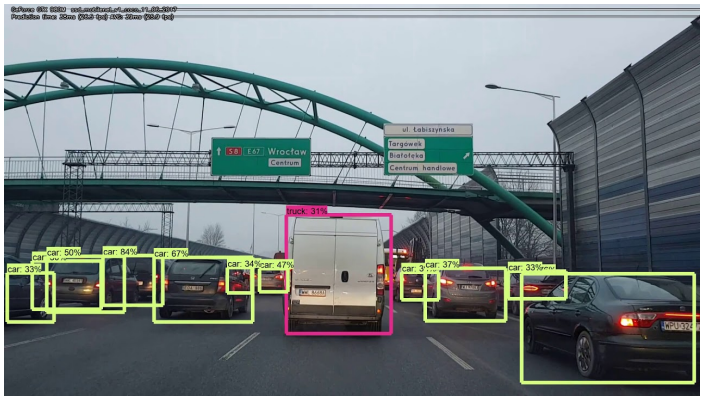
Réflexion sur le sujet

- Traquer un club de golf à l'aide d'un mouvement
- Placer un capteur sur une balle de golf
- Modèle choisi
- Matériel choisi
- Choix d'une approche informatique
- Traquer deux points colorés sur un club de golf

Recherches effectuées



OpenCV



OpenCV (Open Source Computer Vision Library) est une bibliothèque de vision par ordinateur et de traitement d'images. Elle offre un large éventail de fonctionnalités pour la manipulation, l'analyse et le traitement d'images et de vidéos.
Il est possible d'effectuer des opérations telles que la détection d'objets, la reconnaissance faciale, le suivi de mouvement, la stéréovision, la reconstruction 3D, etc.

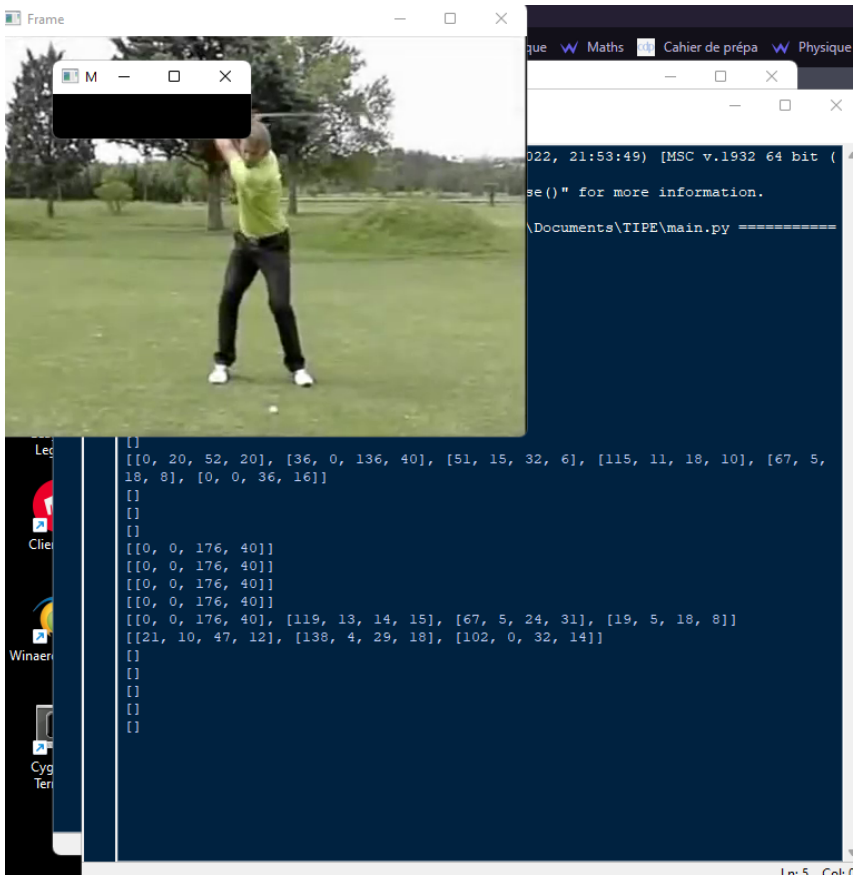
OpenCV - Application et pseudo code

Algorithme 1 : Déterminer les positions du club de golf à intervalle de temps régulier

Entrée : Vidéo d'un "swing".
Sortie : Tableau contenant les positions du club à T intervalle de temps.

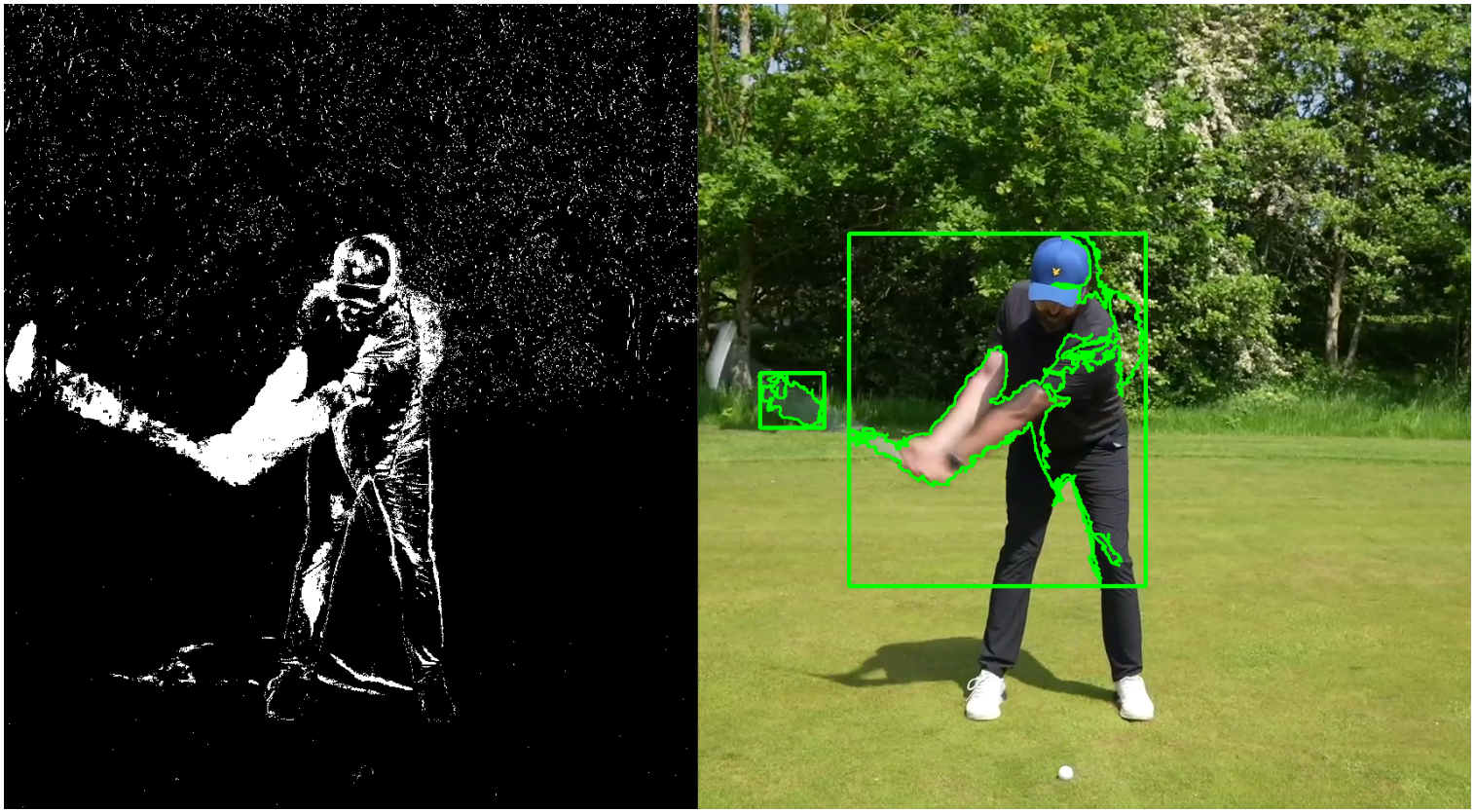
```
initialisation;
while la vidéo n'est pas terminée do
    on applique un masque sur l'image en cours d'analyse;
    for détection d'un élément to le contour est dessiné do
        Calculer le contour de l'élément détecté;
    end
    if le contour calculé correspond à un objet then
        afficher l'objet et son contour;
    end
end
```

Première approche



Première approche

En utilisant une nouvelle vidéo on obtient ceci :



Deuxième approche

1re expérience :

- Matériel : club de golf

- Objectif : Déterminer "à la main" le centre de gravité du club de golf

2e expérience :

- Matériel : Caméra à haut taux de rafraîchissement, club de golf.

- Placer 2 pastilles de couleur au niveau des mains du golfeur et du centre de gravité. Se placer dans une pièce éclairée, face à un mur blanc et effectuer un swing.

Traîtement des données

- Utiliser les positions des deux points traqués pour déterminer : la vitesse ; l'accélération et l'énergie méchanique du club de golf.
- Déterminer l'énergie communiquée à la balle de golf par le club.
- Déterminer la trajectoire que suivra la balle.

À venir

- Améliorer le modèle en estimant des forces de frottements sur le club, la balle (réalisation d'une 3e expérience).
- Reproduire l'expérience sur un terrain : confrontation modèle théorique et réalité.

Annexe

Références : "Golf Swing au ralenti sur Golf Academy TV" - ChaineGolfAcademyTV consulté le 05/06/23 / "How to Swing a golf club (The EASIEST way)" - Rick Shiels Golf consulté le 05/06/23 / OpenCV - Wikipédia consulté le 14/05/23 / "Object Tracking with OpenCV" - Livecodestream.dev consulté le 14/05/23

Annexes

```
import cv2
from tracker import *

tracker = EuclideanDistTracker()

cap = cv2.VideoCapture("Swing.mp4")

# Détection d'objet depuis la caméra stable
object_detector = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=200)
while True:
    ret, frame = cap.read()
    height, width, _ = frame.shape

    # Extraire la région d'intéret
    roi = frame[0:1000, 1000:1920]

    # 1. Détection d'objet
    mask = object_detector.apply(roi)
    _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    detections = []

    # 2. Traquer l'objet
    boxes_ids = tracker.update(detections)
    for box_id in boxes_ids:
        x, y, w, h, id = box_id
        cv2.putText(roi, str(id), (x, y - 15), cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
        cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 3)
```

Annexes

```
    for cnt in contours:
        # Retirer les éléments inintéressants
        area = cv2.contourArea(cnt)
        if area > 300:
            cv2.drawContours(roi, [cnt], -1, (0, 255, 0), 2)
            x, y, z, h = cv2.boundingRect(cnt)
            cv2.rectangle(roi, (x, y), (x + z, y + h), (0, 255, 0), 3)
            detections.append([x, y, z, h])

    print(detections)
    cv2.imshow("roi", roi)
    cv2.imshow("mask",mask)

    key = cv2.waitKey(30)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Annexes

```python
import math

class EuclideanDistTracker:
    def __init__(self):
        self.center_points = {}
        self.id_count = 0

    def update(self, objects_rect):
        objects_bbs_ids = []
        for rect in objects_rect:
            x, y, w, h = rect
            center_x = (x + x + w) // 2
            center_y = (y + y + h) // 2
            same_object_detected = False
            for id, pt in self.center_points.items():
                prev_center_x, prev_center_y = pt
                distance = math.sqrt((center_x - prev_center_x) ** 2 + (center_y - prev_center_y) ** 2)
                if distance < 25:
                    self.center_points[id] = (center_x, center_y)
                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break
            if not same_object_detected:
                self.center_points[self.id_count] = (center_x, center_y)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        new_center_points = {}
        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```