Confronto algoritmi di tracciamento di OpenCV

Avendo i file con le coordinate di ogni tracker per ogni video si segue con il confronto

Importazione dati e creazione dataframe con coordinate

```
##         TYPE NFRAME FPS A_X A_Y B_X B_Y video_name
## 1   BOOSTING      1  44 539 131 579 228   pingpong
## 2   BOOSTING      2  42 539 131 579 228   pingpong
## 3   BOOSTING      3  45 539 131 579 228   pingpong
## 4   BOOSTING      4  45 539 131 579 228   pingpong
## 5   BOOSTING      5  49 539 131 579 228   pingpong
## 6   BOOSTING      6  46 539 131 579 228   pingpong
## 7   BOOSTING      7  46 539 131 579 228   pingpong
## 8   BOOSTING      8  40 539 131 579 228   pingpong
## 9   BOOSTING      9  42 539 131 579 228   pingpong
## 10  BOOSTING     10  42 539 131 579 228   pingpong
```

Per ogni video è stato scelto un algoritmo *migliore* trovato *empiricamente*

- pingpong :CSRT
- soccer :CSRT
- car_racing :CSRT
- mucche :CSRT
- aereoplani :MOSSE
- tennis :MOSSE

Metriche di confronto

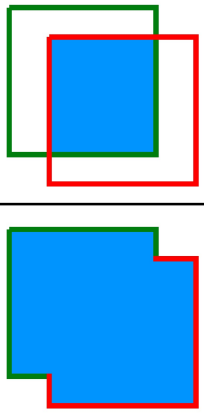
- IoU
- Precision e Recall

Intersection over Union

In object detection problems, IoU evaluates the overlap between ground-truth mask (gt) >and the predicted mask (pd). It is calculated as the area of intersection between gt and pd divided by the area of >the union of the two, that is,

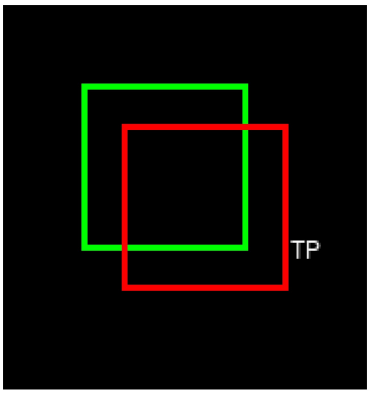
$$\text{IoU} = \frac{\text{area}(gt \cap pd)}{\text{area}(gt \cup pd)}$$

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

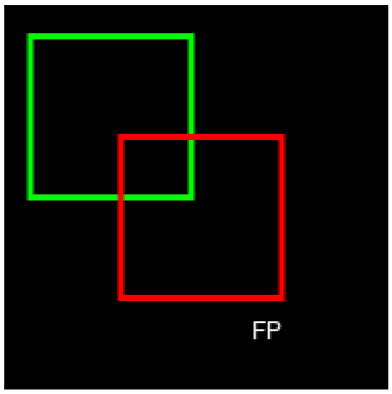


Diagrammatically, IoU is defined as:

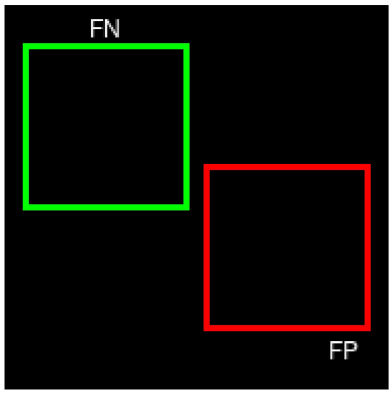
We can, therefore, redefine TP (correct detection) as a detection for which $IoU >= a$ and FP (invalid detection) with $IoU < a$. FN is a ground-truth missed by the model. For example, at IoU threshold, $a = 0.5$ (or 50%), we can define TP, FP and FN as shown in the Fig 4 below.



IoU = 0.86



IoU = 0.24



IoU = 0

Note: If we raise IoU threshold above 0.86, the first instance will be a FP and if we lower IoU threshold below 0.24, the second instance becomes TP.

Funzione di calcolo area overlap, intersezione e IOU

L'input della funzione che calcola l'IoU è $[Ax, Ay, Bx, By]$, un array contenente le coordinate del punto in alto a sinistra del box e di quello in basso a destra.

```
calcolo_area<-function(coord_1){
  return((coord_1[3]-coord_1[1])*(coord_1[4]-coord_1[2]))
}

intersezione_area <- function(coord_1, coord_2) {
  dx <-min(coord_1[3],coord_2[3])- max(coord_1[1],coord_2[1])
  dy<-min(coord_1[4],coord_2[4])- max(coord_1[2],coord_2[2])
  if (dx>=0 & dy>=0){
     return(dx*dy)}
    else{ return(0)}
}

unione_area <-function(coord_1,coord_2){
  return(calcolo_area(coord_1)+calcolo_area(coord_2)-intersezione_area(coord_1,coord_2))
}
IOU <- function(coord_1,coord_2){
  if(unione_area(coord_1,coord_2)!=0){
     return(intersezione_area(coord_1,coord_2)/unione_area(coord_1,coord_2))
  }else
    return(0)}
```

Dataframe unico con numero del frame e coordinate dell'oggetto tracciato da ogni algortimo

> Alcuni algoritmi hanno divere coordinate al primo frame perché quello è il primo frame dove hanno iniziato a tracciare mentre le coordinate dell'oggetto nel frame 0 passate all'algoritmo sono uguali per tutti I frame con coordinate (0,0,0,0) sono quelli dove il tracker non ha trovato l'oggetto

```
##    NFRAME  coord_BOOSTING video_name       coord_KCF       coord_MIL
## 1       1 539 131 579 228   pingpong 539 131 579 228 537 131 577 228
## 2       2 539 131 579 228   pingpong 539 131 579 228 537 131 577 228
## 3       3 539 131 579 228   pingpong 539 131 579 228 540 129 580 226
## 4       4 539 131 579 228   pingpong 539 131 579 228 540 130 580 227
## 5       5 539 131 579 228   pingpong 539 131 579 228 538 130 578 227
## 6       6 539 131 579 228   pingpong 539 131 579 228 540 128 580 225
## 7       7 539 131 579 228   pingpong 539 131 579 228 538 129 578 226
## 8       8 539 131 579 228   pingpong 539 131 579 228 537 131 577 228
## 9       9 539 131 579 228   pingpong 539 131 579 228 537 131 577 228
## 10     10 539 131 579 228   pingpong 539 131 579 228 537 131 577 228
##           coord_TLD       coord_CSRT coord_MEDIANFLOW     coord_MOSSE
## 1  534 141 576 240 539 131 580 230  539 131 578 227 536 128 576 228
## 2  534 141 576 240 539 131 579 228  539 131 578 227 535 128 575 228
## 3  534 141 576 240 539 131 580 230  539 130 579 227 535 128 575 228
## 4  534 141 576 240 540 131 580 228  539 131 579 227 535 128 575 228
## 5  534 141 576 240 540 131 581 230  539 131 579 228 535 128 575 228
## 6  534 141 576 240 539 131 580 230  539 131 579 227 535 128 575 228
## 7  538 188 558 236 539 131 579 228  539 131 578 227 535 128 575 228
## 8  535 177 559 235 539 131 580 230  539 131 578 227 535 128 575 228
## 9  542 177 566 235 539 131 579 228  539 131 578 227 535 128 575 228
## 10 532 165 566 248 539 131 580 230  539 131 579 227 535 128 575 228
```

Dataframe con IoU per ogni frame

```
##    NFRAME video_name IOU_BOOSTING   IOU_KCF   IOU_MIL   IOU_TLD IOU_MOSSE
## 1       1   pingpong    0.9559005 0.9559005 0.8666823 0.6687652 0.8029083
## 2       2   pingpong    1.0000000 1.0000000 0.9047619 0.6679809 0.7958067
## 3       3   pingpong    0.9559005 0.9559005 0.9180962 0.6687652 0.7646157
## 4       4   pingpong    0.9512195 0.9512195 0.9795918 0.6384020 0.7569677
## 5       5   pingpong    0.9102502 0.9102502 0.8501515 0.6391382 0.7279160
## 6       6   pingpong    0.9559005 0.9559005 0.8997368 0.6687652 0.7646157
## 7       7   pingpong    1.0000000 1.0000000 0.9136868 0.1862745 0.7958067
## 8       8   pingpong    0.9559005 0.9559005 0.8666823 0.2414029 0.7646157
## 9       9   pingpong    1.0000000 1.0000000 0.9047619 0.3023715 0.7958067
## 10     10   pingpong    0.9559005 0.9559005 0.8666823 0.3423722 0.7646157
##    IOU_MEDIANFLOW IOU_CSRT
## 1       0.9223947        1
## 2       0.9649485        1
## 3       0.9368139        1
## 4       0.9416499        1
## 5       0.9102502        1
## 6       0.9460458        1
## 7       0.9649485        1
## 8       0.9223947        1
## 9       0.9649485        1
## 10      0.9460458        1
```

Dataframe con IoU per ogni frame con valore di threshold di 0.5

```
##    NFRAME video_name IOU_BOOSTING IOU_KCF IOU_MIL IOU_TLD IOU_MOSSE
## 1       1   pingpong            1       1       1       1         1
## 2       2   pingpong            1       1       1       1         1
## 3       3   pingpong            1       1       1       1         1
## 4       4   pingpong            1       1       1       1         1
## 5       5   pingpong            1       1       1       1         1
## 6       6   pingpong            1       1       1       1         1
## 7       7   pingpong            1       1       1       0         1
## 8       8   pingpong            1       1       1       0         1
## 9       9   pingpong            1       1       1       0         1
## 10     10   pingpong            1       1       1       0         1
##    IOU_MEDIANFLOW IOU_CSRT
## 1               1        1
## 2               1        1
## 3               1        1
## 4               1        1
## 5               1        1
## 6               1        1
## 7               1        1
## 8               1        1
## 9               1        1
## 10              1        1
```

Precision & recall

As started earlier TN are not used in object detection problems and therefore one as to avoid metrics based on this confusion matrix component such as True Negative Rate (TNR), False Positive Rate (FPR), Negative Predictive Value (NPC) and Receiver Operating Characteristic (ROC) curve. Instead, the evaluation of object detection models based on Precision (P) and Recall (R) which are defined as

Precision is the ability of a classifier to identify relevant objects only. It is the proportion of true positive detections. $Precision = tp/(tp + fp) = tp/(detections)$

Recall, on the other hand, measures the ability of the model to find all relevant cases (that is, all ground-truths) — the proportion of true positives detected among all ground-truths. $Recall = tp/(tp + fn) = tp/(ground - truths)$

- A **good model** is a model that can identify most ground-truth objects (**high recall**) while only finding the relevant objects (**high precision**) often.

- A perfect model is the one with FN=0 (recall=1) and FP=0 (precision=1). The former is usually the objective, the latter often unattainable.

False positive: If IoU $<$ 0.5. This means that we are drawing a bounding box around an object but classifying it as another object. Suppose that we are detecting dogs and cats in an image. When the algorithm detects a dog but misclassifies it as a cat, then it is the case false positive. **True positive: if IoU >= 0.5**. This means that there is an object and we have detected it. **False negative** This is the case **when the algorithm does not detect anything when it should be detecting**. For example, there is a dog in the image but the algorithm does not draw any bounding box around it.

Precision e recall misurate per ogni algoritmo

Nel nostro caso i falsi negativi sono tutti i frame in cui l'oggetto non è stato trovato poiché sono video in cui l'oggetto è sempre presente.

```
##          TYPE PRECISION    RECALL
## 1    BOOSTING 0.5558131 1.0000000
## 2         KCF 0.4750869 0.6320658
## 3         MIL 0.7068366 1.0000000
## 4         TLD 0.2286597 0.9472000
## 5       MOSSE 0.5376593 0.8004600
## 6  MEDIANFLOW 0.1525686 0.3542601
## 7        CSRT 0.6852066 0.9833703
```

Bibliografia

https://learnopencv.com/object-tracking-using-opencv-cpp-python/ https://machinelearning4developers.blogspot.com/2021/09/computer-vision.html https://debuggercafe.com/evaluation-metrics-for-object-detection/