#1 TRACKING CODE

In that portion of the code, the logic is, we suppose that a fish can’t move from one frame to another to a distance over 40px, so in the case where the closest bounding box (bb) of a kalman box is located over this distance, it means that it’s not it original bb (so it has lost track on its bounding box), it’s why we decide not to move it attributing it its last position.

# In case where it's located 40 px away from its previous position the kalman box remain in that previous one

if distances[min\_index] > 40:

                posicion = anteriores[kalman\_id]

                dic\_key[kalman\_id]+=1

        else:

            dic\_key[kalman\_id] = 0

#2

In case it has lost track for more than half of a seconde (fps/2), we check if the number of bb detected in the last frame its smaller than the current one by 1. If so, it means that its bounding box had disappeared and now is back. So we associate that kalman box to the new bb detected.

#Dans le cas ou il se soit eloigné pendant une demi seconde (fps/2)

        if kalman\_id in dic\_key and dic\_key[kalman\_id]>=fps/2:

            #Si sa lubinas a reapparu cet a dire on est passé de 4 lubinas à 5 lubinas

            if len(boxes) == len(prev\_box)+1:

                for key, value in boxes.items():

                    #On cible l'id qui n'etait pas là au frame precedent et on actualise les coord de sa position avec ceux de la bb correspondante

                    if key not in prev\_box:

                        posicion = np.array([math.ceil(value[0] + (value[2]- value[0] )/2),math.ceil(value[1] + (value[3]-value[1])/2)])

                        bool = True

            prev\_box =boxes #On actualise le prev box seulement quand on est sur que ca da disparu pour que ca soit plus precis

#3

Here we handle the case where a Kalman box has lost its bb for too long (more than 2 sec), we assume it means that its bb has not be misdetected but instead has moved suddendly and found itself over those 40px. In this case store all the bb that are associated to a kalman box and we keep in memory the kalman box which not.

if dic\_key[kalman\_id]>=fps\*2 and bool== False:

            b = 1

            petit\_joueur2 = kalman\_id

        else:

            #je stocke tous les id de bb qui sont associés à des kal

            for key, value in boxes.items():

                if math.dist( (int(value[0]), int(value[1])), pt1) <=5:

                    bob2.append(key)

In the case we’ve found one, we look for a bb that’s not associated with any kalman box, and when we find it, we associate it to its kalman box

if b==1 and kalman\_id==len(kalman\_list)-1:

            for key, value in boxes.items():

                if key not in bob2:

                    pos2 = np.array([math.ceil(value[0] +

(value[2]- value[0])/2),math.ceil(value[1] + (value[3]-value[1])/2)])

                    medidas[petit\_joueur2] = pos2

                    kalman\_list[petit\_joueur2].update(medidas[petit\_joueur2])

                    b=0

#4

We now want to handle the case where a two Kalman boxes associate themselves to the same bounding box, as a result one of them lost track on its own bb. To handle this, we first store in a dictionary the association each kalman box and its respective bb. In the case where we find a kalman box associated to bb already attributed to another. We keep the id of this kalman box, and iterate its occurrence.

 for key, value in boxes.items():

            if math.dist( (int(value[0]), int(value[1])), pt1) <=5:

                if key not in associated\_bb.values():

                    associated\_bb[kalman\_id] = key

                else:

                    a=1

                    lost\_id = kalman\_id

                    pos\_petit = pt1

                    index[lost\_id]+=1

                    associated\_bb[kalman\_id] = key

        print(f" boxes = {len(boxes)} -> kalman = {len(kalman\_list)}")

In case it happens and so for more than half a second, we iterate over the dictionary to get the id of the two kalman boxes that are associated to the same bounding box, we calculate the distance between the position of each of these kalman box before the stick up, and the bounding box on witch one of the two had lost tracked in the current frame. So, that untracked bb will be attributed to the kalman box with the shorter distance.

#Si verifie s'il s'agit bien du cas ou deux boxes de kal suivent la meme lubina et je m'assure aussi que nous somme de l'indice de la derniere box de kal

        #Comme ca on est sur que les bb not prises en compte sont celles qui ne correspondent pas au critere

        if a==1 and kalman\_id==len(kalman\_list)-1 and lost\_occurence2[lost\_id]>=fps/2:

            # Lorsque deux lubinas se croisent les boxes de kal ont tendance à suivre une seule lubina et l'autre se retrouve sans rien

            #Dans ce cas je recupere les id de ces deux lubinas sous les noms de first and second key

            first\_occurrence = {}

            first\_key = None

            second\_key = None

            for kb, bb in associated\_bb.items():

                if bb in first\_occurrence:

                    first\_key = first\_occurrence[bb]  # Get the first key where this value was seen

                    second\_key = kb

                    break

                else:

                    first\_occurrence[bb] = kb  # Store the first key where the value appears

            for key, value in boxes.items():

                #Je parcous les bb et je note celle qui n'est plus suivie

                if key not in associated\_bb2 :

                    #A ce niveau on a deux boxes de kal qui suivent la meme lubinas, pour attribuer la bonne à chacune

                    #On verifie laquelles de la positions de ces boxes de kal avant leur choque est la plus proche de la position de bb non non tracké

                    #Celle dont la position avant la collision est la plus proche de la position actuel de la bb non tracké sera attribué à celle ci

                    if math.dist( (int(value[0]), int(value[1])), posicion\_key[first\_key]) < math.dist( (int(value[0]), int(value[1])), posicion\_key[second\_key]):

                        pos = np.array([math.ceil(value[0] + (value[2]- value[0] )/2),math.ceil(value[1] + (value[3]-value[1])/2)])

                        medidas[first\_key] = pos

                        kalman\_list[first\_key].update(medidas[first\_key])

                        a=0

                        lost\_occurence2[lost\_id]=0

                    else:

                        pos = np.array([math.ceil(value[0] + (value[2]- value[0] )/2),math.ceil(value[1] + (value[3]-value[1])/2)])

                        medidas[second\_key] = pos

                        kalman\_list[second\_key].update(medidas[second\_key])

                        a=0

                        lost\_occurence2[lost\_id]=0

#5

In case where a misdetection leads to greater number of Kalman boxes than actual seabasses in the tank, we look for kalman boxes that lost their bounding boxes for quite long (2 sec) and we don’t find a bb untracked (key not in associated\_bb2), it means that all kalman boxes are already associated and this one doesn’t have any bb to be associated with yet, so we proceed the removal of this kalman box.

        # We handle the case where at the begining the program detects more kalman box than tha actual number of seabass in the tank

        if  kalman\_id==len(kalman\_list)-1 and lost\_occurence2[lost\_id]>=fps\*2 and len(kalman\_list)<4:

            k=0

            for key, value in boxes.items():

                if key not in associated\_bb2 :

                    k=1

            if k==0:

                lost\_occurence2[lost\_id] =0

                prev\_length -=1

                # Remove Kalman filter at index 7

                del kalman\_list[lost\_id]

                # Adjust other lists to maintain alignment with kalman\_list

                del medidas[lost\_id]

                del col\_caja[lost\_id]

                #del DictX[col\_caja[lost\_id]]

                #del DictY[col\_caja[lost\_id]]

                continue  # Skip the rest of the loop for kalman\_id == 7