Correction

Exercice 1

- Cas 1 : Intersection = 0, Union = 2
- Cas 2 : Intersection = 1, Union = 3
- Cas 3 : Intersection = 1, Union = 3

Exercice 2 et 3

```python
def compute_iou(x1, y1, w1, h1, x2, y2, w2, h2):
    a = x1
    b = x1 + w1
    c = x2
    d = x2 + w2
    a_int = max(0, min(b, d) - max(a, c))

    e = y1
    f = y1 + h1
    g = y2
    h = y2 + h2
    a_hor = max(0, min(f, h) - max(e, g))

    intersection = a_hor * a_int
    area1 = w1 * h1
    area2 = w2 * h2
    union = area1 + area2 - intersection

    return intersection / union

# Numerical test cases
assert abs(compute_iou(10, 10, 50, 50, 30, 10, 50, 50) - 0.42857142857142855) < 1e-6
assert abs(compute_iou(0, 0, 100, 100, 50, 50, 100, 100) - 0.14285714285714285) < 1e-6
assert abs(compute_iou(0, 0, 50, 50, 60, 60, 50, 50) - 0.0) < 1e-6
```

Exercice 4 et 5

```python
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np

def pourcentage_above_threshold(ious, threshold):
    above = 0
    n = 0
    for iou in ious:
        if iou >= threshold:
            above += 1
```

```
        n += 1
    return above/n

def compute_iou(x1, y1, w1, h1, x2, y2, w2, h2):
    a = x1
    b = x1 + w1
    c = x2
    d = x2 + w2
    a_int = max(0, min(b, d) - max(a, c))

    e = y1
    f = y1 + h1
    g = y2
    h = y2 + h2
    a_hor = max(0, min(f, h) - max(e, g))

    intersection = a_hor * a_int
    area1 = w1 * h1
    area2 = w2 * h2
    union = area1 + area2 - intersection

    return intersection / union

SEQ_ROOT = 'A DEFINIR'
for seq in os.listdir(SEQ_ROOT):
    tracker = cv2.TrackerMIL_create()


    SEQ_ROOT_PATH = os.path.join(SEQ_ROOT, seq)

    GT_PATH = os.path.join(SEQ_ROOT_PATH, 'groundtruth_rect.txt')
    IMG_PATH = os.path.join(SEQ_ROOT_PATH, 'img')

    boxes = open(GT_PATH).readlines()

    IOUS = []
    for idx, img_name in enumerate(sorted(os.listdir(IMG_PATH))):
        img_cv2 = cv2.imread(os.path.join(IMG_PATH, img_name))
        if ',' in boxes[idx]:
            gt_box = list(map(lambda x:int(x),boxes[idx].strip().split(',')))
        else:
            gt_box = list(map(lambda x:int(x),boxes[idx].strip().split('\t')))
        if idx == 0:
            tracker.init(img_cv2, gt_box)
        else:
            success, box = tracker.update(img_cv2)
            cv2.rectangle(img_cv2, gt_box, (0, 255, 0), 2)
            cv2.rectangle(img_cv2, box, (0, 0, 255), 2)
            cv2.imshow('Tracking', img_cv2)
            cv2.waitKey(1)
            iou = compute_iou(box[0], box[1], box[2], box[3], gt_box[0], gt_box[1],
    gt_box[2], gt_box[3])
            IOUS.append(iou)

    S = []
    for t in np.linspace(0, 1, 100)[1:]:
        S.append(pourcentage_above_threshold(IOUS, t))

    AUC = np.trapezoid(S, dx=0.01)
```

```
70
71      plt.title('L\'aire sous la courbe (AUC) est de {:.2f}%'.format(AUC*100))
72      plt.xlabel('Seuil d\'IOU pour considérer une prédiction comme correcte')
73      plt.ylabel('Pourcentage de prédictions correctes pour ce seuil')
74      plt.plot(np.linspace(0, 1, 100)[1:], S)
75      plt.show()
```